Noora Nieminen

Garbling Schemes and Applications

Turku Centre *for* Computer Science

TUCS Dissertations

No 219, March 2017

Garbling Schemes and Applications

Noora Nieminen

To be presented, with the permission of the Faculty of Mathematics and Statistics of the University of Turku, for public criticism in Tauno Nurmela Hall (Lecture Hall I) on March 17, 2017, at 12 noon.

University of Turku
Department of Mathematics and Statistics
FI-20014 Turku, Finland

2017

Supervisors

Professor Valtteri Niemi
Department of Computer Science
University of Helsinki
PL 68 (Gustaf Hällströmin katu 2b)
Helsingin yliopisto
Finland

Senior Researcher Tommi Meskanen
Department of Mathematics and Statistics
University of Turku
FI-20014 Turku
Finland

Reviewers

Professor Vladimir Oleshchuk
Department of Information and Communication Technology
University of Agder
Jon Lilletunsvei 9, Grimstad
Norway

Assistant Professor Billy Brumley
Department of Pervasive Computing
Tampere University of Technology
P.O. Box 15400, FI-33720 Tampere
Finland

Opponent

Professor Benny Pinkas
Department of Computer Science
Bar Ilan University
Ramat Gan
Israel

Painosalama Oy, Turku

ISBN 978-952-12-3515-3
ISSN 1239-1883

Abstract

The topic of this thesis is *garbling schemes* and their applications. A garbling scheme is a set of algorithms for realizing *secure two-party computation*. A party called *a client* possesses a private algorithm as well as a private input and would like to compute the algorithm with this input. However, the client might not have enough computational resources to evaluate the function with the input on his own. The client outsources the computation to another party, called *an evaluator*. Since the client wants to protect the algorithm and the input, he cannot just send the algorithm and the input to the evaluator. With a garbling scheme, the client can protect the privacy of the algorithm, the input and possibly also the privacy of the output.

The increase in network-based applications has arisen concerns about the privacy of user data. Therefore, *privacy-preserving* or *privacy-enhancing* techniques have gained interest in recent research. Garbling schemes seem to be an ideal solution for privacy-preserving applications. First of all, secure garbling schemes hide the algorithm and its input. Secondly, garbling schemes are known to have efficient implementations.

In this thesis, we propose two applications utilizing garbling schemes. The first application provides privacy-preserving *electronic surveillance*. The second application extends electronic surveillance to more versatile monitoring, including also *health telemetry*. This kind of application would be ideal for assisted living services.

In this work, we also present theoretical results related to garbling schemes. We present several new security definitions for garbling schemes which are of practical use. Traditionally, the same garbled algorithm can be evaluated once with garbled input. In applications, the same function is often evaluated several times with different inputs. Recently, a solution based on *fully homomorphic encryption* provides arbitrarily reusable garbling schemes. The disadvantage in this approach is that the arbitrary reuse cannot be efficiently implemented due to the inefficiency of fully homomorphic encryption.

We propose an alternative approach. Instead of arbitrary reusability, the same garbled algorithm could be used a limited number of times. This gives us a set of new security classes for garbling schemes. We prove several relations between new and established security definitions. As a result, we

obtain a complex hierarchy which can be represented as a product of three directed graphs. The three graphs in turn represent the different flavors of security: *the security notion*, *the security model* and *the level of reusability*.

In addition to defining new security classes, we improve the definition of *side-information function*, which has a central role in defining the security of a garbling scheme. The information allowed to be leaked by the garbled algorithm and the garbled input depend on the representation of the algorithm. The established definition of side-information models the side-information of circuits perfectly but does not model side-information of Turing machines as well. The established model requires that the length of the argument, the length of the final result and the length of the function can be efficiently computable from the side-information function. Moreover, the side-information depends only on the function. In other words, the length of the argument, the length of the final result and the length of the function should only depend on the function. For circuits this is a natural requirement since the number of input wires tells the size of the argument, the number of output wires tells the size of the final result and the number of gates and wires tell the size of the function. On the other hand, the description of a Turing machine does not set any limitation to the size of the argument. Therefore, side-information that depends only on the function cannot provide information about the length of the argument. To tackle this problem, we extend the model of side-information so that side-information depends on both the function and the argument. The new model of side-information allows us to define new security classes. We show that the old security classes are compatible with the new model of side-information. We also prove relations between the new security classes.

Tiivistelmä

Tämä väitöskirja käsittelee *garblausskeemoja* ja niiden sovelluksia. Garblausskeema on työkalu, jota käytetään turvallisen kahden osapuolen laskennan toteuttamiseen. *Asiakas* pitää hallussaan yksityistä algoritmia ja sen yksityistä syötettä, joilla hän haluaisi suorittaa tietyn laskennan. Asiakkaalla ei välttämättä ole riittävästi laskentatehoa, minkä vuoksi hän ei pysty suorittamaan laskentaa itse, vaan joutuu ulkoistamaan laskennan toiselle osapuolelle, *palvelimelle*. Koska asiakas tahtoo suojella algoritmiaan ja syötettään, hän ei voi vain lähettää niitä palvelimen laskettavaksi. Asiakas pystyy suojelemaan syötteensä ja algoritminsa yksityisyyttä käyttämällä garblausskeemaa.

Verkkopohjaisten sovellusten kasvu on herättänyt huolta käyttäjien datan yksityisyyden turvasta. Siksi yksityisyyden säilyttävien tai yksityisyyden suojaa lisäävien tekniikoiden tutkimus on saanut huomiota. Garblaustekniikan avulla voidaan suojata sekä syöte että algoritmi. Lisäksi garblaukselle tiedetään olevan useita tehokkaita toteutuksia. Näiden syiden vuoksi garblausskeemat ovat houkutteleva tekniikka käytettäväksi yksityisyyden säilyttävien sovellusten toteutuksessa. Tässä työssä esittelemme kaksi sovellusta, jotka hyödyntävät garblaustekniikkaa. Näistä ensimmäinen on yksityisyyden säilyttävä sähköinen seuranta. Toinen sovellus laajentaa seurantaa monipuolisempaan monitorointiin, kuten terveyden kaukoseurantaan. Tästä voi olla hyötyä etenkin kotihoidon palveluille.

Tässä työssä esitämme myös teoreettisia tuloksia garblausskeemoihin liittyen. Esitämme garblausskeemoille uusia turvallisuusmääritelmiä, joiden tarve kumpuaa käytännön sovelluksista. Perinteisen määritelmän mukaan samaa garblattua algoritmia voi käyttää vain yhdellä garblatulla syötteellä laskemiseen. Käytännössä kuitenkin samaa algoritmia käytetään usean eri syötteen evaluoimiseen. Hiljattain on esitetty tähän ongelmaan ratkaisu, joka perustuu *täysin homomorfiseen salaukseen*. Tämän ratkaisun ansiosta samaa garblattua algoritmia voi turvallisesti käyttää mielivaltaisen monta kertaa. Ratkaisun haittapuoli kuitenkin on, ettei sille ole tiedossa tehokasta toteutusta, sillä täysin homomorfiseen salaukseen ei ole vielä onnistuttu löytämään sellaista. Esitämme vaihtoehtoisen näkökulman: sen sijaan, että samaa garblattua algoritmia voisi käyttää mielivaltaisen monta kertaa, sitä

voikin käyttää vain tietyn, ennalta rajatun määrän kertoja. Tämä näkökulman avulla voidaan määritellä lukuisia uusia turvallisuusluokkia. Todistamme useita relaatioita uusien ja vanhojen turvallisuusmääritelmien välillä. Relaatioiden avulla garblausskeemojen turvallisuusluokille saadaan muodostettua hierarkia, joka koostuu kolmesta komponentista.

Tieto, joka paljastuu garblatusta algoritmista tai garblatusta syötteestä riippuu siitä, millaisessa muodossa algoritmi on esitetty, kutsutaan sivutiedoksi. Vakiintunut määritelmä mallintaa loogisen piiriin liittyvää sivutietoa täydellisesti, mutta ei yhtä hyvin Turingin koneeseen liittyvää sivutietoa. Tämä johtuu siitä, että jokainen yksittäinen looginen piiri asettaa syötteensä pituudelle rajan, mutta yksittäisellä Turingin koneella vastaavanlaista rajoitusta ei ole. Parannamme sivutiedon määritelmää, jolloin tämä ongelma poistuu. Uudenlaisen sivutiedon avulla voidaan määritellä uusia turvallisuusluokkia. Osoitamme, että vanhat turvallisuusluokat voidaan esittää uudenkin sivutiedon avulla. Todistamme myös relaatioita uusien luokkien välillä.

Acknowledgements

I would like to express my appreciation and thanks to my advisor Professor Dr. Valtteri Niemi for guiding and supporting me over the years. I would also like to express my gratitude to my co-advisor Dr. Tommi Meskanen, who has also been a great mentor and supported me whenever I have needed it. I would like to thank both of you for encouraging my research and for allowing me to grow as a research scientist.

I also want to thank Prof. Dr. Niemi and Dr. Meskanen for providing their expertise in co-authoring the publications on which this thesis based. Special thanks also belong to Dr. Arto Lepistö, who provided his expertise in programming to help me to implement an application utilizing garbling.

I am thankful for Professor Vladimir Oleshchuk and Assistant Professor Billy Brumley for the preliminary examination of my work. It was an honor for me that Professor Benny Pinkas agreed to act as the opponent. I am grateful that Professor Juhani Karhumäki has accepted to act as a custos of my dissertation.

I am grateful for funding provided by the Turku Centre for Computer Science (TUCS) and the Academy of Finland project *Cloud Security Services (CloSe)*.

Thanks to the Department of Mathematics and Statistics. I want to thank my colleagues for providing a nice working environment. Especially I want to thank Tuire Huuskonen, Lasse Forss and Sonja Vanto for seamless cooperation in organizing the department exams and solving other administrative problems. I have also enjoyed the numerous scientific and "not so" scientific discussions with my roommates Anne Seppänen and Mari Ernvall.

A special thanks to my family. Words cannot express how grateful I am to my mother, father, grandmother and grandfather for all of the sacrifices that you have made on my behalf. I would also like to thank my sister and her family. Thank you for supporting me for everything, and especially I cannot thank you enough for encouraging me throughout this experience.

List of original publications

I T. Meskanen, V. Niemi, and N. Nieminen. Classes of Garbling Schemes. *Infocommunications journal*, V(3):8–16, 2013

II T. Meskanen, V. Niemi, and N. Nieminen. Hierarchy for Classes of Garbling Schemes. *Studia Scientiarum Mathematicarium Hungarica*, 52(2):1–12, 2015

III T. Meskanen, V. Niemi, and N. Nieminen. Hierarchy for Classes of Projective Garbling Schemes. In *International Conference on Information and Communications Technologies (ICT 2014)*, pages 1–8. IEEE, 2014

IV T. Meskanen, V. Niemi, and N. Nieminen. On Reusable Projective Garbling Schemes. In *2014 IEEE International Conference on Computer and Information Technology (CIT 2014)*, pages 315–322. IEEE, 2014

V T. Meskanen, V. Niemi, and N. Nieminen. Garbling in Reverse Order. In *The 13th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-14)*, pages 53–60. IEEE, 2014

VI T. Meskanen, V. Niemi, and N. Nieminen. Extended Model of Side-Information in Garbling. In *The 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-15)*, pages 950–957. IEEE, 2015

VII T. Meskanen, V. Niemi, and N. Nieminen. How to Use Garbling for Privacy Preserving Electronic Surveillance Services. *Cyber Security and Mobility*, 4(1):41–64, 2015

VIII N. Nieminen and A. Lepistö. Privacy-Preserving Security Monitoring for Assisted Living Services. In *Proceedings of the 17th International Symposium on Health Information Management Research (ISHIMR 2015)*, pages 189–199. York St. John Universty & University of Sheffield, 2015

Contents

Part I

Summary

Chapter 1

Introduction

Today, billions of users are connected to the Internet via various devices like computers, laptops, smartphones and tablets. In addition, devices like washing machines, cameras and televisions can be connected to computer networks. Recent innovations of computer networks have increased the computational power and eased everyday tasks in several ways. Unfortunately, new innovations in the field of computer networks have not been developed without problems. Security and privacy of network-based solutions are among the main concerns.

Attempts to hide secret information have been present as long as people have communicated with each other. Already in the Ancient Rome, Caesar used a cryptographic method to hide messages. Since then, cryptographic methods have evolved greatly, from encryption machines into cryptographic algorithms run on personal computers. The Internet era has brought new challenges to data protection. The Internet has allowed data to be publicly retrievable. However, the question of right to retrieve, possess or process data still remains one of the biggest concerns. It is relatively easy to protect locally stored data on personal computers. A bigger issue is data stored in cloud environment: Who owns the data, who is allowed to access the data and who can only view the data and who can also modify it?

Encryption of data is often used as a method to secure data when it is stored in potentially malicious environment. However, recent advances in distributed computing and cloud computing have enabled also outsourcing computation to potentially untrustworthy parties, which requires further protection methods. There are different approaches to protecting information in such scenario. On one hand, the input data to the computing algorithm should be protected. There are various cryptographic methods that enable this. On the other hand, the algorithm itself may require protection mechanisms. Often, the first aim of an attacker is to understand the behavior of the system or the software. This can be achieved by monitoring

data flow and control flow when various processes in the system are run. Therefore, it is important to have protection mechanisms which prevent an attacker from gaining sensitive information about the algorithm or sensitive parts of it.

During past decades, several proposals for secure computation have been suggested. There are both theoretical and practical solutions for computations on encrypted data. Theoretical solutions often provide strong provable security while being impractical in certain aspects (e.g. fully homomorphic encryption). On the other hand, practice-oriented solutions may not provide as strong theoretical security but these solutions may have efficient implementations that still provide sufficient level of security from the practical point-of-view. Some theoretical solutions also seem to evolve into practical solutions (e.g. garbled circuits).

The main topic of interest in this thesis is secure computation between two or more parties. More specifically, we study garbling schemes which can be used for securely evaluating a function in a potentially insecure environment. We provide both theoretical and practical results: we consider security definitions of garbling schemes and design privacy-preserving applications based on garbling schemes achieving security under certain security definitions.

1.1 Structure of this thesis

This thesis consists of three parts. The first part summarizes the thesis by introducing related research and the contributions of this work. The second part contains the original publications on which this thesis is based. The third part is devoted to proofs that have been omitted from the original publications.

Part I consists of five chapters and the chapters are divided into several sections. Chapter 1 is an introduction to the topic of this work and explains the structure of this thesis.

Chapter 2 is a review of different methods used for securing computations. We start by describing software protection methods, such as *code obfuscation* and *white-box cryptography*. Thereafter, we handle *oblivious transfer* which is a central tool for *secure multiparty computation*. Secure multiparty computation protocols are the topic of Section 2.3.

In Chapter 3 we focus on a specific secure multiparty protocol, *Yao's garbled circuits* and *garbling schemes*. Section 3.1 contains the fundamental security definitions related to secure multiparty computation and related to garbling. Section 3.2 contains a literature review on the research related to Yao's garbled circuits. In Section 3.3, we move to *garbling schemes*, which

are generalizing and formalizing Yao's garbled circuit protocol. Section 3.4 introduces applications for garbling schemes.

Contributions of this thesis in the field of garbling schemes are presented in Chapter 4. This thesis contains both theoretical and practical results. Sections 4.1 – 4.5 are devoted to the theoretical results, whereas section 4.6 introduces proposals for applications using garbling schemes as a privacy-enhancing tool.

Chapter 5 concludes Part I by presenting the conclusions and suggestions for future work.

Part II contains the original publications. There are in total 8 publications on which this thesis is based. The first six papers contain theoretical results related to garbling schemes while the last two papers deal with applications of garbling schemes. Three of the original publications have been published in scientific journals, whereas the rest five have been published in conference proceedings with referee practice.

Part III is a collection of proofs that have been omitted in the original publications, because of strict length limitations of conference proceedings.

1.2 Contributions of the author

All eight papers included in this thesis are joint works. The author of this thesis has contributed to the joint work by writing the first version of each paper based on the joint ideas with the co-authors. In publication VIII, the author of this thesis was entirely responsible for the actual writing of the paper. Arto Lepistö contributed in publication VIII by being responsible for the programming of garbled circuits and the test environment needed for the performance evaluation.

Chapter 2

Secure computation

Software as well as their inputs can be vulnerable to leaking sensitive information. In this section we introduce various methods for securing computation. We begin with software protection methods, which aim at protecting software code or parts of it. Then we present techniques which aim at protecting the computation itself as well as the input to the computation.

2.1 Software protection

As the number of personal computers and other computation devices has increased, *e-piracy*, copying and reselling electronic material (e-books, music, movies, software, games) illegally, has become a serious issue. Many technologies have been developed to tackle electronic piracy acts. For example, *Digital Rights Management* technologies have been developed especially for preventing illegal copying of e.g. computer games, music, films and e-books.

Another threat against computer security is *reverse-engineering*. Reverse-engineering allows "attackers to understand the behavior of software and extract proprietary algorithms and data structures (e.g. cryptographic keys) from it" [148]. Especially, observing control or data flow of software gives valuable information about the software behavior. Therefore, various protection methods to prevent reverse-engineering of software are needed. A widely used solution is to make control and data flow more complex while the observable behavior is still the same. There are also other approaches for software protection.

According to van Oorschot [158], there are four main approaches to software protection. *Code obfuscation* is a technique against reverse-engineering. *White-box cryptography* protects secret keys against malicious hosts. Program integrity is protected by *software tamper resistance* methods. A protection against automated attack scripts and widespread malicious programs is provided by *software diversity* techniques.

Next we give a brief introduction of each of these four protection techniques. Then we discuss code obfuscation and white-box cryptography in more details in sections 2.1.1 and 2.1.2.

Trying to understand software code is often the first task of an attacker since it gives valuable information for further attacks. To gain understanding of a program, one needs to reverse-engineer the software code. Since software code is often intellectual property, the code needs to be protected against such reverse-engineering attempts. As mentioned above, code obfuscation is a method that is designed to protect against reverse-engineering. There are basically two possible ways of obfuscating code: obfuscate data flow or control flow. Data flow can be obfuscated by dead code insertion: the algorithm contains sections which are executed as a part of the original source code but whose results are never used in any other part of the program execution. Control flow can be obfuscated by implanting some auxiliary steps to the code. One of the earliest papers on obfuscation is the one of Cohen [39] in which obfuscation is suggested as a defense method against computer viruses. The first theoretical paper on software protection was from Goldreich and Ostrovsky [73], whose protection of software code was based on encrypting the code. After that, other ideas were proposed, all of which base on the idea of transforming the program code into a code which is functionally equivalent to the original program. The idea of transforming code was first introduced by Collberg et al. in [40, 41]. We provide more detailed discussion on code obfuscation in section 2.1.1.

Unfortunately, code obfuscation does not prevent all attempts of reverse-engineering. For example, code obfuscation does not have method for protecting against *class breaks*. A class break is an attack that is developed for single entity but can easily be extended to break any similar entity. The first scheme against class breaks was proposed by Anckaert et al. [5].

Another approach in software protection is white-box cryptography, which aims at protecting secret keys when evaluating a cryptographic software. The reason for such a protection method arises from the threat of *untrusted host environment* which is related to *malicious host problem*: How should a trusted program be protected from a potentially malicious host. Malicious hosts are a serious threat in *white-box attack context*. In such context, an attacker can freely control the execution platform as well as the software implementation, analyze the binary code and intercept system calls among other actions. In white-box context, extracting cryptographic keys is also a potential threat. White-box cryptography (WBC) is a technology that aims at protecting cryptographic keys in the white-box attack context. We will discuss WBC in section 2.1.2.

Attacks against software integrity are usually followed by reverse-engineering. Software tamper resistance is a method which protects against such violations of integrity. Fundamentals of software tamper-resistance

were provided by Aucsmith [8], who defines software tamper-resistance as ability to resist observation and modification as well as ability to function properly in hostile environments. Aucsmith and Graunke [9] also provided an architecture against tampering, which checks integrity of critical code segments. Since that, different techniques for tamper-resistance have been proposed, see e.g. [41, 34, 87, 33].

The fourth technique for code protection is *software diversity*. The idea of software diversity is very simple, and it is adapted from the nature: genetic diversity protects an entire species from becoming extinct by a single virus or disease. The same idea adapted to the computer world would mean that diversity of software would provide protection against exploitation vulnerabilities and program-based attacks, such as computer viruses. Diversity as a software protection mechanism was first suggested by Cohen [39]. Transforming a software code into a new, functionally equivalent code is the trick against reverse-engineering of one software. However, the same technique can be used for creating several instances of equivalent code, providing a method for software diversity [158]. Other techniques for software diversity can be found in [153, 53, 148].

2.1.1 Code obfuscation

One of the first techniques to protect software code was introduced by Goldreich and Ostrovsky [73]. They introduced the concept of *software-protecting compilers*. Their idea relies on the following idea. The program code is encrypted. A central processing unit (CPU) is able to run the code if it has the corresponding decryption key. The task of an attacker is to reconstruct the code from the encrypted code by executing the program on a *random-access machine* (RAM) on arbitrary inputs and by possibly modifying the data between the CPU and the memory.

Definitions of a code obfuscator proposed after this first proposal do not usually consider encrypted program codes. Next, we provide a definition of a code obfuscator for pseudo-random functions, following [82]. We start with a definition of a *function ensemble*, adapted from [68, p.149] which is needed for the definition of a code obfuscator. For simplicity we consider ensembles of length-preserving functions.

Definition 2.1.1 *Let $\ell : \mathbb{N} \to \mathbb{N}$. A function ensemble is a sequence $\mathcal{F} = \{F_n\}_{n\in\mathbb{N}}$ of random variables such that the random variable F_n assumes values in the set of functions that map bit strings of length $\ell(n)$ to bit strings of the same length.*

A code obfuscator should hide the program code of an algorithm in such a way that the obfuscated code still has the same functionality as the original code. The following definition describes a code obfuscator as a machine

that transforms the original program code into a functionally equivalent, obfuscated program code.

Definition 2.1.2 *Let $\mathcal{F} = \{F_n\}_{n\in\mathbb{N}}$ be a function ensemble. A code obfuscator C for $\mathcal{F}$ is a probabilistic polynomial time ($\mathcal{PPT}$) machine which takes code $\pi(f)$ of a function $f \in F_n$ as input and returns another, functionally identical code $\Pi(f)$ as output.*

Code obfuscator C is said to be secure against adversary $\mathcal{A}$, if everything that $\mathcal{A}$ gains from having access to code $\Pi(f)$ can also be gained by a $\mathcal{PPT}$ machine having only *black-box access* to f. By blackbox-access we mean that an attacker is allowed to see only the input and output functionality of the program and is not allowed to have control on the execution platform or the software implementation. Furthermore, analyzing the binary code or intercepting system calls is not allowed, unlike in white-box context.

Code obfuscators, according to [83], should have two properties:

1. *functionality*, which requires that the obfuscated program has the same functionality as the original

2. *virtual black-box property*: an adversary having access to the obfuscated program can gain same information from the obfuscated program as it would gain by having a black-box access to the functionality.

These two properties are also explicit in the above definition.

It was long an open question whether code obfuscators exist at all. On one hand, Chow [38] presented an obfuscation technique whose resistance relates to a $\mathcal{PSPACE}$-hard problem. Wang states in his PhD thesis [160] that analyzing transformed programs statically is $\mathcal{NP}$-hard. On the other hand, Hada [82] showed the first impossibility result which states that there is no secure obfuscator for a certain class of functions. Another negative result was given by Barak et al. [11, 12], which states that there is no secure obfuscator for general classes of functions represented as logical circuits or Turing machines, when assuming the virtual black-box property.

Also positive results have been reported for special classes of functions. For example, Lynn [112] showed that a reduction between classes of functions imply that if one is obfuscatable then so is the other. Another positive result was reported by Wee [162], who proposed a secure obfuscator for point functions.

Goldwasser and Kalai [76] provided another negative result. They modified the definition of virtual black-box property by including an auxiliary input. Goldwasser and Kalai show that many useful circuits cannot be obfuscated w.r.t. auxiliary input. Barak et al. extended the impossibility results in [11, 12]. In addition, they proposed an alternative definition for virtual black-box property: *indistinguishability obfuscation*. This definition

guarantees that obfuscations of two equal-sized, distinct programs which implement identical functionalities are computationally indistinguishable from each other. The first candidate for indistinguishability obfuscator was proposed by Garg et al. [59]. Recently, Sahai and Waters have proposed a new technique, *punctured programs*, to apply indistinguishability obfuscation towards cryptographic problems. Indistinguishability obfuscators have also had influence on secure multiparty computation [58].

Even though the purpose of code obfuscation is to protect against adversaries performing reverse-engineering, obfuscation can also be used for malicious purposes. Code obfuscation can, for example, be used as method to prevent virus detection. A reliable static detection of a particular class of metamorphic viruses has been shown to be a $\mathcal{NP}$-complete problem [28]. As an application, Borello and Mé [28] present an obfuscating approach which is resilient enough to defeat static analysis tools for metamorphic viruses. A more recent example of obfuscated malware is VirTool:Win32 and its many variants, including the recent exploit of Adobe Flash vulnerabilities [124].

2.1.2 White-box cryptography

Like any algorithm, also cryptographic algorithms are targets of reverse-engineering. The aim of attacking cryptographic algorithms is to recover the cryptographic keys used by the algorithm. Traditionally, the threat models against encryption schemes are *chosen-plaintext attacks* and *chosen-ciphertext attacks*. In these threat models, the attacker has access only to the input and output to the cryptographic algorithm but not to the actual algorithm. This kind of scenario is known as *black-box attacker context*. However, it is also a realistic threat that the attacker has control over the encryption/decryption algorithms, i.e. can access the internal state of these programs. The scenario in which the attacker has the black-box attacker capabilities in addition to the aforementioned capabilities is known as *white-box attacker context*. White-box cryptography provides solutions against white-box attackers by protecting the cryptographic keys.

Some of the first known results concerning white-box cryptography are due to Chow et al. [36, 37]. In these two papers, Chow et al. present a white-box implementation for symmetric key block ciphers Data Encryption Standard (DES) and Advanced Encryption Standard (AES). Their implementation relies on a technique that transforms a cipher into a series of key-dependent look-up tables. The look-up tables contain the hard-coded secret key which is protected by randomization techniques.

Unfortunately, the two WBC implementations proposed by Chow et al. [37, 36] have been broken. The weakness of WBC implementation of DES as well as the WBC implementation of AES is related to the look-up table approach.

The WBC implementation of DES was broken by Goubin [79] and Wyseur [167]. The main reason for the insecurity of the WBC implementation of DES is that it is possible to distinguish the Feistel structure of DES from the look-up table presentation [166].

The WBC implementation of AES was broken by Billet et al. [24]. The attack uses an algebraic cryptanalysis technique against the strategy of randomizing the look-up tables. Wyseur showed that algebraic attacks can be used to defeat the whole look-up table based WBC approach [165]. After these results, other WBC implementations have been proposed, see e.g. [111, 125] for more details.

The first theoretical formulation of security of WBC was proposed by Saxena et al. [147]. They relate code obfuscation to white-box cryptography by defining *white-box property* (WBP) of an obfuscator under certain security notion. They also provide the following impossibility and possibility results. There is a set of programs (e.g. encryption and digital signature schemes) for which certain security notions cannot be satisfied if adversaries have white-box access to the functionality whereas the same notions can be achieved when the adversary has black-box access to the functionality. On the other hand, Saxena et al. show that there is an obfuscator for a symmetric encryption scheme for which *Chosen Plaintext Attack* (CPA) security, among other security measures, is preserved also in the white-box setting. The results of Saxena et al. imply a way of transforming a security notion from black-box context into white-box context, which can also be seen as a way to turn a secure symmetric encryption scheme into a secure asymmetric encryption scheme. The security notions related to WBC have recently been extended by Delerablée et al. [46]. For example, they introduce the concept of *white-box compiler* which turns a symmetric encryption scheme into randomized white-box programs.

2.2 Oblivious transfer

This section is devoted to oblivious transfer, which is an important protocol widely used in other cryptographic protocols. The protocol is run between two parties: *a sender* and *a receiver*. The sender has potentially many pieces of secret information. The receiver would like to learn this secret information. On one hand, the sender wants to reveal only one of the potentially many pieces of information. On the other hand, the receiver does not want the sender to learn which pieces of the secret information he has learned. A protocol that solves this problem is called oblivious transfer.

1:	The sender sends N, e and $m^e \mod N$ to the receiver.
2:	The receiver picks a random $x \in \mathbb{Z}_N$ and sends $z = x^2 \mod N$ to the sender.
3:	The sender computes the square root y of z and sends it to the receiver.

Figure 2.1: Rabin's oblivious transfer protocol

2.2.1 Rabin's oblivious transfer protocol

The first oblivious transfer protocol was proposed by Rabin [139]. This protocol does not directly share pieces of secret information as described above. Instead, in this protocol there is only one secret which is revealed to the receiver with probability $\frac{1}{2}$. Let us now study how Rabin's OT protocol works.

The steps of the protocol are shown in fig. 2.1. Before the actual protocol is started, the sender generates RSA public modulus $N = p \cdot q$ where p and q are large prime numbers. The sender chooses also e relatively prime to $(p-1) \cdot (q-1)$. The sender encrypts the secret message m as $m^e \mod N$.

The sender starts the protocol by sending the modulus N, the public key e and the encrypted message m^e to the receiver. The receiver chooses a random element x from $\mathbb{Z}_N$ and sends $z = x^2 \mod N$ to the sender. Then the sender computes a modular square root of z, i.e. tries to find an element $y \in \mathbb{Z}_N$ such that $y^2 = z \mod N$. The sender then sends y to the receiver.

Now, the receiver can decrypt m^e with probability $\frac{1}{2}$. First of all, there is an overwhelming probability that $\gcd(x, N) = 1$, which implies that there are four square roots of $x^2 \mod N$. If the square root computed at step 3 is x or $-x$, then the receiver cannot obtain information about the secret m. Instead, if $y \neq \pm x$, then the receiver can factorize N by computing $p = \gcd(x + y, N)$, $q = \frac{N}{p}$ and hence recover m.

2.2.2 1-out-of-2 oblivious transfer

A more sophisticated form of oblivious transfer, *1-out-of-2 oblivious transfer*, was proposed by Even, Goldreich and Lempel in [51]. This 1-out-of-2 protocol turns out to be equivalent to Rabin's protocol [43]. Compared to Rabin's OT protocol, there is one fundamental difference. In 1-out-of-2 protocol the sender has two messages m_0 and m_1 whereas in Rabin's protocol there is only one message. In 1-out-of-2 protocol, the sender stays oblivious to which of the two messages is sent. In addition, the receiver does not learn the contents of the other message that he did not receive.

Next, we provide the description of this protocol. 1-out-of-2 protocol is of great significance from secure multi-party computation point-of-view. Kilian [96] showed already in 1988 that it is possible to securely evaluate any

1:	The sender chooses two random values x_0, x_1 and sends them along with (e, N) to the receiver.
2:	The receiver chooses $b \in \{0, 1\}$ and selects x_b based on his choice of b.
3:	The receiver then generates a random value k and computes $v = (x_b + k^e) \mod N$ and sends v to the sender.
4:	The sender computes $k_0 = (v - x_0)^d \mod N$ and $k_1 = (v - x_1)^d \mod N$. Then he combines k_i with message m_i by computing $m'_i = m_i + k_i$ for $i \in \{0, 1\}$. After that, he sends m'_0 and m'_1 to the receiver.
5:	The receiver now chooses m'_b and computes $m_b = m'_b - k$

Figure 2.2: 1-out-of-2 oblivious transfer protocol

$\mathcal{PT}$ function using an implementation of OT protocol without any additional cryptographic primitives. In the next chapter, we discuss a SFE protocol that is based on 1-out-of-2 oblivious transfer, which provides another reason to discuss the 1-out-of-2 protocol in more details.

The sender first generates RSA public/private key pair, i.e. (e, N) and (d, N). Then, both parties follow the protocol presented in fig. 2.2. Note that on step 4, the sender computes two values k_0 and k_1, one of which represents the value k chosen by the receiver. However, the sender does not know which of the two values represent the correct k. On the other hand, on step 5, the receiver is able to recover m_b because he knows the correct k.

2.2.3 Private information retrieval

There are further generalizations of OT protocols - *1-out-of-n OT* and *k-out-of-n OT*. The first of these generalizes transferring one of the two messages into transferring one of n messages to the receiver. In the latter one, some k messages out of the total n messages are transferred to the receiver. The 1-out-of-n OT generalizations have been introduced by several authors, including Aiello, Ishai and Reingold [4], Naor and Pinkas [127] as well as Laur and Lipmaa [101]. The k-out-of-n OT was proposed by Ishai and Kushilevitz [90]. This solution is based on 1-out-of-2 OT protocol. More recently, k-out-of-n OT based on secret sharing schemes have been proposed by Shankar et al. [150] and Tassa [155].

Oblivious transfer is closely related to *private information retrieval* (PIR). PIR allows a user to retrieve an item from a database, let us say of size n, without revealing which item was retrieved. Therefore, PIR can be thought of a weaker version of 1-out-of-n OT, which would in addition require that the user does not learn anything about the $n - 1$ items in the database. It is also sometimes said that 1-out-of-n is a symmetric case of PIR, because of the symmetric nature of hiding information from other party. This is justified by the result of Di Crescenzo et al. [44]: single-database PIR implies 1-out-of-n OT.

2.3 Secure multi-party computation

In this section we present cryptographic protocols that are used for computing security-related functionalities by using cryptographic primitives. The protocols describe how two or more parties should interact in order to achieve a certain (security) goal. There is a wide variety of cryptographic protocols designed for a specific task. These protocols include *entity authentication*, *key agreement*, *secret sharing methods* and *multi-party computation*, among many others. The protocols can be combined to obtain further protocols. As an example, the communications security of modern networks is based on `TLS` protocol, containing protocols for entity authentication and key agreement.

Next we focus on *secure multi-party protocols* (MPC) and *secure function evaluation protocols* (SFE). SFE protocols are a special case of MPC protocols and they are also known as *secure two-party protocols*. In secure multi-party computation, several parties aim at computing a functionality in such way that none of the parties learns more than their own share of input and the output. More formally, secure multi-party computation is defined as follows. Suppose that there are n parties, each having their private input data $d_1, d_2, \ldots, d_n$. Parties would like to compute a value of a jointly agreed functionality f with the private values $d_1, d_2, \ldots, d_n$. Each party should learn the outcome of the computation $f(d_1, d_2, \ldots, d_n)$. In addition, party i should not be able to learn d_j for $j \neq i$. For example, Alice and Bob would like to find out which of them earns more money without revealing their salary to the other party. To solve the problem, Alice and Bob could use a protocol solving the well-known Yao's Millionaire Problem [168].

Zero-knowledge proofs, *homomorphic encryption* and *garbled circuits* are just some examples of secure multi-party computation protocols. We discuss zero-knowledge proofs and homomorphic encryption in brief in this section. Garbled circuits are covered in details in the next chapter.

The basic properties of secure function evaluation protocols are *input privacy*, *correctness*, *fairness* and *validity* [151]. A protocol computing a public function (chosen and known by all the parties) should not reveal d_i to any party $j \neq i$. However, the output of the function might reveal some information about the inputs, without violating the input privacy property. There are two ways to characterize correctness of a secure multi-party protocol. A protocol is called *robust*, if honest parties are able to output the correct result in spite of the fact that any proper subset of the n parties behaves in a malicious manner, e.g. by sharing information or deviating from the protocol. Another approach is to let honest parties abort the protocol if they find that some of the parties is cheating. This kind of protocol is called *MPC protocol with abort*. The fairness property requires that none of the parties should learn the result of the evaluation if they deny sharing

it with the other parties. The validity property in turn requires that the insecure version of the MPC protocol computes the same output as the secure version, given the same inputs. By secure and insecure version of the MPC protocol we mean the following. In the insecure version the protocol run with the real participants. In the secure version of the protocol, the function is computed by a trusted party.

The idea of MPC was introduced by Yao in [168], where he described the famous Millionaire's Problem. Yao's protocol was the first formally introduced two-party computation protocol. Since that, secure computation has been generalized to multi-party setting. Moreover, a wide set of protocols computing different functionalities has emerged ever since. In this section, we present some of these protocols. First, we introduce *zero-knowledge proofs* in which one party tries to convince another party of the validity of a certain statement without revealing anything beyond the validity of the assertion. There are two alternative ways of realizing SFE: *garbled circuits* and *homomorphic encryption*. Homomorphic encryption is presented later in this section whereas garbled circuits are handled in the following chapter.

2.3.1 Zero-knowledge proofs

Zero-knowledge protocols are designed to solve the following scenario. A party called *the prover* has a fact whose truth arises suspicions in another party called *the verifier*. The prover tries to convince the verifier about the fact by generating a proof. However, the prover does not want to reveal the contents of the proof to the verifier. The proof presented by the prover should still convince the verifier.

Zero-knowledge proofs of knowledge have their origins in *interactive proof systems*, first introduced by Goldwasser et al. [78]. In an interactive proof system, a prover has unlimited resources of time and space, whereas the verifier is a $\mathcal{PPT}$ machine. The prover presents a proof that an element x belongs to a language L and the verifier checks the correctness of the proof. The prover and verifier take a polynomial number $p(|x|)$ of interaction steps before the verifier must decide whether $x \in L$, with only a $\frac{1}{3}$ chance of error.

An interactive proof system must fulfill two requirements, which are *completeness* and *soundness*. The completeness of interactive proof system requires that if $x \in L$ then the prover must be able to convince the verifier to accept a certificate of the proof with probability greater than $\frac{2}{3}$. The soundness of interactive proof systems requires that if $x \notin L$, then the verifier is not able to convince the verifier with probability exceeding $\frac{1}{3}$. In other words, the two requirements state that a prover is able to convince the verifier of true statements with overwhelming probability (which is obtained by iteration of the protocol), whereas the verifier will reject a proof of a false statement with overwhelming probability.

Every language in $\mathcal{NP}$ has an interactive proof system [68, p. 194]. Therefore, the class containing all languages having interactive proof systems ($\mathcal{IP}$) contains complexity class $\mathcal{NP}$. Also the class of decision problems solvable by a probabilistic Turing machine in polynomial time and with a bounded error probability ($\mathcal{BPP}$) is included in $\mathcal{IP}$ for a very natural reason (each language in $\mathcal{BPP}$ has a polynomial time verifier that determines the membership without interaction). In addition, Shamir proves that the class of decision problems solvable by a deterministic Turing machine in polynomial space ($\mathcal{PSPACE}$) equals to $\mathcal{IP}$ [149]. This result implies that any language in $\mathcal{PSPACE}$ has an interactive proof system.

A natural question related to interactive proofs is how much knowledge should the prover transfer to the verifier in order to get the verifier convinced of the given statement. It turns out that in some cases it is possible that no knowledge needs to be transferred - these kind of interactive proof systems are called *zero-knowledge interactive proofs*. A proof is said to be zero-knowledge, if any information obtained by efficient computation after interacting with the prover on input $x \in L$ can also be computed from x without any interaction. Formulated in another way, a verifier learns nothing about the proof but the assertion of the statement. In addition, the verifier cannot reconstruct the proof after the interaction with the prover.

There are three flavors of zero-knowledge: *perfect*, *statistical* and *computational*. To define perfect zero-knowledge, let (P, V) be an interactive proof system with computationally unlimited, probabilistic prover P and polynomially bounded, probabilistic verifier. We use here the definition of [68, p. 201]: P is perfect zero-knowledge, if for every $\mathcal{PPT}$ interactive verifier V^* there exists a $\mathcal{PPT}$ algorithm M^* such that for every $x \in L$ the following random variables are identically distributed

- $\langle P, V^* \rangle (x)$, which is the output of the verifier V^* after interacting with prover P
- $M^*(x)$ the output of machine M on input x

The concepts of statistical and computational zero-knowledge relax the requirement of the two distributions being identical. In the case of statistical zero-knowledge, there is a statistically negligible[1] difference between the distributions (in terms of the length of x). Computational zero-knowledge in turn requires that the aforementioned distributions are computationally indistinguishable: there is no efficient algorithm to distinguish between two distributions[2].

The most commonly used definition of zero-knowledge is *auxiliary-input zero-knowledge*. This definition takes into account the information that the

[1] *Negligibility* is formally defined in section 3.1

[2] *Computational indistinguishability* is formally defined in section 3.1.

adversary might have prior to entering interaction with the prover and which can improve the chances of the verifier to gain unwanted knowledge from the prover. The traditional definition proposed by Goldwasser, Micali and Rackoff in [78] was improved by Goldreich et al. in [72], after which auxiliary-input zero-knowledge became the more commonly used definition.

Zero-knowledge proofs were introduced by Goldwasser, Micali and Rackoff in the same paper [78] where they introduced the $\mathcal{IP}$ hierarchy. Goldreich et al. [71] showed that all languages in $\mathcal{NP}$ have a zero-knowledge proof by assuming unbreakable encryption. Their proof shows how to create a zero-knowledge interactive proof system for the graph coloring problem with three colors, which is an $\mathcal{NP}$-complete problem (so any other problem in $\mathcal{NP}$ is efficiently reducible to this problem). In a different paper, Goldreich et al. [21] show that anything that can be proved by an interactive proof system can be proved by a zero-knowledge proof, assuming only existence of one-way functions.

There is also a variant of zero-knowledge proofs that does not require interaction between the prover and the verifier - *non-interactive proofs* (NIZK) introduced by Blum et al. [26]. A NIZK proof system, for input $x \in L$, with witness w, consists of three algorithms: *Key Generator*, prover and verifier. The first of these algorithms generates a reference string σ used by both the prover and the verifier. Prover algorithm generates the proof π based on the reference string, x and the witness w. Verifier either accepts or rejects the proof by using σ, x and π as its inputs. Non-interactive zero-knowledge proofs are used for digital signatures and message authentication [15] as well as voting [80], among many other applications.

Zero-knowledge proofs can be composed in three ways: *sequential*, *parallel* and *concurrent*. In sequential composition, the protocol is invoked polynomially many times, where the next protocol is invoked after the previous has terminated. It was shown in [68] that the protocol in [78] is not closed under sequential composition, whereas the auxiliary-input ZK protocol is [70]. In parallel composition, polynomially many instances of ZK protocol are invoked at the same time and the protocols proceed at the same pace. According to [67, 49], there exist zero-knowledge protocols for $\mathcal{NP}$ that are closed under parallel composition. In general, zero-knowledge is not closed under parallel composition. Concurrent composition is a generalization of the other two types of composition. In the concurrent composition, polynomially many zero-knowledge proofs can be invoked at arbitrary times and proceed at arbitrary pace. There are two models for concurrent composition: purely asynchronous model and asynchronous model with timing. The traditional way of defining zero-knowledge proofs yielded the first known impossibility results in purely asynchronous model, even in the auxiliary-input model of zero-knowledge proofs [69, 32].

Until 2001, all zero-knowledge proofs relied on *black-box simulation*. A proof is called zero-knowledge, if for every verifier strategy V^* there exists a simulator that uses algorithm V^* as random oracle (i.e. as a black-box). However, Barak showed that the algorithm V^* can be used by the simulator also in non-black-box manner [10], which broke some of the impossibility results related to concurrent purely asynchronous zero-knowledge. Barak's non-black-box technique is also closely connected to code obfuscation.

Zero-knowledge proofs are also connected to MPC and SFE protocols. In presence of an adversary who deviates from the protocol, there is a need for a mechanism which ascertains that the cheating party has not gained unwanted information by cheating. The earliest solutions used zero-knowledge proofs as a countermeasure. However, due to ineffectiveness of constructing zero-knowledge proofs, other methods were introduced.

2.3.2 Homomorphic encryption

In this section, we focus on homomorphic encryption. The aim of homomorphic encryption is to perform computations on encrypted data. Performing computations on encrypted data is advantageous. For example, it is more secure to outsource computation to a party if the input or the algorithm is not revealed to the party.

There are three types of homomorphic encryption: *group homomorphic encryption*, *somewhat homomorphic encryption* and *fully homomorphic encryption*. Briefly, the three types of homomorphic encryptions differ in the following way. Group homomorphic encryption enables arbitrary computations based on only one binary group operation. Somewhat homomorphic encryption enables arbitrary computations with one binary group operation and limited number of computations with another binary group operation. Fully homomorphic encryption in turn enables arbitrary number of computations with two different binary operations in the set, enabling computations on arbitrary functions.

Next, we briefly discuss each three types of homomorphic encryption. To do this, we need some definitions. We start by defining what is *a group homomorphism*.

Definition 2.3.3 *Let G be a group with binary operation $\cdot$ and let G' be another group with binary operation $\circ$. A group homomorphism from $(G, \cdot)$ to $(G, \circ)$ is a function $h : G \to G'$ such that for all elements $x_1, x_2 \in G$ it holds that*

$$h(x_1 \cdot x_2) = h(x_1) \circ h(x_2).$$

The history of computing on encrypted data traces back to 1978, when Rivest et al. [142] introduced the term *privacy homomorphism* which could

do the task. By privacy homomorphism we mean an encryption function $\mathcal{E} : (G, \cdot) \to (G', \circ)$ which is a group homomorphism satisfying

$$\mathcal{E}(x_1, pk) \circ \mathcal{E}(x_2, pk) = \mathcal{E}(x_1 \cdot x_2, pk),$$

where pk is the public key used for encrypting group elements x_1 and x_2.

It was shown by Brickell and Yacobi [30] that all other privacy homomorphisms presented in [142] but RSA were insecure. Unpadded RSA was the first example of a group homomorphic encryption. The homomorphic property of unpadded RSA is shown below:

$$\mathcal{E}(x_1 \cdot x_2, e) = (x_1 \cdot x_2)^e = x_1^e \cdot x_2^e = \mathcal{E}(x_1, e) \cdot \mathcal{E}(x_2, e) \mod n.$$

Another example of a multiplicative homomorphic encryption scheme is ElGamal cryptosystem [50]. In ElGamal cryptosystem, the encryption of x is $\mathcal{E}(x) = (g^r, x \cdot h^r)$, where g is a generator in a cyclic group G of order q, $h = g^{sk}$ (sk denotes the secret key) and r is a randomly chosen value from the set $\{0, 1, \ldots q-1\}$. The public key is $pk = (G, q, g, h)$ The homomorphic property of ElGamal is then

$$\begin{aligned}\mathcal{E}(x_1, pk)\mathcal{E}(x_2, pk) &= (g^{r_1}, x_1 \cdot h^{r_1})(g^{r_2}, x_2 \cdot h^{r_2}) \\ &= (g^{r_1+r_2}, (x_1 \cdot x_2) \cdot h^{r_1+r_2}) = \mathcal{E}(x_1 \cdot x_2, pk).\end{aligned}$$

There are also several additive homomorphic encryption functions, like the ones proposed by Goldwasser-Micali [77], Benaloh [22](which is an extension of Goldwasser-Micali cryptosystem allowing blocks of bits to be encrypted) and Paillier [135] cryptosystems. Let us consider Goldwasser-Micali cryptosystem as an example of an additive homomorphic encryption. In Goldwasser-Micali cryptosystem, the public key x is a quadratic non-residue modulo n, where n is a composition of two prime numbers p and q. The encryption of a bit b is $\mathcal{E}(b, x) = x^b \cdot r^2 \mod n$ where r is a random value in $\{0, 1, \ldots n-1\}$. The homomorphic property in $(\mathbb{Z}_2, \oplus)$, where $\oplus$ is the binary operation corresponding to logical exclusive-OR, is then

$$\begin{aligned}&\mathcal{E}(b_1, x) \cdot \mathcal{E}(b_2, x) = x^{b_1} r_1^2 \cdot x^{b_2} r_2^2 \\ &= x^{b_1+b_2}(r_1 r_2)^2 = \mathcal{E}(b_1 \oplus b_2, x) \mod n.\end{aligned}$$

Other well-known homomorphic encryptions can be found in [126, 133, 45, 91].

After finding several secure homomorphic encryptions, a natural question was asked: is it possible to find an encryption scheme that would be homomorphic on both additive and multiplicative operations. This was an open question until Gentry [63] presented the first construction of a fully-homomorphic encryption in 2009. Before that, there were several attempts

to find even somewhat homomorphic encryption schemes, which allow arbitrarily many computations of one operation and limited number of computations of the other operation. Boneh-Goh-Nissim [27] was the first somewhat homomorphic encryption scheme, based on bilinear pairings of elliptic curve groups. Boneh-Goh-Nissim allows arbitrary number of additions but only one multiplication. Also other somewhat homomorphic encryptions have been proposed [146, 114, 66].

The major breakthrough in the research of homomorphic encryption was the discovery of the first fully-homomorphic encryption scheme which allows arbitrarily many additions and multiplications. Gentry's construction [63] uses ideal lattices and is based on the hardness of *learning with errors* (LWE) problem (see [141]). Gentry's idea is to transform a somewhat homomorphic encryption scheme into a *bootstrappable scheme*. This bootstrappable scheme can then be transformed into fully-homomorphic scheme. Bootsrapping is a method used for reducing the noise of LWE-based ciphertexts. Unfortunately, bootstrapping technique is quite costly and increases the computational overhead. In [61], some methods to overcome this issue are presented, either by removing the need of bootstrapping (by using quantum error correction) or by eliminating the noise overall (Nuida's pure noise-free FHE using non-abelian groups [131]).

The first working implementation of Gentry's fully-homomorphic scheme was presented in [64]. Recently, another implementation based on AES block cipher was provided for a more recent fully-homomorphic scheme. This scheme uses approximately 40 minutes to evaluate a block [65]. At the present, fully-homomorphic encryption schemes are still considered to be impractical.

An attractive alternative for fully-homomorphic encryption is Yao's garbled circuits and other garbling techniques. The most important reason is the fact that garbled circuits have efficient implementations: for instance, JustGarble [94] garbles and evaluates an AES128 circuit with 36 500 gates in roughly 0.1 milliseconds, compared to the 40 minutes/block if FHE were used [65].

Garbling schemes, of which garbled circuits are a special case, are the main topic of this work. We extend the security notions for practical purposes and study the relations of the various security notions. We also introduce two scenarios in which garbling schemes can be naturally used in a context combining Internet-of-Things technologies with cloud services.

Chapter 3

Garbling techniques

In previous chapter we introduced several ways of protecting algorithms against various threats. Code-obfuscation is designed to protect algorithms from reverse-engineering. White-box cryptography protects the private keys when executing cryptographic algorithms. These two methods protect the code of the algorithm (i.e. the functionality to be computed). Secure multiparty protocols in turn aim at protecting the private inputs of the parties willing to evaluate a publicly chosen functionality, without a need for a trusted party. We presented zero-knowledge proofs and homomorphic encryption as examples of MPC protocols in the previous section. In the strict sense, zero-knowledge proofs are not used for securely evaluating a function. Instead, one party generates a proof which is verified by another party in the zero-knowledge protocol. Homomorphic encryption comes closer to the topic of interest - homomorphic encryption can be used for evaluating algorithms on encrypted data.

Unfortunately, there are no efficient implementations known for FHE. In this chapter we introduce an alternative cryptographic primitive for secure function evaluation which can be efficient and can be used for protecting both the algorithm (function) and its input (argument). We start with the well-known two-party protocol, *garbled circuits*, first introduced by Yao in [168, 169]. Then we present related research concentrating on implementations and optimizations of garbled circuit protocol. Garbling technique can also be applied to functions represented in computation models other than circuits. This idea leads to a generalization of garbled circuits, called *garbling schemes*.

3.1 Basic security definitions

We start this section by presenting basic definitions and notations. We first need the concepts of negligibility and computational indistinguishability.

Then we discuss security definitions used for garbling protocols, especially those related to garbled circuits.

Definition 3.1.4 *Let $\mu(c)$ be a function $\mu : \mathbb{N} \to \mathbb{R}$. Function μ is a negligible function, if for any positive integer c there exists an integer N_c such that for all $x > N_c$*

$$|\mu(x)| < \frac{1}{x^c}.$$

The following definition formalizes the concept of two probability distributions "looking the same". To tell two distributions apart, we use a computationally bounded algorithm called *a distinguisher*. The notation $x \leftarrow X$ means that an element from distribution X is assigned to variable x. In this context, notation $\mathtt{A}(x) = 1$ is used for denoting that the distinguisher algorithm $\mathtt{A}$ is evaluated with input x and it outputs *a decision bit* $\mathtt{A}(x)$ with value 1.

In the definition below, we use the concept of *non-uniform probabilistic polynomial-time algorithm*. Informally, a non-uniform probabilistic polynomial time algorithm is a Turing machine which is allowed to seek *advice* before it outputs its decision. Using other words, a non-uniform Turing machine takes two inputs: an input of length n and an advice of length polynomial in n. Traditional Turing machines are uniform in the sense that they work on all inputs of arbitrary lengths. Supplying an advice whose length depends on the length of the input makes the Turing machine non-uniform, since the machine uses different instructions for different input lengths.

Definition 3.1.5 *Let $E = \{E_k\}_{k\in\mathbb{N}}$ and $D = \{D_k\}_{k\in\mathbb{N}}$ be two distribution ensembles, indexed by security parameter k. Ensembles E and D are* computationally indistinguishable *if for any non-uniform probabilistic polynomial-time (nu$\mathcal{PPT}$) algorithm $\mathtt{A}$, the quantity $\delta(k)$ defined below is a negligible function in k.*

$$\delta(k) = \left| \Pr_{x\leftarrow D_k}[\mathtt{A}(x) = 1] - \Pr_{x\leftarrow E_k}[\mathtt{A}(x) = 1] \right|$$

Computational indistinguishability of E and D is denoted by $E \approx D$.

In the above definition, the quantity $\delta(k)$ can be considered as the success probability that a *nu$\mathcal{PPT}$* distinguisher is capable of distinguishing the distribution ensemble E from distribution ensemble D. Ensemble E is computationally indistinguishable from D, if the success probability $\delta(k)$ is negligible in k.

Next we discuss security definitions used with secure multi-party computation and with garbled circuits. Traditionally, the adversarial models used in secure multi-party computation are based on *the full simulation ideal/ real-model paradigm* [107]. This paradigm has two adversaries: a real-model adversary and an ideal-model adversary. In the ideal-model, the protocol

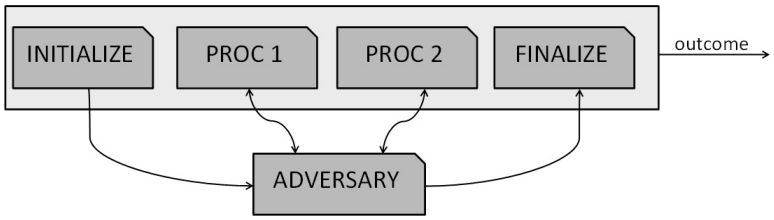


Figure 3.1: A game consists of codes for `INITIALIZE`, `FINALIZE` and other named procedures. An adversary playing the game is also treated as code which can interact with the other named procedures.

is run with a trusted party. The trusted party computes the function and sends the outputs to the original parties. In the real-model, there is no trusted party. The protocol is run with the original parties. A protocol is said to be secure, if the result of a real execution of f is computationally indistinguishable from the result of an ideal execution, where f is computed by the trusted party.

There are also different models for adversaries. A *semi-honest adversary* follows the protocol but he also controls one of the protocol parties in order to gain more information than intended by the protocol specification. A *malicious adversary* can deviate from the protocol and run any efficient strategy to carry out his attack. A *covert adversary* is a model between semi-honest and malicious adversarial models. A covert adversary cheats always when there is no fear of getting caught. In this context, cheating means that the adversary is able to do something that is impossible in the ideal model. The fear of getting caught is regulated by a concept called *deterrence factor*, a value between 0 and 1. Deterrence factor tells the probability that the honest parties detect any attempt to cheat by a real adversary. For more formal definitions of different adversarial models, see [84, Chapter 2].

A traditional way to formalize security of cryptographic protocols is to use the ideal/real paradigm which we have already described above. There is also an alternative way of formalizing security for cryptographic protocols. This approach is called *code-based game-playing*. Code-based game-playing proof is any proof in which adversary's interaction with its environment is conceptualized as a certain type of game [20]. Following the terminology presented in [20], a game is a collection of procedures (or procedure codes). This collection of procedures may contain three types of procedures: `INITIALIZE`, `FINALIZE` and other named procedures. The word "may" is used, since all the procedures in a game are optional.

The entity playing a game is called *adversary*. When the game is run with an adversary, first the `INITIALIZE` procedure is called. It possibly provides an input to the adversary, who in turn may invoke other procedures before the `FINALIZE` procedure. The game ends when the adversary gives an input

to `FINALIZE` procedure which then creates a string that tells the outcome of the game typically consisting of one bit of information: whether the adversary has won the game or not. This description of code-based games is quite informal and gives only the intuition behind the concept. Figure 3.1 serves as an illustration of a code-based game. For a more formal description, we refer to [20].

In this work, we use the code-based game-playing approach instead of semi-honest/malicious adversarial models. We share the ideology of [20, 18] that the code-based game playing approach makes proofs more unified by their structure. Moreover, it is easier to verify the correctness of proofs using the code-based game-playing approach. In addition, code-based games make comparing the different security notions for garbling schemes easier.

3.2 Yao's garbled circuits

Yao presented the idea of secure function evaluation in two seminal papers [168, 169]. However, Yao neither used the term "garbled circuit" nor provided an implementable protocol in these papers. The protocol was formally documented later by other researches [70, 14]. Several applications have utilized garbled circuits, even though a formal proof of security has been missing. The first proof was provided by Lindell and Pinkas in presence of semi-honest adversaries [106]. Later, the protocol has been improved and optimized to be secure also in presence of malicious adversaries. We will discuss these results later in this section.

Next, we provide an informal description of Yao's garbled circuit protocol. Then we discuss the details of the protocol that are related to an efficient and secure implementation of such a protocol.

1:	Bob generates a logical circuit presentation C_c of function f which takes input i_B from Bob and input i_A from Alice
2:	Bob creates garbled circuit C_g
3:	Bob sends C_g and the garbled input I_B to Alice
4:	Alice uses 1-out-of-2 oblivious transfer to get the garbled values I_A from Bob
5:	Alice evaluates the garbled circuit C_g with garbled inputs I_A and I_B and outputs the result.

Figure 3.2: Informal description of Yao's garbled circuit protocol

Constructing the garbled circuit includes the following steps. For each gate g in the circuit, the truth table is first transformed into garbled truth table by generating six keys k_i^α, two for each three wires in the gate. Here α represents the two possibilities 0 and 1 for the value of the wire i. The

six keys are generated randomly and independently using a separate key generation algorithm. The next step is to generate an encrypted truth table. The steps to create an encrypted truth table based on the original truth table are shown in fig. 3.3. Finally, the encrypted rows e_g^{00}, e_g^{01}, e_g^{10} and e_g^{11} of the truth table are shuffled using a random permutation. The resulting values e_g^0, e_g^1, e_g^2 and e_g^3 now form the garbled truth table for gate g. Permuting the rows of the encrypted truth table ensures that Alice does not know which row in the garbled table corresponds to which row in the original truth table. For a circuit C, each gate is treated in a similar manner. As a result, we obtain a garbled circuit, denoted by $G(C)$. Notice that also a `NOT` gate, which has only one incoming wire, can be implemented as a gate taking two incoming wires: a `XOR` gate with constant 1 as one of the incoming wires.

Truth table for gate g

w_1	w_2	w_3
0	0	$g(0,0)$
0	1	$g(0,1)$
1	0	$g(1,0)$
1	1	$g(1,1)$

Keys for encrypted truth table

w_1	w_2	w_3
k_1^0	k_2^0	$k_3^{g(0,0)}$
k_1^0	k_2^1	$k_3^{g(0,1)}$
k_1^1	k_2^0	$k_3^{g(1,0)}$
k_1^1	k_2^1	$k_3^{g(1,1)}$

Encrypted truth table

$$e_g^{00} = \mathtt{En}_{k_1^0}(\mathtt{En}_{k_2^0}(k_3^{g(0,0)}))$$
$$e_g^{01} = \mathtt{En}_{k_1^0}(\mathtt{En}_{k_2^1}(k_3^{g(0,1)}))$$
$$e_g^{10} = \mathtt{En}_{k_1^1}(\mathtt{En}_{k_2^0}(k_3^{g(1,0)}))$$
$$e_g^{11} = \mathtt{En}_{k_1^1}(\mathtt{En}_{k_2^1}(k_3^{g(1,1)}))$$

Garbled truth table (an example)

$$e_g^0 = e_g^{01} = \mathtt{En}_{k_1^0}(\mathtt{En}_{k_2^1}(k_3^{g(0,1)}))$$
$$e_g^1 = e_g^{11} = \mathtt{En}_{k_1^1}(\mathtt{En}_{k_2^1}(k_3^{g(1,1)}))$$
$$e_g^2 = e_g^{00} = \mathtt{En}_{k_1^0}(\mathtt{En}_{k_2^0}(k_3^{g(0,0)}))$$
$$e_g^3 = e_g^{10} = \mathtt{En}_{k_1^1}(\mathtt{En}_{k_2^0}(k_3^{g(1,0)}))$$

Figure 3.3: Transforming a truth table into a garbled truth table.

Bob's garbled input to the garbled circuit are the keys for the wires that correspond to his input in the original circuit. Alice knows her input but not the keys for them. Without the keys, Alice cannot correctly evaluate the garbled circuit. In order to get the keys from Bob, Alice and Bob use 1-out-of-2 oblivious transfer. Suppose that Alice wants to find key k_i^b for her input wire i which is assigned with bit b. Bob knows both k_i^0 and k_i^1 but does not reveal both of them. Moreover, Bob should not learn which of the two keys Alice is requesting; otherwise he would learn Alice's input. The two keys k_i^0 and k_i^1 are Bob's secret messages in the protocol. The 1-out-of-2 protocol is run as explained in fig. 2.2. In the protocol, Alice chooses the challenge based on the bit b which guarantees that she learns k_i^b in the end of the protocol.

Alice and Bob need to run 1-out-of-2 OT protocol as many times as is the length of Alice's input. If Alice's input consists of n bits, then the OT protocol needs to be run n times before Alice can start evaluating the garbled circuit with the given inputs.

Alice evaluates a garbled gate by decrypting the four elements in the garbled truth table using the keys for the two incoming wires. Only one of the four options should return a correct decrypted value; the others should return $\perp$, a symbol used for failure. This requires a special kind of encryption algorithm which provides also authentication: when wrong keys are used then the authentication part in the decryption process fails.

In summary, Alice does not learn any intermediate values, only the keys representing the intermediate values. Similarly, the output consists of keys corresponding to the outgoing wires in the circuit but Alice does not know which bits the keys represent.

In the final step, Alice shares the result, the keys for the outgoing wires, with Bob. Now Bob needs to reveal the final result of the evaluation by revealing whether the keys Alice sent represent bit 0 or 1. It is possible that Bob is behaving maliciously and refuses to reveal the decrypted result to Alice. In the protocol, there is no mechanism to prevent Bob from deviating from the protocol in the end. Also Alice can refuse to reveal the keys for Bob but then neither learns the outcome because only Bob knows the correspondence between a key and the bit value.

Let us next give an example to demonstrate how Yao's garbled circuit protocol works. For simplicity, we consider a circuit with only three logic gates and one output bit representing a result of a certain decision problem.

Example 1 Alice and Bob need to choose one out of four candidates who should get a grant. The candidates are numbered from 0 to 3. Alice and Bob need to determine whether they chose the same candidate but if this is *not* the case then Alice must not learn Bob's choice and vice versa. To achieve this goal, Alice and Bob agree on using garbled circuit protocol.

Let the input from Alice be i_A and the input from Bob be i_B. The function $f(i_A, i_B)$ computes whether the choice of Alice coincides with the choice of Bob, i.e. whether $i_A = i_B$. Let us assume that Alice has chosen candidate 2 whereas Bob has chosen candidate 1. The function f should return $f(2, 1) = 0$ which means that the choices differ.

Step 1: Bob generates a logical circuit presentation for function f, taking input i_A from Alice and i_B from himself. The inputs i_A and i_B are encoded as two-bit strings. For example, 10 represents Alice's choice 2 and 01 represents Bob's choice 1. Let $x_A y_A$ be the two-bit representation for i_A and let $x_B y_B$ be the representation for i_B, respectively. The function f now can be presented as a circuit C as shown in fig. 3.4. The circuit takes input $x_A x_B y_A y_B$ and outputs one bit. The equality of the first bits, $x_A = x_B$, is

checked with a XOR gate and the equality of the second bits, $y_A = y_B$, is checked with another XOR gate. Finally, a NOR gate is used for determining whether the first bits and the second bits were equal. The output bit 1 yields a positive result (the choices are the same) and output bit 0 yields a negative result (the choices differ).

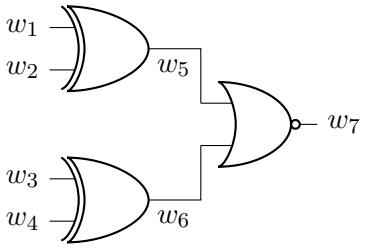


Figure 3.4: The circuit for testing the equality consists of two XOR gates and one NOR gate.

Let us first justify why the given circuit correctly evaluates the function f. With the input 10 from Alice and 01 from Bob, the circuit is evaluated with input 1001. The first XOR gate returns 1 since the bits are different. Also the second XOR gate returns 1 for the same reason. Finally, the NOR gate returns 0, because $\texttt{NOR}(1,1) = 0$. As another example, let the input from Alice be 10 and let the output from Bob be 10 as well. Now the circuit takes input 1100. Both XOR gates output 0. Now, the NOR gate outputs 1, which is correct since both Alice and Bob chose the same candidate 2. The 14 other cases can be checked in a similar manner.

Step 2: Bob creates garbled circuit $G(C)$ for the circuit in fig. 3.4. He starts by generating two keys, k_i^0 and k_i^1,for each wire $i \in [1, 7]$, as shown in fig. 3.5.

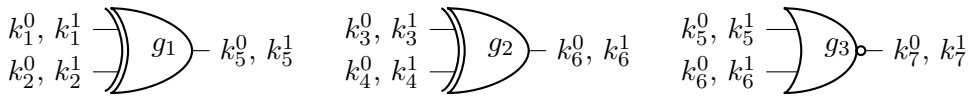


Figure 3.5: Each wire w_i is associated with two keys, k_i^0 and k_i^1.

Then Bob creates encrypted truth tables for all three gates. These encrypted truth tables are shown in fig. 3.6. Bob creates four ciphertexts for each gate g_1, g_2 and g_3. He uses the keys $k_1^0, k_1^1, k_2^0, k_2^1$ associated with the input wires to encrypt the key for the gate g_1's output wire w_5. In a similar manner he encrypts output wires k_6 and k_7 using the associated keys for the input wires of the appropriate gates.

Then Bob shuffles the rows of each encrypted truth table by using a random permutation. For example, after applying a random permutation the garbled truth tables may appear as in fig. 3.7.

$$
\begin{array}{lll}
e_{g_1}^{00} = \mathtt{En}_{k_1^0}(\mathtt{En}_{k_2^0}(k_5^0)) & e_{g_2}^{00} = \mathtt{En}_{k_3^0}(\mathtt{En}_{k_4^0}(k_6^0)) & e_{g_3}^{00} = \mathtt{En}_{k_5^0}(\mathtt{En}_{k_6^0}(k_7^1)) \\
e_{g_1}^{01} = \mathtt{En}_{k_1^0}(\mathtt{En}_{k_2^1}(k_5^1)) & e_{g_2}^{01} = \mathtt{En}_{k_3^0}(\mathtt{En}_{k_4^1}(k_6^1)) & e_{g_3}^{01} = \mathtt{En}_{k_5^0}(\mathtt{En}_{k_6^1}(k_7^0)) \\
e_{g_1}^{10} = \mathtt{En}_{k_1^1}(\mathtt{En}_{k_2^0}(k_5^1)) & e_{g_2}^{10} = \mathtt{En}_{k_3^1}(\mathtt{En}_{k_4^0}(k_6^1)) & e_{g_3}^{10} = \mathtt{En}_{k_5^1}(\mathtt{En}_{k_6^0}(k_7^0)) \\
e_{g_1}^{11} = \mathtt{En}_{k_1^1}(\mathtt{En}_{k_2^1}(k_5^0)). & e_{g_2}^{11} = \mathtt{En}_{k_3^1}(\mathtt{En}_{k_4^1}(k_6^0)). & e_{g_3}^{11} = \mathtt{En}_{k_5^1}(\mathtt{En}_{k_6^1}(k_7^0)).
\end{array}
$$

Figure 3.6: The encrypted truth tables for gates g_1, g_2 and g_3.

$$
\begin{array}{lll}
e_{g_1}^0 = e_{g_1}^{00} & e_{g_2}^0 = e_{g_2}^{10} & e_{g_3}^0 = e_{g_3}^{00} \\
e_{g_1}^1 = e_{g_1}^{01} & e_{g_2}^1 = e_{g_2}^{00} & e_{g_3}^1 = e_{g_3}^{10} \\
e_{g_1}^2 = e_{g_1}^{11} & e_{g_2}^2 = e_{g_2}^{11} & e_{g_3}^2 = e_{g_3}^{01} \\
e_{g_1}^3 = e_{g_1}^{10}. & e_{g_2}^3 = e_{g_2}^{01}. & e_{g_3}^3 = e_{g_3}^{11}.
\end{array}
$$

Figure 3.7: Garbled truth tables for gates g_1, g_2 and g_3. The permutations used in this example were obtained by using a random sequence generator at `random.org`.

Bob has now generated three garbled truth tables $\left\{e_{g_1}^0, e_{g_1}^1, e_{g_1}^2, e_{g_1}^3\right\}$, $\left\{e_{g_2}^0, e_{g_2}^1, e_{g_2}^2, e_{g_2}^3\right\}$ and $\left\{e_{g_3}^0, e_{g_3}^1, e_{g_3}^2, e_{g_3}^3\right\}$ which now form the garbled circuit $G(C)$.

Step 3: Bob sends the garbled circuit $G(C)$ and his garbled input $I_B = (k_2^{x_B}, k_4^{y_B})$ to Alice. Recall that Bob's choice was candidate 1, so his input to the function is 01. Therefore, the garbled input sent by Bob is $I_B = (k_2^0, k_4^1)$.

Step 4: To evaluate the circuit, Alice first needs the keys corresponding to her inputs, i.e. keys $k_1^{x_A}$ and $k_3^{y_A}$. She starts a series of 1-out-of-2 oblivious transfer protocols with Bob to get the needed keys. The keys k_i^0 and k_i^1 for wire w_i act as Bob's secrets. Alice's choice of b in each round of 1-out-of-2 OT protocol is based on the input values x_A and y_A.

As an example, consider the case of g_1 where Alice needs key k_1^1, corresponding to input $x_A = 1$. Bob's secrets are k_1^0 and k_1^1. Alice chooses $b = 1$ in the 1-out-of-2 OT protocol and if the both participants correctly follow the protocol shown in fig. 2.2 then Alice learns k_1^1. In similar way, Alice receives k_3^0 via another run of 1-out-of-2 OT protocol from Bob.

Step 5: Alice starts evaluating the circuit with keys k_1^1, k_2^0, k_3^0 and k_4^1. She first decrypts values $e_{g_1}^0$, $e_{g_1}^1$, $e_{g_1}^2$, $e_{g_1}^3$ with keys k_1^1 and k_2^0. Only one of the

decryption attempts is successful, namely $e_{g_1}^3$, returning k_5^1. This is the case, because $e_{g_1}^3 = e_{g_1}^{10} = \mathtt{En}_{k_1^1}(\mathtt{En}_{k_2^0}(k_5^0))$. Other three decryptions yield $\perp$. In similar way, Alice obtains k_6^1 by decrypting the values $e_{g_2}^0$, $e_{g_2}^1$, $e_{g_2}^2$ and $e_{g_2}^3$ with keys k_3^1 and k_4^0. In this case, the output k_6^1 is obtained from decrypting value $e_{g_2}^0$ whereas the other three decryptions yield $\perp$. As the final step, Alice finds out k_7^0 by decrypting values $e_{g_3}^0$, $e_{g_3}^1$, $e_{g_3}^2$ and $e_{g_3}^3$ with keys k_5^1 and k_6^1. In this case, k_7^0 is obtained from decrypting $e_{g_3}^3$; the other three decryptions again yield $\perp$. Then Alice sends k_7^0 to Bob, who then reveals that key k_7^0 corresponds to output 0.

The result of garbled evaluation is 0, which means that the choices differ. This is the correct answer, since the choice of Alice was candidate 2 and the choice of Bob was candidate 1, which obviously are different choices. This concludes our example.

Yao's garbled circuit protocol is valid since the protocol always returns the correct final output due to the correctness of the existing constructions. Yao's garbled circuit protocol also fulfills the requirement for secure function evaluation. In Yao's protocol, neither party learns other party's input. However, if the function f is chosen badly, the inputs of the parties can be revealed to the other party, without violating the protocol privacy property. As an example, if Alice and Bob want to know what is their average age, then the result of f always leaks the other participant's age. In the protocol, there is no mechanism that protects against badly chosen functions. If Alice and Bob are both following the protocol, both parties always learn the final outcome of the computation. However, it is possible that Bob refuses to provide the final outcome of the computation to Alice. This causes a problem with the fairness of the protocol in the presence of a malicious party (Bob, who deviates from the protocol).

The first proof of security of Yao's garbled circuits is due to Lindell and Pinkas in static semi-honest adversarial model [107]. New techniques are required to extend Yao's protocol to malicious adversarial model. Let us first consider the vulnerable steps of Yao's protocol that make the protocol insecure in malicious adversarial model.

The steps in the garbled circuit protocol are all vulnerable to failure or deviation from the protocol: transforming the function into circuit representation, constructing the garbled circuit by generating garbled truth tables, sending garbled values to the second party, oblivious transfer for getting the input from the second party and evaluating the garbled circuit. A malicious adversary might be any of the two parties. First of all, the garbled circuit might not be constructed in an appropriate way. Then, the oblivious transfer step is vulnerable for malicious parties. Thirdly, a malicious party constructing the circuit might also try to give corrupt values to the other

party and in this way gain advantage of getting information from the other input.

In order to fight the threats above, the protocol needs improvements in the following three areas: 1-out-of-2 OT against malicious adversaries, assuring that the garbled circuit is constructed properly and preventing the party constructing the circuit from gaining advantage by sending corrupt values as her input to the other party. There are known solutions to achieve 1-out-of-2 OT against malicious adversaries [19, 127]. There are also two approaches to secure the circuit construction: zero-knowledge proofs and cut-and-choose(see the next section for more details) approach. Zero-knowledge proofs are known to be costlier than the cut-and-choose technique introduced by Lindell and Pinkas [106, 108]. Lindell and Pinkas also solve the problem of corrupted inputs in [106].

In the next section, we present in more detail various improvements of garbled circuit implementations which make garbled circuits less vulnerable against malicious adversaries. Then we discuss circuit and communication optimizations that improve the efficiency of garbled circuit protocol. After that, we present some of the known implementations of secure function evaluation using garbled circuit protocol.

3.2.1 Securing circuit construction

As discussed earlier, the construction of the garbled circuit is a major threat in garbled circuits protocol with malicious adversaries. One solution is to use cut-and-choose technique. The first, simple cut-and-choose method was used in this context in Fairplay [113]. The idea of cut-and-choose is the following. Bob creates several garbled circuits and sends them all to Alice. Alice asks Bob to open half of them. If they correspond to the function agreed by Bob and Alice, then Alice evaluates the rest of the functions with certain inputs.

By the above technique, Alice can now ascertain that the function is correct and the garbled circuits are constructed in an appropriate way. However, there is no mechanism that prevents Bob from changing his input when the different garbled circuits are evaluated. Lindell and Pinkas solve this problem by applying the cut-and-choose test both over the circuit and the inputs [106]. The latter test for inputs is realized by commitments which cause quite a big overhead if the number of circuits evaluated is large. To reduce the effect of commitments, Lindell and Pinkas published an improved version of their protocol in [108].

The above approach applies cut-and-choose method to the entire circuit. Another approach is to apply cut-and-choose to single gates in the circuit. This method was introduced by Nielsen and Orlandi in [128]. This method

was improved by Frederiksen [55]. These solutions are described in more details in section 3.2.3. The advantages of Lindell's cut-and-choose and those of Nielsen's and Frederiksen's cut-and-choose techniques were recently combined in the paper by Huang et al. [88]. The protocol in [88] utilizes multiple execution setting, where the different garbled circuits can be evaluated in parallel or sequentially.

A more recent approach for securing the circuit construction is called *cut-and-choose and forge-and-loose*. This approach combines cut-and-choose approach with the more novel *forge-and-loose* approach suggested independently by Lindell [104] and Brandão [29]. According to Brandão [29], the cut-and-choose method allows Bob to deviate from the protocol by providing inconsistent input wire keys for the garbled circuits. Alice should have a chance to abort the protocol if she notices that some of the evaluated circuits return a different value than the majority of the circuits, yielding that Bob has deviated from the protocol by changing his inputs. In other words, Alice can catch cheating Bob but cannot do anything to abort the protocol. To overcome this problem, Lindell [104] suggest an additional protocol after the cut-and-choose phase. If there are inconsistencies in the results of the evaluated circuits, Alice generates a proof of inconsistent output values. If Alice's proof is valid, then Bob must give his input to Alice. In this way, Alice can locally compute the correct value. If Alice's proof is not valid, then she does not get Bob's input. The approach of Brandão is somewhat similar. In his protocol [29] , Bob must commit to his input keys and the keys on the output wires by using trapdoor commitments. Alice can recover Bob's input by using a trapdoor if two different output keys are achieved for the same wire in two different garbled circuits.

Both solutions [104, 29] increase the computational overhead of the garbled circuit protocol. The protocol proposed by Frederiksen et al. in [54] uses the cut-and-choose, forge-and-loose approach but does not use any additional secure computation step like Lindell or contain computationally heavy trapdoor commitments like Brandão. The idea in [54] is that, instead of giving Bob's input to Alice, Bob's input is hashed with a universal hash function determined by Alice. Alice learns the output of the original function and the hash digest of Bob's input. If any of the Bob's input hash digests diverge, then Alice can abort safely without leaking her input to Bob.

3.2.2 Optimizations

The most time and memory consuming phases in Yao's garbled circuit protocol are the construction and evaluation of the garbled circuit as well as the multiple oblivious transfers during the protocol. Different methods to

decrease the computational load in the circuit construction have been discovered. There are three main optimizations: *point-and-permute*, *free-XOR* and *row-reduction*. Next, we shortly introduce each of these circuit optimizations. Then, we briefly discuss how the communication in Yao's garbled circuit protocol could be improved.

One of the first optimizations for logical circuits is point-and-permute technique, presented by Rogaway in [144]. In Yao's protocol, the circuit evaluator must decrypt four ciphertexts from which only one gives the correct key, yielding three unnecessary decryptions. Point-and-permute technique uses *select bits* to reduce unnecessary decryption steps. Each wire is assigned a randomly chosen select bit, which is independent of the actual truth value assigned to the wire. Therefore, the select bit can be revealed to the evaluator. Usually, the select bit is appended to the key (associated to the wire) as the least significant bit. In this way, the select bits can be considered as pointers to the appropriate ciphertexts, reducing the number of decryptions from four to only one.

Next we show how point-and-permute technique works for garbled circuits. The notations are adapted from [35]. Let $k_i^{b_i}$ and $k_j^{b_j}$ be the randomly chosen keys for incoming wires i and j and for bits $b_i, b_j \in \{0,1\}$. Let k_l^0 and k_l^1 denote the randomly chosen keys for outgoing wire l. We choose secret *permutation bits* for each wire i, j, l at random - these permutation bits are denoted by π_i, π_j and π_l. Each of the wires i, j and l is assigned with a select bit. For example, the select bit of wire i having value b_i is $\lambda_i = b_i \oplus \pi_i$. The garbled truth table now consists of the following ciphertexts:

$$\begin{aligned}
&\mathtt{En}_{k_i^{\pi_i}}\left(\mathtt{En}_{k_j^{\pi_j}}\left(k_l^{g(\pi_i,\pi_j)}||\pi_l \oplus g(\pi_i,\pi_j)\right)\right)\\
&\mathtt{En}_{k_i^{\pi_i}}\left(\mathtt{En}_{k_j^{1\oplus\pi_j}}\left(k_l^{g(\pi_i,1\oplus\pi_j)}||\pi_l \oplus g(\pi_i,1\oplus\pi_j)\right)\right)\\
&\mathtt{En}_{k_i^{1\oplus\pi_i}}\left(\mathtt{En}_{k_j^{\pi_j}}\left(k_l^{g(1\oplus\pi_i,\pi_j)}||\pi_l \oplus g(1\oplus\pi_i,\pi_j)\right)\right)\\
&\mathtt{En}_{k_i^{1\oplus\pi_i}}\left(\mathtt{En}_{k_j^{1\oplus\pi_j}}\left(k_l^{g(1\oplus\pi_i,1\oplus\pi_j)}||\pi_l \oplus g(1\oplus\pi_i,1\oplus\pi_j)\right)\right)
\end{aligned}$$

in this order.

To evaluate this garbled gate, the party evaluating the gate holds values $k_i^{b_i}||\lambda_i$ and $k_j^{b_j}||\lambda_j$. The keys corresponding to $k_i^{b_i}||\lambda_i$ and $k_j^{b_j}||\lambda_j$ can now be used for decrypting the ciphertext at position λ_i, λ_j of the above array. As an example, consider the case where g is an `AND` gate, $b_1 = 1$, $b_2 = 0$, $\pi_1 = 0$ and $\pi_2 = 1$. Then the select bits of incoming wires 1 and 2 are $\lambda_1 = b_1 \oplus \pi_1 = 1 \oplus 0 = 1$ and $\lambda_2 = b_2 \oplus \pi_2 = 0 \oplus 1 = 1$.

The garbled circuit is now

$$\begin{aligned}
&\mathtt{En}_{k_1^0}\left(\mathtt{En}_{k_2^1}\left(k_3^0||\pi_3\oplus 0\right)\right)\\
&\mathtt{En}_{k_1^0}\left(\mathtt{En}_{k_2^0}\left(k_3^0||\pi_3\oplus 0\right)\right)\\
&\mathtt{En}_{k_1^1}\left(\mathtt{En}_{k_2^1}\left(k_3^1||\pi_3\oplus 1\right)\right)\\
&\mathtt{En}_{k_1^1}\left(\mathtt{En}_{k_2^0}\left(k_3^0||\pi_3\oplus 0\right)\right).
\end{aligned}$$

The evaluator now wants to evaluate the circuit with $k_1^0||1$ and $k_2^1||1$. The evaluator now takes the fourth row of the garbled truth table, which corresponds to the position $\lambda_1 = 1$, $\lambda_2 = 1$ and the keys k_1^1 and k_2^0. In this way, the evaluator obtains $k_3^0||\pi_3$, which is correct since $\mathtt{AND}(1,0) = 0$ and $\lambda_3 = \mathtt{AND}(1,0) \oplus \pi_3 = \pi_3$.

The second well-known optimization for circuits is the free-XOR technique. In Yao's garbled circuit protocol, each gate is equally costly. Free-XOR technique proposed by Kolesnikov reduces the cost of XOR-gates in a simple way [99]. Let w_i and w_j be two incoming wires for an XOR-gate G, and let w_l be the outgoing wire for this gate. The wire values can be garbled in the following way. Randomly choose $k_i^0, k_j^0, R \in \{0,1\}^N$. Then set $k_l^0 = k_i^0 \oplus k_j^0$, and $\forall s \in \{i,j,l\}$ set $k_s^1 = k_s^0 \oplus R$. It can be easily verified that the garbled output is obtained by simply XORing the garbled inputs. Using this method, there is no need to randomly choose all labels for the wires in XOR gates. For XOR-rich circuits, the savings can be quite large. The security of free-XOR relies on a cryptographic hash function, which is modeled as a random oracle [99, 35]. A recent variant, *flexible-OR* introduced by Kolesnikov et al. in [98], utilizes the free-XOR approach to optimize also non-XOR gates. They claim that their flexible-XOR approach makes the garbled circuit even 30% smaller.

The third optimization technique is called *row-reduction* and it is based on point-and-permute technique. The basic idea of row-reduction technique was already proposed by Rogaway, but it was improved by Pinkas et al. in [137]. The following simple solution reduces the size of the garbled truth tables by 25%. Instead of assigning two random values to the output wire of a gate, one of the output wires is defined as a function of garbled values of the two input wires that give this output. Using other words, if a and b are two input bits to gate G, we define the garbled value of $G(a,b)$ as a function of the garbled values of a and b. The function can be chosen in such a way that the first ciphertext of a gate is a constant, and therefore it does not need to be stored (and hence the 25% reduction in the size). The solution utilizes a key derivation function, which is assumed to be correlation robust

(for definition, see [137, p. 256]). This assumption supports the use of free-XOR approach. Pinkas et al. also propose another row-reduction method, which provides a 50% reduction but does not support free-XOR technique anymore.

Also other optimizations for garbled circuits have been proposed. Kolesnikov and Kumaresan [97] optimize the communication complexity of secure two-party function evaluation by using *information-theoretic garbled circuits* and a specific oblivious transfer protocol. Information-theoretic garbled circuits are constructed by using a specific secret sharing scheme. The secrets are the output wire keys, and the constructor produces four secret shares, one for each of the wire keys of each input wire. This approach provides significant reductions especially for shallow circuits. A recent optimization of Zahur et al. [170] introduce a technique called *half-gate*, where AND-gates require only two ciphertexts instead of the traditional four. Their half-gate method provides 33% reduction in size for many circuits.

3.2.3 Implementations of garbled circuit protocol

One of the first attempts to create a practical implementation of Yao's garbled circuit protocol, *Fairplay*, was presented in [113]. Fairplay has some mechanisms against malicious parties. It introduced a simple cut-and-choose mechanism against cheating in the garbled circuit construction phase. Bob sends m garbled circuits to Alice, who randomly chooses one of them. Bob then exposes the secrets of the $m-1$ circuits not chosen by Alice. Alice then verifies that these $m-1$ circuits represent the original function f. Fairplay provides a chance of $1-\frac{1}{m}$ to detect corrupt circuits. However, Fairplay does not have a mechanism to protect against corrupt inputs.

Fairplay was tested against four functions, bitwise `AND`, "the billionaire's problem", keyed database search and the median function. The most complex circuit computed with this system contained 4383 gates, taking 320 bits as input. The execution time for this circuit was measured to be on the order of seconds. This result showed for the first time that garbled circuits are more than of theoretical interest.

The next remarkable implementation of garbled circuit protocol was *Large Efficient Garbled-circuit Optimization*(LEGO) presented by Nielsen and Orlandi [128]. LEGO improves the tolerance against active adversaries. The basic idea is that both parties participate in the construction of the garbled circuit. Bob prepares and sends a set of garbled `NAND` gates to Alice. Alice checks a fraction of them in order to guarantee an overwhelming probability that there are very few bad gates among the non-checked gates. Alice permutes the `NAND` gates and appends them to garbled circuit using a fault-tolerant circuit design. This assures that Alice is still able to compute the desired function even though there might be a few random faulty gates

in the circuit. After doing this, Alice evaluates the circuit as in the original garbled circuit protocol.

The difference between Fairplay and LEGO is that in the Fairplay protocol, the cut-and-choose method was applied to the entire circuit. In LEGO protocol, cut-and-choose was applied at the gate level, speeding up the protocol. However, the LEGO protocol was considered too complex to implement and use. In addition, it was not compatible with the known optimizations (free-XOR, row reduction, point-and-permute) for Yao's garbled circuit. These shortcomings were removed in miniLEGO protocol introduced by Frederiksen et al. in [55].

Bellare et al. presented another implementation of Yao's garbled circuits in [16]. The system JustGarble (see [94]) implements three garbling algorithms, all of which use different circuit optimization techniques. The first of the garbling algorithms uses the point-and-permute technique. The second garbling algorithm augments the first algorithm by free-XOR technique. The third garbling algorithm augments the second algorithm by row reduction technique.

Bellare et al. also presented results of a timing test for their implemented JustGarble system. In JustGarble system a circuit with 15.5 million gates is garbled and evaluated in less than a second. As a comparison to JustGarble, it took approximately 10 seconds to garble and evaluate a 4400-gate circuit in Fairplay system. The speed of JustGarble library bases on the use of AES-NI, which is a set of CPU instructions for encryption/decryption and for the AES key expansion.

A recent improved implementation of garbled circuit protocol is TinyGarble [152], presented by Songhori et al. TinyGarble uses sequential circuit description of garbled circuits, which improves the circuit compactness. The improved circuit-compactness has a major advantage: the memory footprint of the garbling operation fits in the processor cache, which reduces the number of CPU cycles. Songhori et al. also show that TinyGarble allows new garbled objects to be implemented. They implement a new standard hash function called SHA-3 which was introduced by Bertoni et al. in [23] and standardized by NIST in 2015 [132]. In addition, they implement a garbled processor based on TinyGarble.

All these implementations rely on rather strong cryptographic assumptions. As an example, AES is assumed to be secure for related keys. The justification for making strong assumptions is that assuming them, fast garbling can be achieved. Recently, Geuron et al. have posed the question whether such strong assumptions for fast garbling are really necessary. Gueron et al. provide new methods for garbling whose security assume only pseudorandom functions [81].

3.3 Garbling schemes

The previous section shows that garbled circuits have been of great interest in recent research. Both theoretical and practical results show that garbled circuits are a powerful cryptographic tool which can be used in several contexts for secure function evaluation. However, garbled circuits also have their disadvantages. One of these is the presentation of the function as a logical circuit. Complex functions should be transformed into circuits before garbling. This can be time and space consuming. Partly due to this issue, garbling technique has been applied also to other computation models such as Turing machines and random-access machines. Garbled Turing machines were proposed by Goldwasser et al. in [74]. Their construction assumes existence of fully-homomorphic encryption schemes. Garbled random-access machines have been constructed in several papers [110, 62, 2].

Even though circuits, TMs and RAMs are very different as computation models, the process of garbling them consists of similar steps. First, a garbled representation of the function should be constructed. Then, the argument to the function should be garbled. Then follows the garbled evaluation, providing the garbled outcome of the computation. The garbled computation result should then be ungarbled to get the actual result of the computation. All these steps can be modeled as separate algorithms, which together form a set of algorithms, called *a garbling scheme*.

More formally, a garbling scheme is a 5-tuple of algorithms, $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{Ev}, \mathtt{De}, \mathtt{ev})$. The last component in this tuple, $\mathtt{ev}$, is the original evaluation algorithm that computes $y = f(x)$ when f and x are given as inputs. The garbling algorithm $\mathtt{Gb}$ is used to compute the garbled function $F = \mathtt{Gb}(1^k, f)$ based on the security parameter k. The second component is an encryption algorithm which is used to compute the garbled argument $X = \mathtt{En}(e, x)$. The third component in the tuple is the garbled evaluation algorithm $\mathtt{Ev}$ which in turn computes the garbled evaluation result $Y = \mathtt{Ev}(F, X)$. The fourth component $\mathtt{De}$ is a decryption function which returns the final evaluation result $y = \mathtt{De}(d, Y) = \mathtt{ev}(f, x)$. Figure 3.8 illustrates how a garbling scheme works.

The definition of a garbling scheme does not set any assumptions regarding the function or the argument. However, sometimes it is required that the argument is fed bit by bit to the function. As an example of such application, consider one-time programs [75]. The bit-by-bit property sets additional requirements for the security definitions of garbling schemes. This requirement is captured by the concept of *projectivity*. Below is the formal definition of what is meant by a projective garbling scheme.

Definition 3.3.6 *Let $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev})$ be a garbling scheme which is used to evaluate function f. Let $x = x_1 \dots x_n \in \{0,1\}^n$ and $x' = x'_1 \dots x'_n \in$*

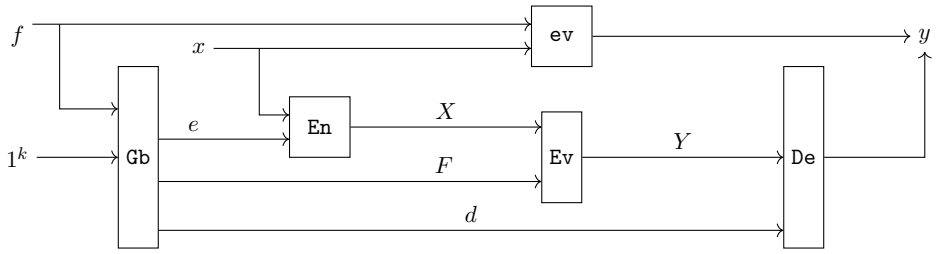


Figure 3.8: Idea behind garbling. The diagram shows that the final value obtained via garbling must coincide with the final value obtained by direct evaluation, i.e. $\texttt{ev}(f, x) = y = \texttt{De}(d, Y)$ where F is the garbled function, $Y = \texttt{Ev}(F, X)$ is the garbled value and $X = \texttt{En}(e, x))$ is the garbled argument.

$\{0,1\}^n$ *be any two arguments for* f*. Let* $\texttt{En}$ *be a non-deterministic algorithm and let* r *denote the explicit randomness value used in computation of* $\texttt{En}$*. We say that garbling scheme* $\mathcal{G}$ *is* projective *if the two garbled arguments* $X = \texttt{En}(e, x, r)$ *and* $X' = \texttt{En}(e, x', r)$ *can be presented in form* $X = (X_1, \ldots, X_n)$*,* $X' = (X'_1, \ldots, X'_n)$ *in such way that the following condition holds:* $x_i = x'_i$ *iff* $X_i = X'_i$*.*

Next we give an example of a garbling scheme which is projective in the sense stated in above definition.

Example 2 The projectivity is a property related to the garbling of the argument, i.e. the encryption algorithm $\texttt{En}$ of the garbling scheme. In this example, we provide an encryption algorithm that has the projectivity property. For constructing the encryption algorithm $\texttt{En}$, we use 128-bit AES. Let e be an encryption key generated by the garbling algorithm for encrypting arguments with $\texttt{En}$. The randomness value r is constructed as follows. It consists of n bit strings, all of length 127, i.e. $r = r_1 \ldots r_n$ where n is the length of the argument x and $r_i \in \{0,1\}^{127}$ for all $i \in \{1, \ldots, n\}$. All the 127-bit values r_i are chosen at random.

The encryption algorithm $\texttt{En}$ takes e, x and r as input. The resulting garbled argument of x is constructed as follows:

$$\texttt{En}(e, x, r) = \texttt{En}_{\texttt{AES}}(x_1||r_1, e) \ldots \texttt{En}_{\texttt{AES}}(x_n||r_n, e)$$

Let us show why this encryption algorithm now guarantees the projectivity of the garbling scheme. Let $x = x_1 \ldots x_n$ and $x' = x'_1 \ldots x'_n$ be two arguments of length n. Let X and X' be the garbled arguments corresponding to these two arguments x and x'. Both arguments are encrypted with the same key e and by using the same randomness value r. Now, the garbled arguments $X = \texttt{En}(e, x, r)$ and $X' = \texttt{En}(e, x', r)$ can both be represented in form $X = X_1 \ldots X_n$ and $X' = X'_1 \ldots X'_n$: $X_i = \texttt{En}_{\texttt{AES}}(x_i||r_i, e)$ and $X'_i = \texttt{En}_{\texttt{AES}}(x'_i||r_i, e)$. If $x_i = x'_i$, then the input to the $\texttt{AES}$ algorithm is the

same for both x_i and x'_i, so the garbled bits X_i and X'_i are also identical because $X_i = \texttt{AES}(x_i||r_i, e) = \texttt{AES}(x'_i||r_i, e) = X'_i$. Conversely, if $X_i = X'_i$, then we have that $x_i = x'_i$. This is shown as follows. When we decrypt X_i with AES we get $x_i||r_i$ and when we decrypt X'_i with AES we get $x_i||r_i$. These decryptions must be equal, because the encryption was performed using the same key with the AES algorithm. Now, since $x_i||r_i = x'_i||r_i$ we must have that $x_i = x'_i$. This now completes the example.

3.4 Security of garbling schemes

This section provides security definitions of garbling schemes. There are three concepts that characterize the security of a garbling scheme: *security notion*, *security model* and *level of adaptivity*. Next we briefly describe each of these three concepts. The descriptions are informal and intuitive. For more formal treatment of security notions, see [18, 17] and Publications I–V.

Security notions: The security notion indicates how much information about the final result y is allowed to be leaked to the evaluator. There are three security notions, privacy (prv), obliviousness (obv) and matchability-only (mao). In the privacy notion, the evaluator is allowed to decrypt the garbled evaluation result and hence learn the final evaluation result y entirely. In the obliviousness notion, the evaluator is not allowed to decrypt the garbled evaluation result and therefore is not allowed to learn y. In the matchability-only notion, the evaluator is not allowed to learn y but is allowed to learn whether evaluating f_0 on x_0 gives the same final evaluation result as evaluating f_1 on x_1, i.e. whether $\texttt{ev}(f_0, x_0) = \texttt{ev}(f_1, x_1)$.

In [18], a fourth notion is also defined - a notion for *authenticity*. The authenticity notion gathers the idea that the evaluator cannot produce a forged garbled evaluation result that would be decrypted into a valid final evaluation result. In this thesis, authenticity notion does not play as important role as the three other notions. The reason is that the authenticity notion has been proven to be very different from the three other notions. This separation between authenticity notion and other security notions is discussed in section 3.4.3 in more details. However, authenticity notion is important from practical point-of-view. For example, the application proposed in Publication VII requires a garbling scheme that achieves authenticity.

Security models: In the simulation-based security model, the task of the adversary is to distinguish whether the garbled function F and the garbled argument X are computed by real garbling algorithms $\texttt{Gb}, \texttt{En}$ or by a probabilistic polynomial-time simulator $\mathcal{S}$ that does *not* have the same information as algorithms En and Gb. In the indistinguishability-based model, the task

of the adversary is to distinguish which one of the argument/function pairs, (f_0, x_0) or (f_1, x_1), has been garbled to (F, X).

Type of adaptivity: Informally, the type of adaptivity tells how a function f and an argument x can be garbled in different situations: in some scenarios, it is enough that the function and the argument are both garbled at one go but in some scenarios it is needed that the function is garbled before the argument is even fixed. In other words, level of adaptivity tells in which order and how the function and the argument can be garbled. In the static model, an adversary fixes both f and x which are garbled at one go. In the adaptive model, the adversary may first fix function f and, based on the garbled function F, fix his argument(s). In adaptive notions, there is an additional parameter ℓ, the reusability parameter. This parameter tells how many times the same garbled function can be used securely for different arguments. Adaptive security for projective garbling schemes is just a special case of adaptive security. Projective schemes allow the argument to be determined bit by bit whereas in the case of general adaptivity all bits of the argument are determined at once. Finally, reverse-order adaptive security changes the roles of function f and argument x: in static and adaptive security models, the function is fixed and garbled only once whereas the argument may vary and there are several different garblings for the argument. In reverse-order garbling, the argument is fixed and garbled only once whereas the function may vary and there are several different garblings for the function. This model is introduced for practical reasons. For example, in statistical analysis the data does not change whereas the algorithms used for the analysis may change. If we used a garbling scheme achieving static or adaptive security then we should garble the algorithm and the data again every time the algorithm is changed. In reverse-order model, we get rid of garbling the data unnecessarily many times.

Next, we briefly describe the security games and skip the exact definitions. The exact definitions for the security games are provided in the following publications. The definitions of the classes of statically secure garbling schemes can be found in [18] and in publication I. The classes of adaptively secure garbling schemes are found in [17] and in publication II. Adaptively secure projective garbling schemes are covered in publications III and IV and reverse-order adaptively secure garbling schemes are studied in publication V.

All security games start with procedure `INITIALIZE`, in which the challenge bit is chosen uniformly at random. All the games end in procedure `FINALIZE`, in which adversary's answer (consisting of one bit) is checked against the challenge bit (which is the correct answer in the game). In between the adversary may query other named procedures. In static security games, there is only one additional procedure `GARBLE`. In simulation-based

model, the challenge bit is used for determining whether the actual garbling algorithm or the simulator is used for creating the garbled function F and the garbled argument X . In indistinguishability model, the challenge bit determines which one of the two function-argument pairs, (f_0, x_0) or (f_1, x_1), is garbled. In various adaptive security games, there are separate procedures for garbling the argument (`GARBLE_ARG`) and for garbling the function (`GARBLE_FUNC`).

3.4.1 Side-information in garbling

All security games are parameterized by two concepts. First one is the security parameter $k \in \mathbb{N}$, which is a part of the input to the garbling algorithm `Gb`. Another parameter points to the crucial concept of a side-information, denoted by Φ. Informally speaking, the concept of side-information captures the information that is allowed to be leaked during the garbled evaluation. More formally, side-information function is a mapping from bit strings to bit strings. In the case of logical circuits, side-information function maps function f deterministically into $\Phi(f)$. Side-information function for circuits always leaks at least the length of input x of circuit f, the length of the output y of circuit f and the size of circuit f. Using other words, $|x|$, $|y|$ and $|f|$ are efficiently computable from side-information $\Phi(f)$.

The concept of side-information provides another point-of-view to the leaked information, compared with the security notions privacy, obliviousness and matchability-only. The security notions tell which level of privacy a garbling scheme provides for f, x and y. For example, a garbling scheme achieving privacy guarantees the privacy of f and x but not the privacy of the final result y whereas a garbling scheme achieving obliviousness provides privacy to f, x and y. The concept of side-information in turn tells how much it is acceptable to learn about f, x and y by following the garbled evaluation and by analyzing the garbled function F and the garbled argument X.

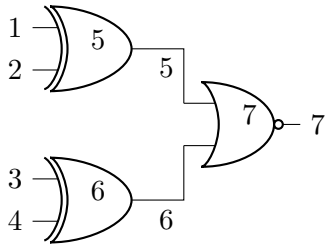
There are three common, intuitive side-information functions for circuits: $\Phi_{\text{size}}(f)$, $\Phi_{\text{topo}}(f)$ and $\Phi_{\text{circ}}(f)$. Φ_{size} leaks only size-related information, i.e. $|x|$, $|y|$ and the number of gates in f. Φ_{topo} leaks in addition the topology of the circuit, i.e. how different gates are connected to each other but not the type of the gates. Φ_{circ} leaks the entire circuit.

In the next example we demonstrate what the three different side-information functions are for the circuit which we used in example 3.4. We adapt the notation of circuits from [18]. According to [18], a circuit f is a 6-tuple $f = (m, n, q, A, B, G)$. The first element m tells the number of input wires. The second element n denotes the number of output wires. The third element q denotes the number of gates in the circuit f. The set of inputs is $\texttt{Inputs} = \{1, \ldots, m\}$. The set of all

wires in the circuit is $\texttt{Wires} = \{1, \ldots, m+q\}$. The set of output wires is $\texttt{Outputs} = \{m+q-n+1, \ldots, m+q\}$. Finally, the set of gates is $\texttt{Gates} = \{m+1, \ldots, m+q\}$. The labeling of the gates can be justified by the following observations: There is a unique outgoing wire from each gate. Every outgoing wire comes from a unique gate. Therefore, the labeling of outgoing wires is unique and hence the labeling of the gates is unique as well. Note that the same does not hold for incoming wires: a wire might be an ingoing wire to several different gates.

Using these sets, we can define three components A, B and G in the 6-tuple (m, n, q, A, B, G). The element A is a function that identifies the first incoming wire of a gate, i.e. $A : \texttt{Gates} \to \texttt{Wires}\backslash\texttt{Outputs}$. The element B is a function that identifies the second incoming wire of a gate, i.e. $B : \texttt{Gates} \to \texttt{Wires}\backslash\texttt{Outputs}$. Finally, the element G is a function that tells the functionality of each gate, i.e. $G : \texttt{Gates} \times \{0,1\}^2 \to \{0,1\}$.

Example 3 Let us consider the following circuit C:



The circuit takes four bits as input and outputs one bit of information, so $m = 4$ and $n = 1$. There are three gates, i.e. $q = 3$. The sets `Inputs`, `Wires`, `Outputs` and `Gates` are as follows:

$$\begin{aligned}
\texttt{Inputs} &= \{1, 2, 3, 4\} \\
\texttt{Wires} &= \{1, 2, 3, 4, 5, 6, 7\} \\
\texttt{Outputs} &= \{7\} \\
\texttt{Gates} &= \{5, 6, 7\}\,.
\end{aligned}$$

The functions A_C, B_C and G_C for circuit C are defined in fig. 3.9.

Now, the circuit presented above is represented as a 6-tuple $f_C = (4, 1, 3, A_C, B_C, G_C)$.

First consider the side-information function $\Phi_{\text{size}}(f_C)$. This side-information function is defined as $\Phi_{\text{size}}(f) = (m, n, q)$ which in this case is $\Phi_{\text{size}}(f_C) = (4, 1, 3)$. In other words, the side-information $\Phi_{\text{size}}(f_C)$ leaks that the circuit f has 4 input wires, 1 output wire and 3 gates.

Then consider the side-information function $\Phi_{\text{topo}}(f)$. This side-information function reveals the topology of the circuit, in addition to the size-related information. By definition in [18], the side-information $\Phi_{\text{topo}}(f)$

$$
\begin{array}{ll}
A_C(5) = 1 & B_C(5) = 2 \\
A_C(6) = 3 & B_C(6) = 4 \\
A_C(7) = 5 & B_C(7) = 6
\end{array}
\qquad
\begin{array}{lll}
G_C(5,(0,0)) = 0 & G_C(6,(0,0)) = 0 & G_C(7,(0,0)) = 1 \\
G_C(5,(0,1)) = 1 & G_C(6,(0,1)) = 1 & G_C(7,(0,1)) = 0 \\
G_C(5,(1,0)) = 1 & G_C(6,(1,0)) = 1 & G_C(7,(1,0)) = 0 \\
G_C(5,(1,1)) = 0 & G_C(6,(1,1)) = 0 & G_C(7,(1,1)) = 0
\end{array}
$$

Figure 3.9: Functions A_C, B_C and G_C for circuit C.

leaks (m, n, q, A, B) which in this case is $\Phi_{\text{topo}}(f_C) = (4, 1, 3, A_C, B_C)$. Functions A_C and B_C reveal how the wires are connected to the gates. However, these two functions A and B do not leak anything about the functionality of the gates. Figure fig. 3.10 illustrates the information leaked by the side-information function $\Phi_{\text{topo}}(f_C)$.

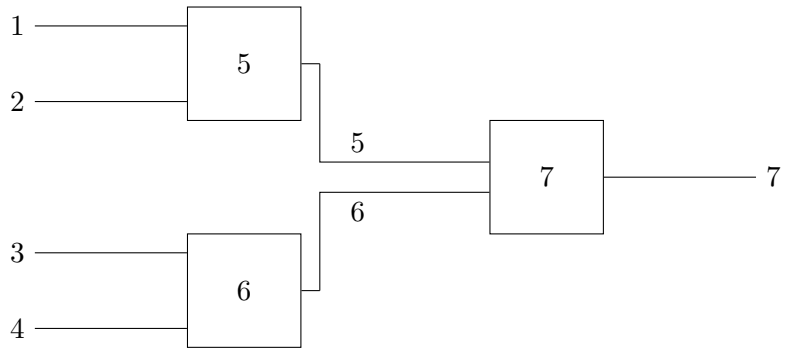


Figure 3.10: Side-information function Φ_{topo} leaks the topology of circuit from example 3.4. The functionality of the three gates is not leaked.

The third side-information function, $\Phi_{\text{circ}}(f)$, leaks the functionality of each gate in addition to the size-related information and the topology of f. In other words, side-information function Φ_{circ} reveals the entire circuit, i.e. $\Phi_{\text{circ}}(f_C) = f_C = (m, n, q, A_C, B_C, G_C)$.

Garbling schemes are not only applicable for logical circuits. In publication VI we point out that the currently used model of side-information depending only on function f is not appropriate for Turing machines. Therefore, the model of side-information is extended so that it depends on the function f, the argument x as well as the encryption and decryption keys e, d. This extension is covered in more details in section 4.5.

3.4.2 Formal security definitions

Next we provide formal definitions of security of a garbling scheme. We use the following conventions in the definitions. We use notation XxxYyyZzz for the code-based security games. Xxx denotes the security notion: Prv corresponds to privacy notion, Obv corresponds to obliviousness notion and Mao corresponds to matchability-only notion. Xxx could also be used for denoting the class of garbling schemes achieving authenticity (Aut). However, in this work we do not consider authenticity notion, so the notion Aut is omitted. Letters Yyy correspond to the security model: Sim denotes the simulation-based security game, Ind denotes the indistinguishability-based game. The logic in simulation-based and indistinguishability-based security classes is fundamentally different. In simulation-based model, security games also depend on the choice of an additional procedure, a simulator $\mathcal{S}$. Intuitively, the task of a simulator is to mimic the actual garbling algorithms `Gb` and `En`. The letters Zzz give the level of adaptivity: Stat corresponds to the static security game, Adap_ℓ corresponds to adaptive security games with ℓ times reusability of the same garbled function, Padap_ℓ corresponds to the adaptive security games for projective garbling schemes having the reusability level ℓ, Radap_ℓ corresponds to reverse-order adaptive security games with reusability level ℓ. A garbling scheme is said to be *xxx.yyy.zzz* secure over a side-information function Φ if an arbitrary adversary has a negligible advantage with respect to the security parameter in the corresponding security game XxxYyyZzz.

We first define *advantage of an adversary* which conceptualizes the adversary's ability to win the code-based game XxxYyyZzz played with certain side-information function against a garbling scheme. The ability to win the game XxxYyyZzz in turn tells how good the chance is to break the xxx.yyy.zzz type of security in the garbling scheme. There is a separate definition for all the games depending on the choice of the Xxx, Yyy and Zzz.

Definition 3.4.7 *Let adversary $\mathcal{A}$ be playing code-based game XxxYyyZzz against garbling scheme $\mathcal{G}$ where Xxx $\in$ {Prv, Obv, Mao}, Yyy $\in$ {Sim, Ind} and Zzz $\in$ {Stat, Adap_ℓ, Padap_ℓ, Radap_ℓ}. The advantage of adversary $\mathcal{A}$ in game XxxYyyZzz is defined over the security parameter k and side-information function Φ as follows*

$$\boldsymbol{Adv}_{\mathcal{G}}^{xxx.yyy.zzz,\Phi}(\mathcal{A},k) = 2 \cdot \Pr\left[\mathcal{A} \text{ wins}\right] - 1.$$

The probability $\Pr[\mathcal{A} \text{ wins}]$ *refers to the probability that adversary $\mathcal{A}$ correctly predicts the challenge bit. Note that in the simulation-based security games the advantage also depends on the simulator $\mathcal{S}$. In this case we use notation* $\boldsymbol{Adv}_{\mathcal{G}}^{xxx.yyy.zzz,\Phi,\mathcal{S}}(\mathcal{A},k)$.

Definition 3.4.8 *Let Xxx* $\in$ {Prv, Obv, Mao} *and Zzz* $\in$ {Stat, Adap_ℓ, Padap_ℓ, Radap_ℓ}. *A garbling scheme* $\mathcal{G}$ *is XxxIndZzz secure over* Φ *if for any* $\mathcal{PT}$ *adversary* $\mathcal{A}$ *the advantage of the adversary* $\mathcal{A}$ *in game XxxSimZzz is negligible over* k.

Definition 3.4.9 *Let Xxx* $\in$ {Prv, Obv, Mao} *and Zzz* $\in$ {Stat, Adap_ℓ, Padap_ℓ, Radap_ℓ}. *A garbling scheme* $\mathcal{G}$ *is XxxSimZzz secure over* Φ *if for any* $\mathcal{PT}$ *adversary* $\mathcal{A}$ *there is a* $\mathcal{PT}$ *simulator* $\mathcal{S}$ *such that the advantage of the adversary* $\mathcal{A}$ *in game XxxSimZzz is negligible over* k.

All garbling schemes achieving certain type of security are said to belong to the same *security class*. The security class of all xxx.yyy.zzz secure garbling schemes over side-information Φ is denoted by $\texttt{GS}(\text{xxx.yyy.zzz}, \Phi)$. Here $xxx \in \{\text{prv, obv, mao}\}$, $yyy \in \{\text{sim, ind}\}$ and $zzz \in \{\text{stat}, \text{adap}_\ell, \text{padap}_\ell, \text{radap}_\ell\}$. In other words, the xxx-part of the notation refers to the security notation (privacy, obliviousness, matchability-only), the yyy-part refers to the security model (simulation-based, indisinguishability-based) and the zzz-part refers to the level of adaptivity.

In the following section we present some of established relations between the security classes which were introduced by Bellare et al. in [18]. The results in publications I-V extend the relations and propose a hierarchical presentation for the relations between the security classes. The extensions and the hierarchy are discussed in more detail in the next chapter.

3.4.3 Relations between the security classes

Bellare et al. have proven the relations shown in fig. 3.11 for static security classes. The diagram shows that simulation-based security implies indistinguishability-based security. In other words, any garbling scheme which achieves simulation-based privacy (or obliviousness, respectively) achieves also indistinguishability-based privacy (or obliviousness, respectively). The converse statement does not hold, unless the side-information function has some additional properties. In fig. 3.11, the inclusion of simulation-based security classes into indistinguishability-based security classes is shown using black, solid arrows. All known non-inclusions are shown as red lines having a slash.

Indistinguishability-based security implies simulation-based security only for *efficiently invertible* side-information functions. We say that side-information function Φ is efficiently invertible, if there is an efficient algorithm which takes side-information $\Phi(f)$ as input and outputs a function f' such that the side-information is the same for both functions, i.e.

$\Phi(f') = \Phi(f)$. The three side-information functions Φ_{size}, Φ_{topo} and Φ_{circ} from section 3.4.1 are efficiently invertible.

We can also consider efficient invertibility for the pair $(\Phi, \mathtt{ev})$. We say that $(\Phi, \mathtt{ev})$ is efficiently invertible, if there is an efficient algorithm which takes $\Phi(f)$ and $y = \mathtt{ev}(f, x)$ as input and outputs (f', x') such that $\Phi(f') = \Phi(f)$ and $\mathtt{ev}(f', x') = y = \mathtt{ev}(f, x)$. In [18] it is shown that $(\Phi_{\text{topo}}, \mathtt{ev}_{\text{circ}})$ and $(\Phi_{\text{size}}, \mathtt{ev}_{\text{circ}})$ are efficiently invertible, whereas $(\Phi_{\text{circ}}, \mathtt{ev}_{\text{circ}})$ is not. Here, $\mathtt{ev}_{\text{circ}}$ denotes the usual evaluation algorithm for circuits. For justifying why $(\Phi_{\text{circ}}, \mathtt{ev}_{\text{circ}})$ is not efficiently invertible, consider a function f which is a one-way function.

There are also other non-inclusions shown in fig. 3.11. These non-inclusions illustrate the fact that authenticity as a security notion is different from the security notions of privacy and obliviousness. On one hand, it holds for all side-information functions that neither simulation-based privacy nor simulation-based obliviousness implies authenticity. On the other hand, authenticity implies neither indistinguishability-based privacy nor indistinguishability-based obliviousness. These two results show that the separation between authenticity and the two other notions is quite strong.

Bellare et al. show in [17] that the same inclusions hold for adaptively secure garbling schemes. Bellare et al. use two different definitions for adaptivity: *coarse-grained* and *fine-grained*. In coarse-grained adaptivity, the function is fixed and garbled before the argument. Fine-grained adaptivity has this same property but now the argument can be fixed and garbled bit by bit instead of garbling it at one go. In our terminology, garbling schemes achieving coarse-grained adaptivity are called adaptively secure. We say that a projective garbling scheme is adaptively secure if it achieves fine-grained adaptivity. The results in [17] show that a garbling scheme that achieves coarse-grained adaptivity for some security notion in simulation-based model then the garbling scheme achieves coarse-grained adaptivity for the same security notion in indistinguishability-based model. The same holds for fine-grained adaptivity. Furthermore, Bellare et al. show that fine-grained security implies coarse-grained security.

The separation between obliviousness/privacy and authenticity is not only theoretical. Recently, Frederiksen et al. have been able to design garbling schemes for circuits that achieve authenticity but not privacy or obliviousness. This shows that the separation between authenticity notion and the two other notions appears also in practice [56].

In publications I-V, we introduce several extensions concerning security notions and their relations. In the next chapter we will discuss all these extensions in more details.

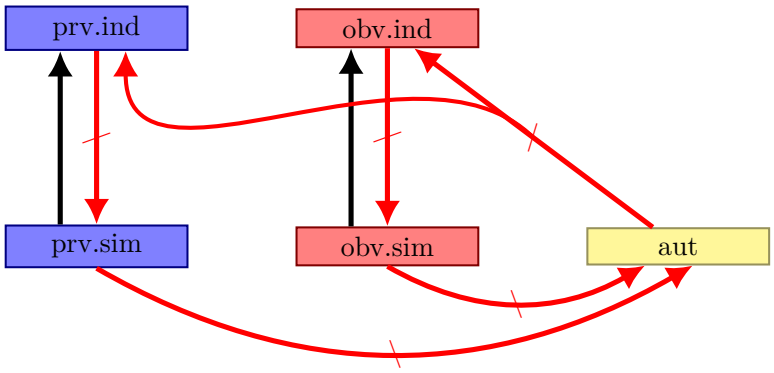


Figure 3.11: Relations between the classes of garbling schemes. The figure is adapted from [18, p.786].

3.5 Applications of garbling schemes

Garbled circuits have been widely used in various applications. They have been applied e.g. for constructing zero-knowledge proofs [25, 95] to achieve oblivious outsourcing as well as to achieve verifiable computation [60]. In addition to applications on theory, garbled circuits, or garbling schemes more generally, have also several practical applications. In this section, we briefly introduce different applications in which garbling could be used for privacy-enhancing and privacy-preserving applications.

Cloud computing and distributed computing are emerging techniques in personal and corporate use. Privacy sensitive personal data and corporate data can be utilized in various ways. As an example, an Internet user may be profiled based on his browsing behavior and the profile information may be used for targeted advertising or a corporate may perform industrial espionage in order to acquire intellectual property from a rival company. To protect against misuse of data, both cloud computing and distributed computing applications require techniques for privacy preservation which protect the privacy-sensitive data.

Another emerging paradigm is the Internet of Things (IoT), a network of low-resource devices that share a common task, e.g. surveillance or monitoring. IoT devices may generate large amounts of data, requiring a large-capacity storage and heavy computations to analyze the data. Therefore, the data is first transferred to a destination having greater resources for computation and storage. An emerging trend is to store and analyze data in cloud environment. However, cloud environments might not be secure enough in order to store privacy-sensitive data collected by the IoT sensors. To tackle this issue, privacy-preserving techniques are needed to protect the privacy of the data.

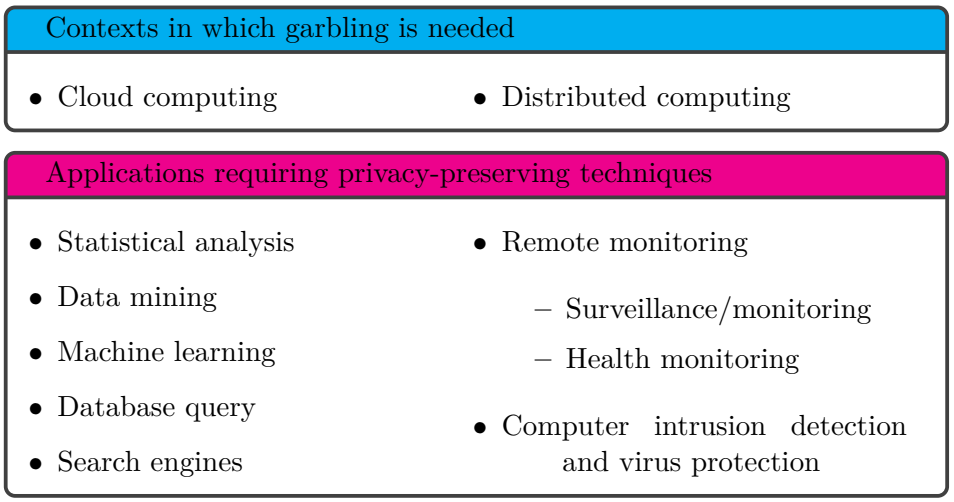


Figure 3.12: Potential applications of garbling schemes

The fig. 3.12 presents the common contexts where garbling can be applied and applications requiring methods for privacy preservation. As mentioned above, privacy-sensitive data can be used both in cloud computing as well as in distributed computing. The applications that may take privacy-sensitive information as input are also shown in fig. 3.12. Statistical analysis, data mining and machine learning are used for finding and utilizing patterns in data. Database queries and search engines are used for finding specific information among entries in a database. Remote monitoring in turn can be applied in various contexts, including security surveillance services and health services.

In the following sections, we discuss the contexts and applications for garbling schemes in more details. We begin with cloud computing and distributed computing. Then we discuss the Internet of Things paradigm. In the last two sections we consider how garbling could be used for eHealth and for assisted living services. eHealth is a healthcare practice supported by electronic processes and communication whereas the aim of assisted living services is to supervise or assist disabled people with activities of daily living.

3.5.1 Distributed and cloud computing

Modern computation aims to be *ubiquitous* - computing is made to appear anytime and everywhere on any device, not only on specific computers. Distributed computing, cloud computing and the Internet of Things are some of the research fields which touch ubiquitous computing. Privacy is one of the biggest obstacles to the success of ubiquitous computing [86].

Multiparty computation protocols can be considered as a solution. Multiparty computation protocols are designed to be used in a situation where a group of parties jointly want to compute a functionality. Therefore, multi-party computation suits well in *distributed computation* scenario where a heavy computational task is distributed between several parties controlled by one central server. This approach has been used in projects like SETI@Home [6], Folding@Home [136] and Mersenne Prime Search [89].

Data mining is an example where distributed computing may be utilized. There are several ways to present the privacy-preserving data mining problem [48]. Lindell and Pinkas [105] consider a scenario in which two parties want jointly perform data mining actions on the union of their two databases without exposing their databases to the other party or any third party. Agrawal and Srikant [3] consider a scenario in which one party is allowed to perform data mining actions on a private database owned by another party without having access to the database. Lindell and Pinkas use secure multi-party computation approach to solve the problem whereas Agrawal and Srikant use data perturbation methods (which are not in the scope of this work).

Another potential application area for secure multi-party computation protocols like garbling schemes is cloud computing. According to NIST [115], "cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources". Various cloud services are thought to fall in some of three known categories: *Infrastructure as a Service (IaaS)*, *Software as a Service (SaaS)* and *Platform as a Service (PaaS)*. In brief, IaaS provides virtual machines or storage from a provider on demand being able to autonomously adapt the capacity to varying workload over time. Google Compute Engine, Microsoft Azure and Amazon EC2/S3 are examples of IaaS. SaaS provides applications that can be run on cloud via the Internet. Google and Microsoft offer office software (e.g. Google Docs and Office 365) that can be used via web browser. PaaS in turn is used for applications while providing cloud components to software. It allows customers to simultaneously develop, run and manage applications at the same cloud-based environment. Examples of PaaS are e.g. Google App Engine, Microsoft Azure and Amazon Web Services Elastic Beanstalk.

Cloud services are increasingly popular even though there still are issues with their security and especially privacy [92, 154]. Depending on the service, different cryptographic tools are needed to guarantee the various security aspects related to cloud services. For data storage, important aspects of security include e.g. proofs of retrievability and provable data possession [140, Chapter 5]. In this work, we are interested in processing data in a potentially insecure environment like clouds, so we do not handle these concepts in more details.

Processing data on cloud in privacy-preserving way requires that the cloud does not learn the private-sensitive data which it processes. In some cases, there is no need to hide the algorithm used for the computation. A solution for this kind of scenario is to perform computations on encrypted data. To achieve this, homomorphic encryption can be used as a solution. In some cases, also the algorithm needs to be kept private, especially if it contains components that are considered to be intellectual property. In this scenario, garbling technique can be applied, since garbling schemes are designed to hide both the input and the algorithm.

3.5.2 Internet of Things

The Internet of Things is a paradigm that aims at creating an environment of networked devices communicating over the Internet and serving to accomplish a joint task. As an example of an IoT application, consider anti-theft system with motion detecting sensors. The sensors at different locations interact with each other in order to detect unauthorized motion and prevent intruders. Many other applications of IoT can be found in [7, 159].

There are several threats that may prevent the success of IoT based applications. The threats are mainly targeted at infrastructure, protocol and network security, data security and privacy, identity management, trust and governance as well as at fault tolerance [145]. Also tamper-resistance and other physical security aspects are important when considering the possible threats against IoT applications [159].

Trust and privacy are important aspects when considering security of IoT applications. According to Weber [161], the IoT technology used by private enterprises must have resilience to attacks, they must authenticate the retrieved address and object information, they must have an access control and ensure client privacy. However, there are numerous scenarios which endanger the security, trust or privacy of the IoT applications [100]. As a demonstrative example, a failed implementation of IoT related technology in a supermarket may violate the client privacy by enabling "the mining of medical data, invasive targeted advertising, and loss of autonomy through marketing profiles or personal affect monitoring" [163].

On the contrary, there are also successful implementations of privacy-preserving applications utilizing IoT paradigm. Many of these implementations are related to electronic surveillance [57, 134] or remote monitoring as an eHealth application [1]. In this work, we also consider applications for privacy-preserving electronic surveillance (publication VII) and remote monitoring for assisted living services (publication VIII).

3.5.3 eHealth

eHealth encompasses a wide variety of services, including electronic health records, clinical decision support, healthcare information systems, mHealth and telemedicine. Services which handle and transfer health related information require privacy-preservation techniques since health related data are considered highly privacy-sensitive. Therefore, eHealth applications must be carefully designed and implemented. Appropriate implementation guarantees that patients' privacy is not easily compromised.

Securing eHealth services requires focusing on both server and client platforms. Many of the current eHealth solutions focus on network security or access control policies, leaving the client platform vulnerable to attacks [109]. Despite the various threats against eHealth services, network-based solutions seem to give great benefits not only for eHealth service providers but also for patients [157].

One of the central tasks in eHealth is to develop user-friendly and efficient privacy-preserving applications. Several such applications have already been proposed and implemented. As an example, Layouni et al. introduce a protocol that allows telemonitoring for eHealth but only at patients' approval [102]. A system for remote ECG analysis has been proposed by Barni et al. in [13]. Automated emergency healthcare process utilizing cloud services has also been studied recently [93, 138].

All the systems mentioned above are designed for health institutions collecting health data from patients. Also systems intended for the use by patients and customers have been proposed. This kind of services are often provided in mobile environment, where patients and customers can access health services and information using their mobile phones, tablet computers and other mobile devices. These types of health services supported by mobile devices are known as mHealth services. Different cloud-assisted mHealth services are presented e.g. in [130, 47, 85]. Achieving privacy-preserving data storage, privacy-preserving data retrieval and auditability against misusing health data are among the main challenges for mHealth applications. A system achieving these three properties has recently been developed by Tong et al. in [156].

3.5.4 Assisted living

Smart environments and ambient assisted living are two paradigms which aim at supporting people in their daily living activities. Smart environment can be described as a small world consisting of small devices which are designed to make inhabitants' lives more comfortable [42]. Ambient assisted living shares the same aim but ambient assisted living services provides more personalized and more adaptive services to achieve interoperability, security

and accuracy [116]. Assisted living services are targeted at people with disabilities whereas smart environments do not have such a specific target group.

Since assisted living services handle health related data by storing and evaluating the data, protecting the privacy of the customer is in central role. Protection of privacy is particularly crucial for network-aided solutions. An example of such a solution is telemonitoring. A patient wears body sensors which measure, for example, the heart beat rate and blood pressure. The sensors are connected to the Internet for sending the data from the sensors to further analysis and storage [52]. Other examples of network-based assisted living solutions include AlarmNet [164] and iSenior [143].

It is also important that the health data is evaluated in a secure manner. This requires a protocol that does not compromise the data integrity and privacy. Several protocols have been proposed to achieve this goal. For example, Brickell et al. [31] use branching programs as the model for diagnostics tool. Lin et al. have improved the protocol: some of the decryption related computation load is moved to the cloud [103]. In publication VIII we propose a health monitoring system based on garbling schemes.

Chapter 4

Contributions

This thesis consists of eight journal and conference publications. Six of these publications, publications I-VI, contain theoretical results related to garbling schemes. These publications extend the results of [17, 18, 74]. The other two publications, publications VII and VIII, deal with applications of garbling schemes concentrating on privacy-preserving security services. Figure 4.1 show how the eight publications are related. We now briefly discuss the results and contributions of each publication.

4.1 Extending earlier results

Publication I extends the results of the seminal work of Bellare et al. [18] in several ways. The first results show the role of side-information in garbling. We show in publication I that the class of garbling schemes which are secure under a certain security notion and security model does not decrease when the side-information function is allowed to leak more information. As an example, the side-information function Φ_{topo} leaks more information than Φ_{size}. Therefore, according to the previously mentioned result, for certain security notions and models, there are at least as many garbling schemes that are secure over Φ_{topo} as there are secure garbling schemes over Φ_{size}.

The next generalization presented in publication I is related to the non-inclusions between the security classes of garbling schemes. Bellare et al. assume a certain type of side-information function (Φ_{topo}) when they prove the separation between simulation-based obliviousness and indistinguishability-based privacy as well as the separation between the authenticity notion and the notions privacy and obliviousness in the indistinguishability-based model. We remove the assumption of specific side-information function and prove that the separations hold for any side-information function.

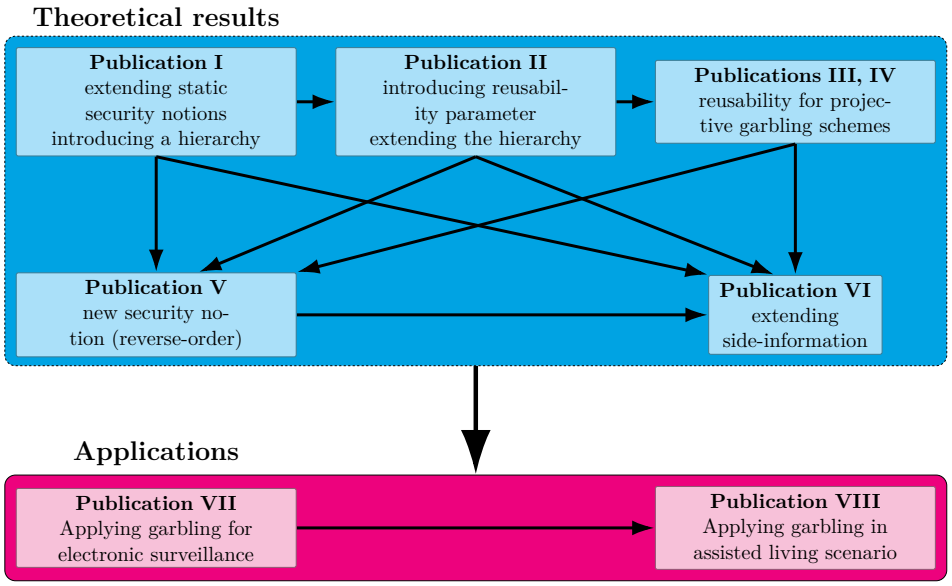


Figure 4.1: A roadmap to the articles included in this thesis.

We also show in publication I that, under certain assumptions, any garbling scheme achieves static indistinguishability-based privacy. In addition, we introduce corresponding assumptions so that any garbling scheme achieves static simulation-based privacy.

In addition to the generalizations mentioned above, we introduce new security notions for garbling schemes. These notions are named mod.ind, mod.sim, mod.ind2 and mod.sim2 in publication I. Out of these four notions, the new notions mod.ind and mod.sim seem to be of practical use. The notions mod.ind and mod.sim are useful, for instance, when the outcome of computations should be kept secret while the comparison (e.g. the equality) of outcomes would still be possible. Instead, the classes mod.ind2 and mod.sim2 are shown to be practically always empty due to too hard security requirements set for a garbling scheme. In later publications, we omit the security classes mod.ind2 and mod.sim2 due to this impracticality. In the publications hereafter, we replace the name mod by matchability-only (mao).

4.2 Reusability of garbled functions

The definitions of Bellare et al. [18, 17] as well as those in publication I support only one-time use of the same garbled function. This is undesirable from practical point-of-view: every time the same function should be evaluated with an argument, the garbled function must be computed again.

Goldwasser et al. introduced a way to construct reusable garbled circuits in [74]. Their solution is based on fully-homomorphic encryption, for which no efficient implementations are known. On the other hand, the solution allows the same function to be used arbitrarily many times.

Inspired by the two problems mentioned above, it is an interesting question whether there are garbling schemes that support at least some level of reusability but would still be practical (efficient in time and space consumption). To take the first steps towards the answer, we need new security definitions that capture the idea of "some level of reusability". In publication II, we introduce a new parameter, the reusability parameter $\ell \in \mathbb{N}$, which tells how many times the same garbled function can be used for garbled evaluation of f. In other words, instead of arbitrary reuse of the same garbled function $F = \mathtt{Gb}(f, 1^k)$, F could be re-used a limited number of times. There are applications for which this type of limited reusability would be as practical as having a chance for arbitrary reuse of the same garbled function.

Publication II provides several results related to the new classes of reusable garbling schemes. We show that the relations of security notions and security models are the same for every fixed value of ℓ. The relations are also exactly the same as for the static security classes. In addition, the classes of reusable, adaptively secure garbling schemes form an infinite chain with respect to the reusability parameter ℓ. The infiniteness is achieved by two results. A security class with threshold value $\ell + 1$ is properly included in the corresponding class with threshold value ℓ, unless both classes are empty. Secondly, a security class allowing the same garbling to be used arbitrarily many times is properly included in the corresponding security class with any threshold value $\ell \in \mathbb{N}$ (unless both are empty). Since there seem to be candidates in the class of arbitrary reusability (e.g. [74]), all the classes from reusability level $\ell = 1$ to arbitrary reusability are non-empty and the class with a greater reusability parameter is properly included into the class with a smaller reusability parameter.

We introduce a more comprehensive way of illustrating the relations between the classes of garbling schemes in publication II. We present the hierarchy as *a Cartesian product of directed graphs*. By Cartesian product $G_1 \times G_2$ we mean the directed graph having vertices in $V = V_1 \times V_2$. The directed edges of $G_1 \times G_2$ are found as follows. There is a directed edge from vertex $u = (u_1, u_2)$ to $v = (v_1, v_2)$ in $G_1 \times G_2$ whenever $u_1 = v_1$ and there is a directed edge from u_2 to v_2, or $u_2 = v_2$ and there is a directed edge from u_1 to v_1. In the presentation of the hierarchy for garbling schemes, the Cartesian product consists of three graphs. The first graph illustrates the relations between the security notions. The second graph shows the relations between the security models. The third graph is infinite and shows

the relations between the various adaptivity levels determined by the value of the reusability parameter ℓ.

4.3 Projectivity and reusability

We extend the concept of reusability for projective garbling schemes in publication III. Projective garbling schemes allow the user to feed the input in smaller pieces, for example, bit by bit. This feature increases the practicality of garbling schemes. Therefore, it is natural to extend reusability to projective garbling schemes and investigate the security notions in this special case as well.

In publication III we define a new concept of security, *bitwise adaptive security*, which generalizes the concept of fine-grained adaptivity in [17]. We obtain several results for these new classes of garbling schemes. Like in the general case of adaptivity, the classes of bitwise adaptively secure garbling schemes form an infinite chain with respect to the reusability parameter ℓ. In addition, the hierarchy for classes of bitwise adaptively secure garbling schemes is similar to the hierarchy of adaptively secure garbling schemes. We also show that a bitwise adaptively secure garbling scheme does not have to be adaptively secure. However, if restricted to the subset of projective garbling schemes, then the class of bitwise adaptively secure garbling scheme is included in the corresponding class of adaptively secure garbling schemes.

Publication III left open some questions related to bitwise adaptive security. Publication IV provides answers to these questions. The first solved question is related to the general existence of bitwise adaptively secure reusable garbling schemes: how long must the garbled argument be so that the scheme can belong to certain security class? In order to achieve a secure garbling scheme in bitwise adaptive setting, there are certain conditions which must be fulfilled. One of these requirements is related to the encryption algorithm $\mathtt{En}$. Namely, we prove in publication IV that the length of the encrypted version of the argument x must be at least $nc \log k + n$, where n is the length of x, c a constant and k the security parameter, under the condition that a projective garbling scheme $\mathcal{G}$ achieves adaptive privacy, obliviousness or matchability-only at reusability level $\ell = 2$.

The second solved open question concerns the relations between the classes of bitwise adaptively secure reusable garbling schemes. In publication III it was left open whether there are conditions under which indistinguishability-based security could imply simulation-based security. We provide such a condition by introducing a new variant for efficient invertibility of side-information function Φ and evaluation algorithm $\mathtt{ev}$ called bit- and componentwise efficient invertibility of $(\Phi, \mathtt{ev})$. The existence of a bit-

and componentwise efficient $(\Phi, \mathtt{ev})$-inverter then ascertains, that privacy in indistinguishability model implies privacy in simulation model.

4.4 Garbling the argument first

In all previous publications, it is assumed that the function f to be evaluated is known prior to its argument x. All security notions rely on this assumption: in all security games, the function is garbled either before or at the same time as the argument. However, in many practical situations, the argument is known before an applicable function is determined. Moreover, it might be that the same argument is to be run with several different functions. Using the established security definitions of a garbling scheme would yield the following: every time we want to use the same argument for a different function, we need to compute a new garbled function and a new garbled argument. To reduce the amount of unnecessary computations, we propose a new security definition, *reverse-order adaptive security*. In this model, first an argument is garbled in the security game, and only after getting the garbled argument the adversary needs to choose which functions to garble.

This generalization requires also a slight change in the definition of a garbling scheme. Instead of treating garbling schemes as 5-tuple of algorithms $(\mathtt{Gb}, \mathtt{En}, \mathtt{Ev}, \mathtt{De}, \mathtt{ev})$, we define a garbling scheme as a 6-tuple $(\mathtt{KeyGen}, \mathtt{Ga}, \mathtt{En}, \mathtt{Ev}, \mathtt{De}, \mathtt{ev})$. The only difference in these definitions is that we split the original function garbling algorithm `Gb` into two separate algorithms `KeyGen` and `Ga`. In previous models, the task of algorithm `Gb` is to compute the garbled function as well as the encryption and decryption keys e and d. In our new model, algorithm `KeyGen` is used for generating three keys g, e and d. The additional key g is used for computing the garbled function, so it is used as an input for algorithm `Ga` together with the function f. The other four algorithms are similar in both the new and the old model. Figure 4.2 illustrates the new model of garbling scheme.

We prove several results regarding this new security notion. First, we show that adaptivity and reverse-order adaptivity are quite different notions except for the static security: adaptive security does not imply reverse-order adaptivity and vice versa for $\ell \geq 2$. Secondly, we show that, similarly to adaptive security classes, the hierarchy of reverse-order adaptive security classes is either infinite or all security classes for $\ell \in \mathbb{N}$ are empty. Thirdly, for a fixed reusability parameter ℓ, the relations between the models and notions are exactly the same for both adaptive and reverse-order adaptive classes.

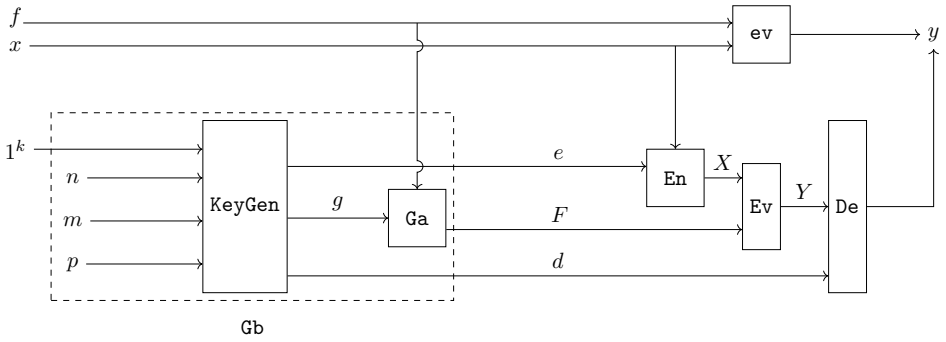


Figure 4.2: Generalized model of a garbling scheme. Compared to the definition in [18], the diagram is the same except of the garbling algorithm `Gb`, which in our model is split to two separate algorithms `KeyGen` and `Ga`.

4.5 Extending the model of side-information

As discussed in section 3.4, side-information function has a central role in measuring security of a garbling scheme – all security games depend on the choice of side-information function. Because of this central role of security definitions, it is crucial that the concept of side-information is appropriately defined.

The definition used by Bellare et al. in [18, 17] describes side-information function as a mapping which maps a function f into a bit string. From this bit string depending only on function f, at least the length of argument x, the length of final evaluation result y and the length of f must be efficiently computable. In the case of logical circuits, this is a natural requirement. The same argument does not hold for e.g. Turing machines: the length of the input and the length of the output cannot generally be determined by the Turing machine only, demonstrating the fundamental differences between circuits and Turing machines.

The issue mentioned above has implications to the definition of side-information function. The side-information function cannot depend only on the function f. In publication VI, the model of side-information function is extended to achieve better compatibility with other computation models than circuits. In the extended model, the side-information depends on the encryption key e, the decryption key d, the function f and the argument x.

The extended model also takes different types of attacks into account. In the established model, the algorithms `Gb`, `En` and `De` were considered to be run on secure environments. However, it is possible that these algorithms are targets of various side-channel attacks. This yields that information about the encryption key e, the decryption key d and computation of $\mathtt{ev}(f, x)$ might

leak. Therefore, it is natural that the extended model of side-information is dependent on e, d, f and x.

Extending the model of side-information has also a positive side-effect. The new model of side-information simplifies the various security definitions of garbling schemes. In publication VI, we provide the new, simplified game definitions. We prove that the new definitions are compatible with the old definitions, when restricting the computation model to logic circuits. Furthermore, publication VI shows that all the known relations for security classes are preserved in the new model.

4.6 Garbling in privacy-preserving applications

Original publications VII and VIII suggest two scenarios in which garbling schemes can be used as a technique to enhance privacy for sensitive information. In publication VII we consider how to implement a privacy-preserving electronic surveillance system. In publication VIII, we extend electronic surveillance to more holistic monitoring that includes processing health data. This kind of application could be used e.g. for assisted living services.

In publication VII, we present a novel way of using garbling schemes to achieve privacy preservation in electronic surveillance and illustrate the power of garbling with an example scenario. An elderly person living alone is subscribing to a security service that includes electronic surveillance. The surveillance data is analyzed by a security company that has outsourced its data services onto a third-party cloud. Garbling allows the private analysis of the surveillance data on cloud - the cloud learns neither the surveillance data nor the analytics tool.

The advantage of our solution is that garbling provides flexibility for the system. The surveillance analytics tool can be almost anything, from comparisons to complex machine learning algorithms. Moreover, the function f can be changed without the need to reconfigure the whole system, easing the system maintenance.

The biggest obstacles to implementing the described system is related to the implementation of efficient garbling schemes. There exist efficient garbling schemes (see section 4.3) that support one-time use of the garbling scheme. But regarbling the analytics tool again for every surveillance data entry is not optimal from practical point-of-view. Reusable garbled circuits would solve this problem - however efficient garbling schemes supporting reusable garbled circuits are not yet known.

The example scenario presented in VII is not the only possible application for garbling. As another related example, a monitoring system can be installed in the homes of people using the services for assisted living. The party monitoring the data should not learn the habits of the person using

the system beyond the situations in which the person needs help. In this scenario, the security company may provide the monitoring services to the company providing the services for assisted living. This adds complexity to preserve privacy.

In publication VIII, we took the challenge of designing a privacy preserving monitoring system including both security and health features. Our example scenario is more complex, including four parties instead of three: a client, a security company, an assisted living service provider and a third-party cloud service provider. In this publication, we show how a garbling scheme can be used among these four parties. In addition, we demonstrate the efficiency of our solution by implementing a simplistic health assessment function and performing garbled computations on that function. The outcome of the experiment was that the garbled evaluation of this simple function is extremely fast, including the garbling phase. Also more complex functions appeared to be fast enough for an application requiring fast response times. As an example, a circuit with 400 000 gates is garbled and evaluated in less than 250 microseconds and a circuit with roughly 15 million gates is garbled and evaluated in less than 1 second.

Chapter 5

Conclusions and future work

In this thesis, we propose several new security classes for garbling schemes. We introduce a new security notion, matchability-only, which may be of practical interest since it provides an intermediate form between privacy and obliviousness. Furthermore, the amount of secure garbling schemes does not decrease when the matchability-only security notion is used instead of the privacy or the obliviousness notion.

Garbled circuits support only single-use of the same garbling. However, computations are often done with the same function but different arguments. For garbled circuits, this would mean that computing the garbled circuit should be done again every time a new evaluation takes place. This generates lots of unnecessary computations. As a solution, we have introduced an idea of leveled reusability in form of a reusability parameter. This parameter characterizes how many times it is secure to use the same garbled function with different argument queries.

We extended the reusability results to projective garbling schemes, which are a special case of general garbling schemes. If a garbling scheme is projective, then it supports garbling the argument in smaller pieces, even bitwise.

Reverse-order garbling is a new form of adaptivity, which flips the roles of the function and the argument. In many practical situations, the data stays the same whereas different algorithms are applied to it. As an example, medical information of a patient stays invariant, whereas different medical analytics algorithms may be applied to the data. Our new notion extends the security of garbling schemes also in this direction.

As a result of investigating the security notions and the relations between the security classes, we obtain a hierarchy which is represented as a directed graph Cartesian product. The product consists of three components: the security notion, the security model and the level of adaptivity. The resulting hierarchy is presented in fig. 5.1.

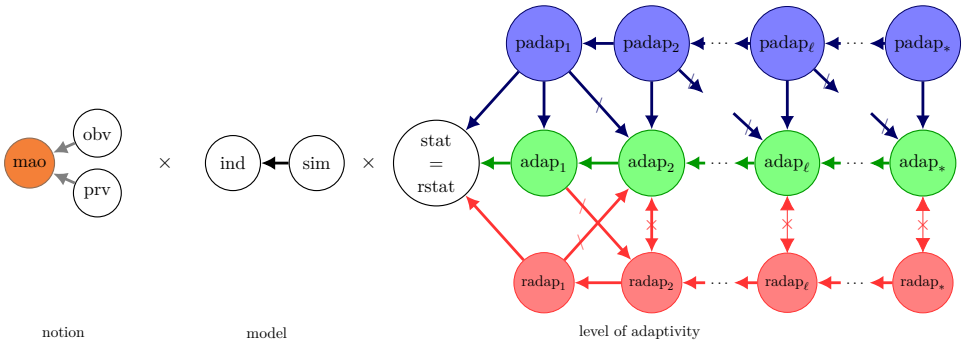


Figure 5.1: Hierarchy of garbling scheme classes represented as a Cartesian product of graphs. Lines $\rightarrow$ show the inclusions between security classes and lines $\nrightarrow$ show that there is no inclusion. Colored vertices give the new security classes introduced in this work.

We also make a further theoretical remark concerning the fundamentals of the security concepts for garbling schemes. The security of garbling schemes is parameterized by side-information, which informally captures the information that is allowed to be leaked during the garbled evaluation. Garbled circuits leak different information compared to e.g. Turing machines. However, the established definition of garbling schemes relies on a model that fits circuits but fails to model e.g. Turing machines. We propose a new model of side-information that takes into account different models of computation in a more appropriate way. We prove that the new model is fully compatible with the established model when restricted to circuit garbling schemes.

This thesis also provides proposals for applications of garbling schemes. We consider the possibility of using garbling schemes for privacy-preserving electronic surveillance. We extend the idea of surveillance into more holistic monitoring of a client. In addition to surveillance, the various sensors and devices collect private health information of the client, in which case the system can be used for assisted living services. By implementing a simple, but still practical health indication function we show that garbling schemes are a potential solution for applications processing privacy-sensitive information.

The advantage of garbling schemes is that efficient implementations, especially for circuits, are known. However, garbled circuits have also some drawbacks. One of the biggest problems for garbling schemes is that the computational load is not distributed optimally. It is the client's task to generate the appropriate representation of the algorithm, generate the keys for garbling as well as generate the garbled function and the garbled argument whereas the evaluator only runs the garbled evaluation. An important

question for future research is whether there exist methods to distribute the workload more evenly.

Another topic of future interest is whether there is need for further security definitions of garbling schemes. In this work, we have presented some, but practical applications may require new definitions. Still, some of the notions presented in this work do not yet have implementation. It would be interesting to see whether there is an efficient garbling scheme providing reusability for some $\ell \in \mathbb{N}$. Currently we know, that efficient solutions are known in static case as well as for $\ell = 1$. Furthermore, Goldwasser's reusable garbled circuits [74] provide a candidate for reusability class enabling arbitrary reusability. However, these reusable garbled circuits do not have a known efficient solution, due to the fact that the construction is based on FHE for which no efficient solution is known.

If the improvements of garbling schemes become reality, garbling schemes may gain new application areas, even in cryptographic applications. Then, garbling schemes would truly achieve the status of being a cryptographic primitive rather than a mere cryptographic technique.

Bibliography

[1] H. Abie and I. Balasingham. Risk-based Adaptive Security for Smart IoT in eHealth. In *Proceedings of the 7th International Conference on Body Area Networks*, BodyNets '12, pages 269–275, ICST, Brussels, Belgium, Belgium, 2012. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[2] A. Afshar, Z. Hu, P. Mohassel, and M. Rosulek. How to Efficiently Evaluate RAM Programs with Malicious Security. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, volume 9056 of *Lecture notes in Computer Science*, pages 702–729. Springer Berlin Heidelberg, 2015.

[3] R. Agrawal and R. Srikant. Privacy-preserving Data Mining. *SIGMOD Rec.*, 29(2):439–450, May 2000.

[4] W. Aiello, Y. Ishai, and O. Reingold. Priced Oblivious Transfer: How to Sell Digital Goods. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, EUROCRYPT '01, pages 119–135, London, UK, UK, 2001. Springer-Verlag.

[5] B. Anckaert, B. D. Sutter, and K. D. Bosschere. Software Piracy Prevention Through Diversity. In *Proceedings of the 4th ACM Workshop on Digital Rights Management*, DRM '04, pages 63–71. ACM, 2004.

[6] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. SETIhome: An experiment in public-resource computing. *Communications of the ACM*, 45(11):333–342, 2009.

[7] L. Atzori, A. Iera, and G. Morabito. The Internet of Things: A Survey. *Computer Networks*, 54(15):2787–2805, 2010.

[8] D. Aucsmith. Tamper Resistant Software: An Implementation. In *Proceedings of 1st International Information Hiding Workshop (IHW)*, volume 1174 of *Lecture notes in Computer Science*, pages 317–333. Springer, 1996.

[9] D. Aucsmith and G. Graunke. Tamper Resistant Methods and Apparatus. Patent US 5892899, 1999. year filed 1996; year published 1999.

[10] B. Barak. How to go beyond the black-box simulation barrier. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science, 2001.*, pages 106–115, Oct 2001.

[11] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (Im)possibility of Obfuscating Programs. In J. Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture notes in Computer Science*, pages 1–18. Springer Berlin Heidelberg, 2001.

[12] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (Im)Possibility of Obfuscating Programs. *Journal of the ACM*, 59(2):6:1–6:48, May 2012.

[13] M. Barni, J. Guajardo, and R. Lazzeretti. Privacy preserving evaluation of signal quality with application to ECG analysis. In *Information Forensics and Security (WIFS), 2010 IEEE International Workshop on*, pages 1–6, Dec 2010.

[14] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *Proc. of the* 22^{nd} *STOC*, pages 503–513. ACM, 1990.

[15] M. Bellare and S. Goldwasser. New Paradigms for Digital Signatures and Message Authentication Based on Non-Interactive Zero Knowledge Proofs. In G. Brassard, editor, *Advances in Cryptology – CRYPTO'89 Proceedings*, volume 435 of *Lecture notes in Computer Science*, pages 194–211. Springer New York, 1990.

[16] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway. Efficient garbling from a fixed-key blockcipher. In *Proc. of Symposium on Security and Privacy 2013*, pages 478–492. IEEE, 2013.

[17] M. Bellare, V. T. Hoang, and P. Rogaway. Adaptively secure garbling scheme with applications to one-time programs and secure outsourcing. In *Proc. of Asiacrypt 2012*, volume 7685 of LNCS, pages 134–153. Springer, 2012.

[18] M. Bellare, V. T. Hoang, and P. Rogaway. Foundations of Garbled Circuits. In *Proc. of ACM Computer and Communications Security (CCS'12)*, pages 784–796. ACM, 2012.

[19] M. Bellare and S. Micali. Non-Interactive Oblivious Transfer and Applications. In G. Brassard, editor, *Advances in Cryptology – CRYPTO '89 Proceedings*, volume 435 of *Lecture notes in Computer Science*, pages 547–557. Springer New York, 1990.

[20] M. Bellare and P. Rogaway. Code-Based Game-Playing Proofs and the Security of Triple Encryption. *Advances in Cryptology – EUROCRYPT2006*, 4004 of LNCS:409–426, 2006.

[21] M. Ben-Or, O. Goldreich, S. Goldwasser, J. Håstad, J. Kilian, S. Micali, and P. Rogaway. Everything Provable is Provable in Zero-Knowledge. In S. Goldwasser, editor, *Advances in Cryptology – CRYPTO'88*, volume 403 of *Lecture notes in Computer Science*, pages 37–56. Springer New York, 1990.

[22] J. Benaloh. Dense Probabilistic Encryption. In *Proceedings of the Workshop on Selected Areas of Cryptography*, pages 120–128, 1994.

[23] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. The Road from Panama to Keccak via RadioGatún. In H. Handschuh, S. Lucks, B. Preneel, and P. Rogaway, editors, *Symmetric Cryptography*, number 09031 in Dagstuhl Seminar Proceedings. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009.

[24] O. Billet, H. Gilbert, and C. Ech-Chatbi. Cryptanalysis of a White Box AES Implementation. In H. Handschuh and M. A. Hasan, editors, *Selected Areas in Cryptography*, volume 3357 of *Lecture notes in Computer Science*, pages 227–240. Springer Berlin Heidelberg, 2005.

[25] N. Bitansky and O. Paneth. Point Obfuscation and 3-Round Zero-Knowledge. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, pages 190–208, 2012.

[26] M. Blum, P. Feldman, and S. Micali. Non-interactive Zero-knowledge and Its Applications. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 103–112, New York, NY, USA, 1988. ACM.

[27] D. Boneh, E. Goh, and K. Nissim. Evaluating 2-DNF Formulas on Ciphertexts. In J. Kilian, editor, *Theory of Cryptography*, volume 3378 of *Lecture notes in Computer Science*, pages 325–341. Springer Berlin Heidelberg, 2005.

[28] J. Borello and L. Mé. Code obfuscation techniques for metamorphic viruses. *Journal in Computer Virology*, 4(3):211–220, 2008.

[29] L. T. Brandão. Secure Two-Party Computation with Reusable Bit-Commitments, via a Cut-and-Choose with Forge-and-Lose Technique. In K. Sako and P. Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013*, volume 8270 of *Lecture notes in Computer Science*, pages 441–463. Springer Berlin Heidelberg, 2013.

[30] E. F. Brickell and Y. Yacobi. On Privacy Homomorphisms (Extended Abstract). In D. Chaum and W. L. Price, editors, *Advances in Cryptology – EUROCRYPT'87*, volume 304 of *Lecture notes in Computer Science*, pages 117–125. Springer Berlin Heidelberg, 1988.

[31] J. Brickell, D. E. Porter, V. Shmatikov, and E. Witchel. Privacy-preserving Remote Diagnostics. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pages 498–507. ACM, 2007.

[32] R. Canetti, J. Kilian, E. Petrank, and A. Rosen. Black-Box Concurrent Zero-Knowledge Requires (Almost) Logarithmically Many Rounds. *SIAM journal of Computing*, 32(1):1–47, Jan. 2003.

[33] H. Chang and M. Atallah. Protecting Software Code by Guards. In *Proceedings of the 1st ACM Workshop on Digital Rights Management (DRM 2001)*, volume 2320 of *Lecture notes in Computer Science*, pages 160–175. Springer, 2002.

[34] Y. Chen, R. Venkatesan, M. Cary, R. Pang, S. Sinha, and M. Jakubowski. Oblivious Hashing: A Stealthy Software Integrity Verification Primitive. In *Proceedings of the 5th Information Hiding Workshop (IHW)*, volume 2578 of *Lecture notes in Computer Science*, pages 400–414. Springer, 2002.

[35] S. G. Choi, J. Katz, R. Kumaresan, and H. Zhou. On the Security of the "Free-XOR" Technique. In R. Cramer, editor, *Theory of Cryptography*, volume 7194 of *Lecture notes in Computer Science*, pages 39–53. Springer Berlin Heidelberg, 2012.

[36] S. Chow, P. Eisen, H. Johnson, and P. C. V. Oorschot. White-Box Cryptography and an AES Implementation. In K. Nyberg and H. Heys, editors, *Selected Areas in Cryptography*, volume 2595 of *Lecture notes in Computer Science*, pages 250–270. Springer Berlin Heidelberg, 2003.

[37] S. Chow, P. Eisen, H. Johnson, and P. C. van Oorschot. A White-Box DES Implementation for DRM Applications. In J. Feigenbaum, editor, *Digital Rights Management*, volume 2696 of *Lecture notes in Computer Science*, pages 1–15. Springer Berlin Heidelberg, 2003.

[38] S. Chow, Y. Gu, H. Johnson, and V. A. Zakharov. An Approach to the Obfuscation of Control-Flow of Sequential Computer Programs. In G. I. Davida and Y. Frankel, editors, *Information Security*, volume 2200 of *Lecture notes in Computer Science*, pages 144–155. Springer Berlin Heidelberg, 2001.

[39] F. B. Cohen. Operating system protection through program evolution. *Computers & Security*, 12(6):565 – 584, 1993.

[40] C. Collberg, C. Thomborson, and D. Low. A Taxonomy of Obfuscating Transformations. Technical report, The University of Auckland, 1997.

[41] C. S. Collberg and C. Thomborson. Watermarking, Tamper-proffing, and Obfuscation: Tools for Software Protection. *IEEE Trans. Softw. Eng.*, 28(8):735–746, Aug. 2002.

[42] D. Cook and S. Das. *Smart Environments: Technology, Protocols and Applications (Wiley series on Parallel and Distributed Computing)*. Wiley-Interscience, 2004.

[43] C. Crépeau. Equivalence Between Two Flavours of Oblivious Transfers. In C. Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, volume 293 of *Lecture notes in Computer Science*, pages 350–354. Springer Berlin Heidelberg, 1988.

[44] G. D. Crescenzo, T. T. Malkin, and R. Ostrovsky. Single Database Private Information Retrieval Implies Oblivious Transfer. In B. Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture notes in Computer Science*, pages 122–138. Springer Berlin Heidelberg, 2000.

[45] I. Damgård and M. Jurik. A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In *Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography*, PKC '01, pages 119–136, London, UK, UK, 2001. Springer-Verlag.

[46] C. Delerablée, T. Lepoint, P. Paillier, and M. Rivain. White-Box Security Notions for Symmetric Encryption Schemes. In T. Lange, K. Lauter, and P. Lisoněk, editors, *Selected Areas in Cryptography – SAC 2013*, volume 8282 of *Lecture notes in Computer Science*, pages 247–264. Springer Berlin Heidelberg, 2014.

[47] C. Doukas, T. Pliakas, and I. Maglogiannis. Mobile healthcare information management utilizing Cloud Computing and Android OS. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 1037–1040, Aug 2010.

[48] W. Du and M. J. Atallah. Secure Multi-party Computation Problems and Their Applications: A Review and Open Problems. In *Proceedings of the 2001 Workshop on New Security Paradigms*, NSPW '01, pages 13–22. ACM, 2001.

[49] C. Dwork, M. Naor, and A. Sahai. Concurrent Zero-knowledge. *Journal of the ACM*, 51(6):851–898, Nov. 2004.

[50] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, Jul 1985.

[51] S. Even, O. Goldreich, and A. Lempel. A Randomized Protocol for Signing Contracts. *Commun. ACM*, 28(6):637–647, June 1985.

[52] G. Fortino, G. Di Fatta, M. Pathan, and A. Vasilakos. Cloud-assisted body area networks: state-of-the-art and future challenges. *Wireless Networks*, 20(7):1925–1938, 2014.

[53] M. Franz. E unibus pluram: massive-scale software diversity as a defense mechanism. In *Proceedings of the 2010 Workshop on New Security Paradigms*. ACM, 2010.

[54] T. K. Frederiksen, T. P. Jakobsen, and J. B. Nielsen. Faster Maliciously Secure Two-Party Computation Using the GPU. In M. Abdalla and R. D. Prisco, editors, *Security and Cryptography for Networks*, volume 8642 of *Lecture notes in Computer Science*, pages 358–379. Springer International Publishing, 2014.

[55] T. K. Frederiksen, T. P. Jakobsen, J. B. Nielsen, P. S. Nordholt, and C. Orlandi. MiniLEGO: Efficient Secure Two-Party Computation from General Assumptions. In T. Johansson and P. Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture notes in Computer Science*, pages 537–556. Springer Berlin Heidelberg, 2013.

[56] T. K. Frederiksen, J. B. Nielsen, and C. Orlandi. Privacy-Free Garbled Circuits with Applications to Efficient Zero-Knowledge. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, volume 9057 of *Lecture notes in Computer Science*, pages 191–219. Springer Berlin Heidelberg, 2015.

[57] K. B. Frikken and M. J. Atallah. Privacy Preserving Electronic Surveillance. In *Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society*, WPES '03, pages 45–52, New York, NY, USA, 2003. ACM.

[58] S. Garg, C. Gentry, S. Halevi, and R. M. Two-Round Secure MPC from Indistinguishability Obfuscation. In Y. Lindell, editor, *Theory of Cryptography*, volume 8349 of *Lecture notes in Computer Science*, pages 74–94. Springer Berlin Heidelberg, 2014.

[59] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits. In *IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS), 2013*, pages 40–49, Oct 2013.

[60] R. Gennaro, C. Gentry, and B. Parno. Non-interactive verfiable computing: Outsourcing computation to untrusted workers. In *Proc. of CRYPTO 2010*, volume 6223 of LNCS, pages 465–482. Springer, 2010.

[61] C. Gentry. Computing on the Edge of Chaos: Structure and Randomness in Encrypted Computation. Cryptology ePrint Archive, Report 2014/610, 2014. `eprint.iacr.org`.

[62] C. Gentry, S. Halevi, S. Lu, R. Ostrovsky, M. Raykova, and D. Wichs. Garbled RAM Revisited. In *Proc. of* 33^{rd} *Eurocrypt*, volume 8441 of LNCS, pages 405–422, 2014.

[63] G. Gentry. Fully Homomorphic Encryption Using Ideal Lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. ACM.

[64] G. Gentry and S. Halevi. Implementing Gentry's Fully-Homomorphic Encryption Scheme. In K. G. Paterson, editor, *Advances in Cryptology – EUROCRYPT'2011*, volume 6632 of *Lecture notes in Computer Science*, pages 129–148. Springer Berlin Heidelberg, 2011.

[65] G. Gentry, S. Halevi, and N. P. Smart. Homomorphic Evaluation of the AES Circuit. In R. Safavi-Naini and R. Canetti, editors, *Advances in Cryptology – CRYPTO'2012*, volume 7417 of *Lecture notes in Computer Science*, pages 850–867. Springer Berlin Heidelberg, 2012.

[66] G. Gentry, S. Halevi, and V. Vaikuntanathan. A Simple BGN-Type Cryptosystem from LWE. In H. Gilbert, editor, *Advances in Cryptology – EUROCRYPT'2010*, volume 6110 of *Lecture notes in Computer Science*, pages 506–522. Springer Berlin Heidelberg, 2010.

[67] O. Goldreich. Concurrent Zero-knowledge with Timing, Revisited. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 332–340, New York, NY, USA, 2002. ACM.

[68] O. Goldreich. *Foundations of Cryptography: volume 1.* Cambridge University Press, New York, NY, USA, 2006.

[69] O. Goldreich and H. Krawczyk. On the Composition of Zero-Knowledge Proof Systems. *SIAM journal of Computing*, 25(1):169–192, Feb. 1996.

[70] O. Goldreich, S. Micali, and A. Wigderson. How to Play ANY Mental Game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 218–229, New York, NY, USA, 1987. ACM.

[71] O. Goldreich, S. Micali, and A. Wigderson. Proofs That Yield Nothing but Their Validity or All Languages in NP Have Zero-knowledge Proof Systems. *Journal of the ACM*, 38(3):690–728, July 1991.

[72] O. Goldreich and Y. Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, 1994.

[73] O. Goldreich and R. Ostrovsky. Software Protection and Simulation on Oblivious RAMs. *Journal of the ACM*, 43(3):431–473, 1996.

[74] S. Goldwasser, Y. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable Garbled Circuits and Succinct Functional Encryption. In *Proc. of the* 45^{th} *STOC*, pages 555–564. ACM, 2013.

[75] S. Goldwasser, Y. Kalai, and G. Rothblum. One-Time Programs. In *Proc. of CRYPTO 2008*, volume 5157 of LNCS, pages 39–56. Springer, 2008.

[76] S. Goldwasser and Y. T. Kalai. On the impossibility of obfuscation with auxiliary input. In *46th Annual IEEE Symposium on Foundations of Computer Science, 2005. FOCS 2005.*, pages 553–562, Oct 2005.

[77] S. Goldwasser and S. Micali. Probabilistic Encryption & How to Play Mental Poker Keeping Secret All Partial Information. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, pages 365–377, New York, NY, USA, 1982. ACM.

[78] S. Goldwasser, S. S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof-systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, pages 291–304, New York, NY, USA, 1985. ACM.

[79] L. Goubin, J. Masereel, and M. Quisquater. Cryptanalysis of White Box DES Implementations. In C. Adams, A. Miri, and M. Wiener, editors, *Selected Areas in Cryptography*, volume 4876 of *Lecture notes in Computer Science*, pages 278–295. Springer Berlin Heidelberg, 2007.

[80] J. Groth. Non-interactive Zero-Knowledge Arguments for Voting. In J. Ioannidis, A. Keromytis, and M. Yung, editors, *Applied Cryptography and Network Security*, volume 3531 of *Lecture notes in Computer Science*, pages 467–482. Springer Berlin Heidelberg, 2005.

[81] S. Gueron, Y. Lindell, A. Nof, and B. Pinkas. Fast Garbling of Circuits Under Standard Assumptions. Cryptology ePrint Archive, Report 2015/751, 2015. `http://eprint.iacr.org/`.

[82] S. Hada. Zero-Knowledge and Code Obfuscation. In T. Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture notes in Computer Science*, pages 443–457. Springer Berlin Heidelberg, 2000.

[83] S. Hada. Secure Obfuscation for Encrypted Signatures. In H. Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture notes in Computer Science*, pages 92–112. Springer Berlin Heidelberg, 2010.

[84] C. Hazay and Y. Lindell. *Efficient Secure Two-Party Protocols*. Springer Berlin Heidelberg, 2010.

[85] D. Hoang and L. Chen. Mobile Cloud for Assistive Healthcare (MoCAsH). In *Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific*, pages 325–332, Dec 2010.

[86] J. I. Hong and J. A. Landay. An Architecture for Privacy-sensitive Ubiquitous Computing. In *Proceedings of the 2Nd International Conference on Mobile Systems, Applications, and Services*, MobiSys '04, pages 177–189. ACM, 2004.

[87] B. Horne, L. Matheson, C. Sheehan, and R. Tarjan. Dynamic Self-Checking Techniques for Improved Tamper Resistance. In *Proceedings of the 1st ACM Workshop on Digital Rights Management (DRM 2001)*, volume 2320 of *Lecture notes in Computer Science*, pages 141–159. Springer, 2002.

[88] Y. Huang, J. Katz, V. Kolesnikov, R. Kumaresan, and A. J. Malozemoff. Amortizing Garbled Circuits. In J. A. Garay and R. Gennaro, editors, *Advances in Cryptology – CRYPTO 2014*, volume 8617 of *Lecture notes in Computer Science*, pages 458–475. Springer Berlin Heidelberg, 2014.

[89] M. R. Inc. GIMPS Home. `http://www.mersenne.org/`. Accessed June 25, 2014.

[90] Y. Ishai and E. Kushilevitz. Private simultaneous messages protocols with applications. In *Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems, 1997*, pages 174–183, Jun 1997.

[91] Y. Ishai and A. Paskin. Evaluating Branching Programs on Encrypted Data. In S. P. Vadhan, editor, *Theory of Cryptography*, volume 4392 of *Lecture notes in Computer Science*, pages 575–594. Springer Berlin Heidelberg, 2007.

[92] W. A. Jansen. Cloud Hooks: Security and Privacy Issues in Cloud Computing. In *Proc. of* 44^{th} *Hawaii International Conference on System Sciences (HICSS)*, pages 1–10, 2011.

[93] N. Karthikeyan and R. Sukanesh. Cloud Based Emergency Health Care Information Service in India. *Journal of Medical Systems*, 36(6):4031–4036, 2012.

[94] S. Keelveedhi. JustGarble. `http://cseweb.ucsd.edu/groups/justgarble/`. Accessed: 2014-10-13.

[95] F. Kerschbaum. Oblivious Outsourcing of Garbled Circuit Generation. In *ACM SAC*, 2015.

[96] J. Kilian. Founding Crytpography on Oblivious Transfer. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 20–31, New York, NY, USA, 1988. ACM.

[97] V. Kolesnikov and R. Kumaresan. Improved Secure Two-Party Computation via Information-Theoretic Garbled Circuits. In I. Visconti and R. D. Prisco, editors, *Security and Cryptography for Networks*, volume 7485 of *Lecture notes in Computer Science*, pages 205–221. Springer Berlin Heidelberg, 2012.

[98] V. Kolesnikov, P. Mohassel, and M. Rosulek. FleXOR: Flexible Garbling for XOR Gates That Beats Free-XOR. In J. A. Garay and R. Gennaro, editors, *Advances in Cryptology – CRYPTO 2014*, volume 8617 of *Lecture notes in Computer Science*, pages 440–457. Springer Berlin Heidelberg, 2014.

[99] V. Kolesnikov and T. Schneider. Improved Garbled Circuit: Free XOR Gates and Applications. In L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldòrsson, A. Ingòlfsdòttir, and I. Walukiewicz, editors, *Automata, Languages and Programming*, volume 5126 of *Lecture notes in Computer Science*, pages 486–498. Springer Berlin Heidelberg, 2008.

[100] D. Kozlov, J. Veijalainen, and Y. Ali. Security and Privacy Threats in IoT Architectures. In *Proceedings of the 7th International Conference on Body Area Networks*, BodyNets '12, pages 256–262, ICST, Brussels, Belgium, Belgium, 2012. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[101] S. Laur and H. Lipmaa. A New Protocol for Conditional Disclosure of Secrets and Its Applications. In J. Katz and M. Yung, editors, *Applied Cryptography and Network Security*, volume 4521 of *Lecture notes in Computer Science*, pages 207–225. Springer Berlin Heidelberg, 2007.

[102] M. Layouni, K. Verslype, M. T. Sandıkkaya, B. De Decker, and H. Vangheluwe. Privacy-Preserving Telemonitoring for eHealth. In E. Gudes and J. Vaidya, editors, *Data and Applications Security XXIII*, volume 5645 of *Lecture notes in Computer Science*, pages 95–110. Springer Berlin Heidelberg, 2009.

[103] H. Lin, J. Shao, C. Zhang, and Y. Fang. CAM: Cloud-Assisted Privacy Preserving Mobile Health Monitoring. *Information Forensics and Security, IEEE Transactions on*, 8(6):985–997, June 2013.

[104] Y. Lindell. Fast Cut-and-Choose Based Protocols for Malicious and Covert Adversaries. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, volume 8043 of *Lecture notes in Computer Science*, pages 1–17. Springer Berlin Heidelberg, 2013.

[105] Y. Lindell and B. Pinkas. Privacy Preserving Data Mining. In *Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '00, pages 36–54. Springer-Verlag, 2000.

[106] Y. Lindell and B. Pinkas. An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries. In M. Naor, editor, *Advances in Cryptology - EUROCRYPT 2007*, volume 4515 of *Lecture notes in Computer Science*, pages 52–78. Springer Berlin Heidelberg, 2007.

[107] Y. Lindell and B. Pinkas. A proof of security of Yao's protocol for secure two-party computation. *Journal of Cryptology*, 22(2):161–188, 2009.

[108] Y. Lindell and B. Pinkas. Secure Two-Party Computation via Cut-and-Choose Oblivious Transfer. *Journal of Cryptology*, 25(4):680–722, 2012.

[109] H. Löhr, A.-R. Sadeghi, and M. Winandy. Securing the e-Health Cloud. In *Proceedings of the 1st ACM International Health Informatics Symposium*, IHI '10, pages 220–229. ACM, 2010.

[110] S. Lu and R. Ostrovsky. How to Garble RAM Programs. In *Proc. of* 32^{nd} *Eurocrypt*, volume 7881 of LNCS, pages 719–734, 2013.

[111] R. Luo, X. Lai, and R. You. A new attempt of white-box AES implementation. In *2014 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*, pages 423–429, Oct 2014.

[112] B. Lynn, M. Prabhakaran, and A. Sahai. Positive Results and Techniques for Obfuscation. In C. Cachin and J. L. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture notes in Computer Science*, pages 20–39. Springer Berlin Heidelberg, 2004.

[113] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay - a secure two-party computation system. In *In USENIX Security Symposium*, pages 287–302, 2004.

[114] C. A. Melchor, P. Gaborit, and J. Herranz. Additively Homomorphic Encryption with d-Operand Multiplications. In T. Rabin, editor, *Advances in Cryptology – CRYPTO'2010*, volume 6223 of *Lecture notes in Computer Science*, pages 138–154. Springer Berlin Heidelberg, 2010.

[115] P. Mell and T. Grance. The NIST Definition of Cloud Computing. Special Publications 800-145, NIST, 2011. available at `http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf`.

[116] M. Memon, S. R. W. C. F. Pedersen, F. H. A. Beevi, and F. O. Hansen. Ambient Assisted Living Healthcare Frameworks, Platforms, Standards, and Quality Attributes. *Sensors*, 14(3):4312–4341, 2014.

[117] T. Meskanen, V. Niemi, and N. Nieminen. Classes of Garbling Schemes. *Infocommunications journal*, V(3):8–16, 2013.

[118] T. Meskanen, V. Niemi, and N. Nieminen. Garbling in Reverse Order. In *The 13th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-14)*, pages 53–60. IEEE, 2014.

[119] T. Meskanen, V. Niemi, and N. Nieminen. Hierarchy for Classes of Projective Garbling Schemes. In *International Conference on Information and Communications Technologies (ICT 2014)*, pages 1–8. IEEE, 2014.

[120] T. Meskanen, V. Niemi, and N. Nieminen. On Reusable Projective Garbling Schemes. In *2014 IEEE International Conference on Computer and Information Technology (CIT 2014)*, pages 315–322. IEEE, 2014.

[121] T. Meskanen, V. Niemi, and N. Nieminen. Extended Model of Side-Information in Garbling. In *The 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-15)*, pages 950–957. IEEE, 2015.

[122] T. Meskanen, V. Niemi, and N. Nieminen. Hierarchy for Classes of Garbling Schemes. *Studia Scientiarum Mathematicarium Hungarica*, 52(2):1–12, 2015.

[123] T. Meskanen, V. Niemi, and N. Nieminen. How to Use Garbling for Privacy Preserving Electronic Surveillance Services. *Cyber Security and Mobility*, 4(1):41–64, 2015.

[124] Microsoft©. VirTool:SWF/Obfuscator.F. https://www.microsoft.com/security/portal/threat/encyclopedia/Entry.aspx?Name=VirTool Accessed July 27th, 2015.

[125] Y. D. Mulder. *White-Box Cryptography – Analysis of White-Box AES Implementations*. PhD thesis, Katholieke Universiteit Leuven – Faculty of Engineering Science, 2014.

[126] D. Naccache and J. Stern. A New Public Key Cryptosystem Based on Higher Residues. In *Proceedings of the 5th ACM Conference on Computer and Communications Security*, CCS '98, pages 59–66, New York, NY, USA, 1998. ACM.

[127] M. Naor and B. Pinkas. Efficient Oblivious Transfer Protocols. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, pages 448–457, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.

[128] J. B. Nielsen and C. Orlandi. LEGO for Two-Party Secure Computation. In O. Reingold, editor, *Theory of Cryptography*, volume 5444 of *Lecture notes in Computer Science*, pages 368–386. Springer Berlin Heidelberg, 2009.

[129] N. Nieminen and A. Lepistö. Privacy-Preserving Security Monitoring for Assisted Living Services. In *Proceedings of the 17th International Symposium on Health Information Management Research (ISHIMR 2015)*, pages 189–199. York St. John Universty & University of Sheffield, 2015.

[130] M. Nkosi and F. Mekuria. Cloud Computing for Enhanced Mobile Health Applications. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 629–633, Nov 2010.

[131] K. Nuida. A Simple Framework for Noise-Free Construction of Fully Homomorphic Encryption from a Special Class of Non-Commutative Groups. Cryptology ePrint Archive, Report 2014/097, 2014. urleprint.iacr.org.

[132] N. I. of Standards and Technology. *FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions.* pub-NIST, pub-NIST:adr, Aug. 2015. Supersedes FIPS PUB 202 2014 May.

[133] T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In K. Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture notes in Computer Science*, pages 308–318. Springer Berlin Heidelberg, 1998.

[134] V. Oleshchuk. Internet of things and privacy preserving technologies. In *1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology, 2009. Wireless VITAE 2009.*, pages 336–340, 2009.

[135] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In J. Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture notes in Computer Science*, pages 223–238. Springer Berlin Heidelberg, 1999.

[136] V. Pande. Foldinghome. `http://folding.stanford.edu/`. Accessed June 25, 2014.

[137] B. Pinkas, T. Schneider, N. P. Smart, and S. C. Williams. Secure Two-Party Computation Is Practical. In M. Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture notes in Computer Science*, pages 250–267. Springer Berlin Heidelberg, 2009.

[138] M. Poulymenopoulou, F. Malamateniou, and G. Vassilacopoulos. Emergency Healthcare Process Automation Using Mobile Computing and Cloud Services. *Journal of Medical Systems*, 36(5):3233–3241, Oct. 2012.

[139] M. O. Rabin. How to exchange secrets with oblivious transfer. Technical Report TR-81, Aiken Computation Lab, Harvard University, 1981.

[140] S. Rass and D. Slamanig. *Cryptography for Security and Privacy in Cloud Computing*. Information Security and Privacy. Artech House, 2014.

[141] O. Regev. On Lattices, Learning With Errors, Random Linear Codes, and Cryptography. In *Proc. of the* 37^{th} *STOC*, pages 84–93. ACM, 2005.

[142] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On Data Banks and Privacy Homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.

[143] A. Rodrigues, J. S. Silva, and F. Boavida. iSenior – A Support System for Elderly Citizens. *IEEE Transactions on Emerging Topics in Computing*, 1(2):207–217, 2013.

[144] P. Rogaway. *The round complexity of secure protocols*. PhD thesis, MIT, 1991.

[145] R. Roman, P. Najera, and J. Lopez. Securing the Internet of Things. *Computer*, 44(9):51–58, Sept 2011.

[146] T. Sander, A. Young, and M. Yung. Non-interactive cryptocomputing for NC1. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 554–566, 1999.

[147] A. Saxena, B. Wyseur, and B. Preneel. Towards Security Notions for White-Box Cryptography. In P. Samarati, M. Yung, F. Martinelli, and C. A. Ardagna, editors, *Information Security*, volume 5735 of *Lecture notes in Computer Science*, pages 49–58. Springer Berlin Heidelberg, 2009.

[148] S. Schrittwieser and S. Katzenbeisser. Code Obfuscation against Static and Dynamic Reverse Engineering. In T. Filler, T. Pevý, S. Craver, and A. Ker, editors, *Information Hiding*, volume 6958 of *Lecture notes in Computer Science*, pages 270–284. Springer Berlin Heidelberg, 2011.

[149] A. Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, Oct. 1992.

[150] B. Shankar, K. Srinathan, and C. P. Rangan. Alternative Protocols for Generalized Oblivious Transfer. In S. Rao, M. Chatterjee, P. Jayanti, S. C. Murthy, and S. K. Saha, editors, *Distributed Computing and Networking*, volume 4904 of *Lecture notes in Computer Science*, pages 304–309. Springer Berlin Heidelberg, 2008.

[151] P. Snyder. Yaoś Garbled Circuits: Recent Directions and Implementations. Accessed 2015. Available at `www.cs.uic.edu\slash`.

[152] E. M. Songhori, S. U. Hussain, A. Sadeghi, T. Schneider, and F. Koushanfar. TinyGarble: Highly Compressed and Scalable Sequential Garbled Circuits. In *36th IEEE Symposium on Security and Privacy (Oakland)*, May 2015.

[153] B. D. Sutter, B. Anckaert, J. Geiregat, D. Chanet, and K. D. Bosschere. Instruction set limitation in support of software diversity. In P. J. Lee and H. J. Cheon, editors, *Proceedings of ICISC 2008*, volume 5461 of *Lecture notes in Computer Science*, pages 152–165. Springer Heidelberg, 2009.

[154] H. Takabi, J. B. D. Joshi, and G.-J. Ahn. Security and Privacy Challenges in Cloud Computing Environments. *Security & Privacy*, 8:24–31, 2010.

[155] T. Tassa. Generalized oblivious transfer by secret sharing. *Designs, Codes and Cryptography*, 58(1):11–21, 2011.

[156] Y. Tong, J. Sun, S. Chow, and P. Li. Cloud-Assisted Mobile-Access of Health Data With Privacy and Auditability. *IEEE journal of Biomedical and Health Informatics*, 18(2):419–429, March 2014.

[157] P. M. Trief, J. Sandberg, R. Izquierdo, P. C. Morin, S. Shea, R. Brittain, E. B. Feldhousen, and R. S. Weinstock. Diabetes Management Assisted by Telemedicine: Patient Perspectives. *Telemedicine and e-Health*, 14(7):647–655, 2008.

[158] P. C. van Oorschot. Revisiting Software Protection. In C. Boyd and W. Mao, editors, *Information Security*, volume 2851 of *Lecture notes in Computer Science*, pages 1–13. Springer Berlin Heidelberg, 2003.

[159] O. Vermesan, M. Harrison, H. Vogt, K. Kalaboukas, M. Tomasella, K. Wouters, S. Gusmeroli, and S. Haller. *Vision and Challenges for Realising the Internet of Things*. European Commission, Information Society and Media, 2010.

[160] C. Wang. *A security architecture for survivability mechanisms*. PhD thesis, University of Virginia, 2001.

[161] R. H. Weber. Internet of Things – New security and privacy challenges. *Computer Law & Security Review*, 26(1):23 – 30, 2010.

[162] H. Wee. On Obfuscating Point Functions. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 523–532. ACM, 2005.

[163] J. S. Winter. Surveillance in Ubiquitous Network Societies: Normative Conflicts Related to the Consumer In-store Supermarket Experience in the Context of the Internet of Things. *Ethics and Information Technology*, 16(1):27–41, 2014.

[164] A. Wood, J. A. Stankovic, G. Virone, L. Selavo, Z. He, Q. Cao, T. Doan, Y. Wu, L. Fang, and R. Stoleru. Context-aware wireless sensor networks for assisted living and residential monitoring. *Network, IEEE*, 22(4):26–33, 2008.

[165] B. Wyseur. *White-Box Cryptography*. PhD thesis, Katholieke Universiteit Leuven, March 2009.

[166] B. Wyseur. White-Box Cryptography: Hiding Keys in Software. in `http://www.whiteboxcrypto.com/`, 2012. Accessed 15th June 2015.

[167] B. Wyseur, W. Michiels, P. Gorissen, and B. Preneel. Cryptanalysis of White-Box DES Implementations with Arbitrary External Encodings. In C. Adams, A. Miri, and M. Wiener, editors, *Selected Areas in Cryptography*, volume 4876 of *Lecture notes in Computer Science*, pages 264–277. Springer Berlin Heidelberg, 2007.

[168] A. Yao. Protocols for Secure Computations. In *Proc. of* 23^{rd} *SFCS, 1982*, pages 160–164. IEEE, 1982.

[169] A. Yao. How to generate and exchange secrets. In *Proc. of* 27^{th} *FOCS, 1986.*, pages 162–167. IEEE, 1986.

[170] S. Zahur, M. Rosulek, and D. Evans. Two Halves Make a Whole. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, volume 9057 of *Lecture notes in Computer Science*, pages 220–250. Springer Berlin Heidelberg, 2015.

Part II

Original publications

I

Paper I

Classes of Garbling Schemes

T. Meskanen and V. Niemi and N. Nieminen (2013). *Infocommunications journal*, V(3):8-16

Classes of Garbling Schemes

Tommi Meskanen, Valtteri Niemi, Noora Nieminen

***Abstract*—Bellare, Hoang and Rogaway elevated *garbled circuits from a cryptographic technique to a cryptographic goal* by defining several new security notions for garbled circuits [3]. This paper continues at the same path by extending some of their results and providing new results about the classes of garbling schemes defined in [3]. Furthermore, new classes of garbling schemes are defined and some results concerning them and their relation to earlier classes are proven.**

***Index Terms*—garbled circuits, garbling schemes, secure multiparty computations, privacy**

I. INTRODUCTION

The history of garbled circuits traces back to A. Yao, who introduced the technique in [7]. The term *garbled circuit* was introduced by Beaver, Micali and Rogaway [2] where they introduced a way of performing secure multiparty computation with Yao's circuit garbling technique. Since then Yao's garbled circuits have been used for various purposes even though there was no formal definition what is meant by garbling. No proof of security existed either - until Lindell and Pinkas introduced one for a particular garbled circuit using a protocol assuming semi-honest adversaries [5], [6]. After this result, also a proof of security against covert and malicious adversaries has been published [1], [6]. Again, these results are obtained for a specific protocol using garbling schemes rather than considering the security of garbling itself.

The first formal definition of a garbling scheme has recently been proposed by Bellare, Hoang and Rogaway in [3]. A garbling scheme is defined as a five-tuple of functions: the actual garbling procedure $\mathtt{Gb}$, the encryption function $\mathtt{En}$, the decryption function $\mathtt{De}$, the garbled evaluation function $\mathtt{Ev}$ and the original evaluation function $\mathtt{ev}$. The idea behind garbling is the following. Let f be a function which is to be evaluated for different inputs x but in such a way that neither f nor x can be learnt from the evaluation process. Therefore, a garbled version F is created and instead of computing $y = \mathtt{ev}(f, x)$ we compute $Y = \mathtt{Ev}(F, X)$ where X is obtained from x by encryption. After this y is obtained from Y by decryption. Figure 1 illustrates the garbling procedure.

Rogaway et al. define also three security notions for garbling schemes. These notions are expressed via code-based games which are defined in such a way that they capture the intuition behind the different notions: privacy, obliviousness and authenticity are all defined to be reached, if the adversary has only a negligible advantage for winning a particular game. Moreover, these notions have two different models, either based on indistinguishability or simulation. Roughly speaking, indistinguishability means that the adversary cannot distinguish between garblings of two functions. The simulation type means that an adversary is incapable of distinguishing garbling of the function of its own choice from another similar looking function devised by a simulator. Here we refer to the next section for the formal definitions.

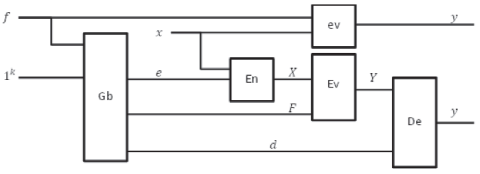


Figure 1: Description of the technique behind garbling. The diagram also illustrates that the result of evaluation with garbling must coincide with the result obtained without garbling.

Another seminal achievement in [3] is that relations between the different security notions have been proven. Rogaway et al. also provide two concrete garbling schemes, one of which achieves not only privacy but also obliviousness and authenticity. This example assures that the defined security classes are not empty.

This paper consists of three sections. In the first section we define all the necessary concepts, and give an informal description of them so that the idea behind the concept would be more comprehensive to the reader. In the second section we provide new results about the already known classes: some of the results are extensions to the results in [3], some inspired by the results in [3]. The third section provides modified definitions of the games used to define the different security notions. In this manner, we obtain new classes of garbling schemes by minor modifications in the games. Then, we prove some relations not only between the new and existing classes but also among the new classes. We also discuss intuition behind these new classes.

II. DEFINITIONS

In this section, we provide the basic definitions and notations. As usual, $\mathbb{N}$ will be the set of positive integers. A *string* is a finite sequence of bits. In addition to the basic strings, there is a special symbol $\perp$. The meaning of this symbol is explained later where the context of usage will be clearer.

Let A be a finite set. Notation $y \leftarrow A$ means that an element is selected uniformly at random from the set A, and this element is assigned to y. If A denotes an algorithm, then notation $A(x_1, \ldots, x_n)$ means the output of the algorithm A on inputs $x_1, \ldots, x_n$.

T. Meskanen is a researcher at the Department of Mathematics and Statistics, University of Turku, Finland (email: tommes@utu.fi).
V. Niemi is a professor at the Department of Mathematics and Statistics, University of Turku, Finland (email: pevani@utu.fi).
N. Nieminen is a doctoral student at Turku Centre for Computer Science, University of Turku, Finland (email: nmniem@utu.fi).

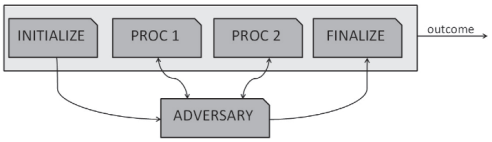


Figure 2: The idea of a code-based game is captured in the above image.

As usual, we say that a function $f : \mathbb{N} \to \mathbb{R}$ is negligible if for every $c > 0$ there is an integer N_c such that $|f(x)| < x^{-c}$ for all $x > N_c$

A. Code-based games

The proofs in this paper are heavily based on *code-based games*. Following the terminology presented in [4], a game is a collection of procedures called *oracles*. This collection may contain three types of procedures: `INITIALIZE`, `FINALIZE` and other named oracles. The word *may* is used, since all the procedures in a game are optional.

The entity playing a game is called *adversary*. When the game is run with an adversary, first the `INITIALIZE` procedure is called. It possibly provides an input to the adversary, who in turn may invoke other procedures before feeding its output to the `FINALIZE` procedure. The `FINALIZE` receives an output of the adversary, and creates a string that tells the outcome from the game typically consisting of one bit of information: whether the adversary has won or not. This description about code-based games is quite informal and gives only the intuition behind the concept. The Figure 2 serves as an illustration. For a more formal description, we refer to [4].

B. Garbling schemes

In this section we provide a formal definition of garbling schemes and their security, and here we follow the guidelines provided in [3].

Formally, a garbling scheme is a 5-tuple $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev})$ of algorithms, from which the first is probabilistic and the rest are deterministic. Let f denote the string that represents the original function. The last component in the 5-tuple is the evaluation function $\mathtt{ev}(f, \cdot) : \{0,1\}^n \to \{0,1\}^m$ which we want to garble. Here, the values $n = f.n$ and $m = f.m$ represent the lengths of the input x and the output $y = ev(f, x)$. They must also be efficiently computable from f. The first component `Gb` denotes the garbling algorithm. It takes f and 1^k as its inputs, where $k \in \mathbb{N}$ is a security parameter, and returns (F, e, d) on this input. String e describes the encryption algorithm $\mathtt{En}(e, \cdot)$ which maps an initial input x to a garbled input $X = \mathtt{En}(e, x)$. String F describes the garbled function $\mathtt{Ev}(F, \cdot)$. It returns the garbled output $Y = \mathtt{Ev}(F, X)$. Finally, string d describes the decryption algorithm $\mathtt{De}(d, \cdot)$ which on a garbled input returns the final output $y = \mathtt{De}(d, Y)$. Here we refer to Figure 1 to get an idea of how a garbling scheme works.

`NOTE:` Occasionally, we use a specific evaluation function $\mathtt{ev}_{circ}$ as `ev` in the 6-tuple. For it, we first define *a conventional circuit* by a 6-tuple $f = (n, m, q, A, B, G)$. The first component denotes the number of input wires $(n \geq 2)$, the second is the number of output wires $(m \geq 1)$, and the third component represents the number of gates $(q \geq 1)$ in the circuit. The function A identifies the first incoming wire, whereas B identifies the second incoming wire of each gate. The remaining component G is a function identifying the functionality of each gate. For a more specific definition of a circuit, see [3]. Finally, the circuit evaluation function $\mathtt{ev}_{circ}$ is the usual canonical evaluation function:

proc $\mathtt{ev}_{circ}(f, x)$
$(n, m, q, A, B, G) \leftarrow f$
for $g \leftarrow n+1$ **to** $n+q$ **do** $a \leftarrow A(g), b \leftarrow B(g), x_g \leftarrow G_g(x_a, x_b)$
return $x_{n+q-m+1} \cdots x_{n+q}$

There are some additional requirements that garbling schemes must fulfill. These are *length, non-degeneracy* and *correctness* conditions. The length condition means that the lengths of F, e, d may only depend on the security parameter k, the values $f.n, f.m$ and the length of the string f. Non-degeneracy condition means the following: if $f.n = g.n$, $f.m = g.m$, $|f| = |g|$, $(F, e, d) = \mathtt{Gb}(1^k, f; r)$ and $(G, e', d') = \mathtt{Gb}(1^k, g; r)$ where r represents random coins of `Gb`, then $e = e'$ and $d = d'$. Correctness requires that $\mathtt{De}(d, \mathtt{Ev}(F, \mathtt{En}(e, x)))$ will always give the same result as $\mathtt{ev}(f, x)$.

By the concept of *a side-information function*, we capture the information revealed about f by the garbling process. In the case of circuits and $\mathtt{ev}_{circ}$, this might be the size of the circuit that was garbled, the topology of it or something else - even the whole initial circuit. Formally, a side-information function Φ deterministically maps string f to string $\Phi(f)$. Let $f = (n, m, q, A, B, G)$ be a circuit. Then, we define $\Phi_{size}(f) = (n, m, q)$, which is the side-information function revealing the size of the garbled circuit. Other side-information functions are $\Phi_{circ}(f) = f$ which thus reveals the entire circuit, and Φ_{topo} which reveals the topology of the initial circuit, i.e. $\Phi_{topo} = (n, m, q, A, B)$.

C. The security notions of garbling schemes

There are three types of security: *privacy, obliviousness* and *authenticity*. The first two types also have two distinct models: one based on *indistinguishability* and another based on *simulation*. In all cases, the security is defined through a code-based game consisting of a procedure named `GARBLE` and finalization procedure `FINALIZE`. The procedure `GARBLE` is not to be confused with the garbling function `Gb` : the garbling function `Gb` is a component of a garbling scheme $\mathcal{G}$, whose security the adversary tries to break via the procedure `GARBLE`.

Before the game starts, the garbling scheme $\mathcal{G}$ and the side-information function Φ are fixed in the games based on indistinguishability model. In simulation model, also the simulator $\mathcal{S}$ is fixed although details of it are not assumed to be known to the adversary. The `GARBLE` procedure gives the challenge of the game to the adversary and the `FINALIZE`

procedure determines whether the adversary wins the game or not. The adversary is assigned a certain advantage depending on the probability of winning the game. This advantage in turn determines whether the garbling scheme is secure or not.

Table 1 gives the different GARBLE procedures needed in the games to define different security notions. Note that in this first formal description we use the subscripts, but after that, we omit them if they are clear from the context. For example, we will write *PrvSim* game instead of $PrvSim_{\mathcal{G},\Phi,\mathcal{S}}$.

Let $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev})$ be a garbling scheme, $k \in \mathbb{N}$ a security parameter and Φ a side-information function. The following definitions are informal, and they are mentioned to capture the idea behind the security notions. For a more formal treatment, see [3].

PRIVACY: Privacy has two types of notions, and hence there are two different games with distinct GARBLE procedures, $PrvInd_{\mathcal{G},\Phi}$ and $PrvSim_{\mathcal{G},\Phi,\mathcal{S}}$. The biggest difference between these two is that the latter requires an auxiliary algorithm to be defined, namely the simulator $\mathcal{S}$.

The game $PrvInd$ consists of a GARBLE procedure, which is called by the adversary *exactly once* during one game, and a FINALIZE procedure. Informally the game goes as follows: the adversary calls the GARBLE procedure having two appropriate functions and their inputs as the feed. The procedure returns a garbled version of one of the functions and its input, and the adversary guesses which of the functions got garbled. The FINALIZE procedure takes two inputs, value of parameter b from GARBLE and adversary's guess b', and tells whether the answer given by the adversary was correct or not, and this will then be the outcome of the game.

The game $PrvSim$ has also two procedures, its own GARBLE and FINALIZE, from which the latter has the same functionality as in $PrvInd$ game. The difference in GARBLE procedure is, that now the other function, from which the function f is to be distinguished, is devised by the simulator. The adversary must tell the difference between an actual function and a "fake" function.

We define the advantage of an adversary A in game $PrvInd$ as follows:

$$\mathbf{Adv}^{prv.ind,\Phi}_{\mathcal{G}}(\mathcal{A},k) = 2 \cdot \Pr\left[PrvInd^{\mathcal{A}}_{\mathcal{G},\Phi}(k)\right] - 1.$$

If the advantage function $\mathbf{Adv}^{prv.ind,\Phi}_{\mathcal{G}}(\mathcal{A},\cdot)$ is negligible for all PT adversaries $\mathcal{A}$ then we say that the garbling scheme $\mathcal{G}$ is *prv.ind secure over* Φ. Similarly, we define the advantage of an adversary $\mathcal{B}$ in game $PrvSim$ as $\mathbf{Adv}^{prv.sim,\Phi,\mathcal{S}}_{\mathcal{G}}(\mathcal{B},k) = 2 \cdot \Pr\left[PrvSim^{\mathcal{B}}_{\mathcal{G},\Phi,\mathcal{S}}(k)\right] - 1$. Then, we define that a garbling scheme $\mathcal{G}$ is *prv.sim secure over* Φ if for every PT adversary there exists a PT simulator $\mathcal{S}$ such that $\mathbf{Adv}^{prv.sim,\Phi,\mathcal{S}}_{\mathcal{G}}(\mathcal{B},k)$ is negligible.

OBLIVIOUSNESS: At first sight, the games for obliviousness seem similar to the privacy games. The difference is that the decryption algorithm d is not given to the adversary, and hence the adversary cannot compute the final output $y = \mathtt{De}(d, \mathtt{Ev}(F,X))$. Informally, the adversary is asked to distinguish two functions and their inputs from each other without knowing the result of evaluation.

The adversary has an advantage which is calculated as in the privacy model. The obv.ind and the obv.sim security of a garbling scheme $\mathcal{G}$ are defined similarly as in the corresponding Prv-games.

AUTHENTICITY: Here the FINALIZE procedure is a little more complex than in the two cases above. The finalization procedure of a game checks whether the adversary is able to produce a valid garbled output Y different to $\mathtt{Ev}(F,X)$ or not. Also the advantage function is slightly different: $\mathbf{Adv}^{aut}_{\mathcal{G}}(\mathcal{A},k) = \Pr\left[Aut^{\mathcal{A}}_{\mathcal{G}}(k)\right]$. Again, a garbling scheme is aut-secure, if for all polynomial time adversaries $\mathcal{A}$ the advantage function $\mathbf{Adv}^{aut}_{\mathcal{G}}(\mathcal{A},\cdot)$ is negligible.

We denote $\mathtt{GS}(xxx,\Phi)$ to be the set of all garbling schemes that are $xxx-$secure over the side-information function Φ, where xxx denotes the type of security: $prv.ind$, $prv.sim$, $obv.ind$, $obv.sim$, $mod.ind$, $mod.sim$, $mod.ind2$ or $mod.sim2$. The notion $\mathtt{GS}(aut)$ means the set of all $aut-$secure garbling schemes. $\mathtt{GS}(ev)$ means the class of garbling schemes which use the evaluation function ev.

III. Results about established classes of garbling schemes

In this section we provide results concerning the security classes *prv.ind, prv.sim, obv.ind, obv.sim* defined in section 2. The first two theorems consider the effect of different side-information functions to the sets of garbling schemes. The following two theorems provide extensions to the existing results in [3] – the non-inclusions are obtained for any side-information function Φ instead of restricting it to Φ_{topo}. Then we continue with two results that provide parallel results to [3]. Finally, the last two theorems in this section provide new results about the established security classes of garbling schemes.

Theorem 1: Suppose that two different side-information functions Φ_a and Φ_b satisfy the condition

$$\Phi_a(f_0) = \Phi_a(f_1) \Rightarrow \Phi_b(f_0) = \Phi_b(f_1). \quad \textbf{(Condition } (*)\textbf{)}$$

Then we have the inclusion $\mathtt{GS}(prv.ind,\Phi_b) \subseteq \mathtt{GS}(prv.ind,\Phi_a)$. If we additionally assume that there exists a polynomial time function g such that $g(\Phi_a(f)) = \Phi_b(f)$ then we also have $\mathtt{GS}(prv.sim,\Phi_b) \subseteq \mathtt{GS}(prv.sim,\Phi_a)$.

Proof: Let $\mathcal{G} = (\mathtt{Gb},\mathtt{En},\mathtt{De},\mathtt{Ev},\mathtt{ev}) \in \mathtt{GS}(prv.ind,\Phi_b)$. Suppose now that $\mathcal{A}$ is an arbitrary adversary playing the $PrvInd_{\Phi_a}$ game and let us construct $\mathcal{B}$ as an adversary playing the $PrvInd_{\Phi_b}$ game and using $\mathcal{A}$ as a subroutine. The latter adversary $\mathcal{B}$ tells the first adversary $\mathcal{A}$ to start the game. Adversary $\mathcal{A}$ chooses its input (f_0,f_1,x_0,x_1) which it wants to send to GARBLE procedure, which now in fact the adversary $\mathcal{B}$ pretends to be. Adversary $\mathcal{B}$ forwards the input from $\mathcal{A}$ to GARBLE procedure in $PrvInd_{\Phi_b}$ game. Adversary $\mathcal{B}$ receives an output (F,X,d) or $\perp$ from GARBLE. Now, if $\Phi_b(f_0) \neq \Phi_b(f_1)$, adversary $\mathcal{B}$ sends $\perp$ to $\mathcal{A}$. This is the normal answer: According to our assumption, $\Phi_b(f_0) \neq \Phi_b(f_1) \Rightarrow \Phi_a(f_0) \neq \Phi_a(f_1)$ and hence adversary $\mathcal{A}$ should receive $\perp$ also from its genuine GARBLE procedure. Otherwise, adversary $\mathcal{B}$ forwards the response from

proc $\texttt{GARBLE}(f_0, f_1, x_0, x_1)$ Game $\text{PrvInd}_{\mathcal{G},\Phi}$ $b \leftarrow \{0,1\}$ **if** $\Phi(f_0) \neq \Phi(f_1)$ **then return** $\perp$ **if** $\{x_0, x_1\} \not\subseteq \{0,1\}^{f_0.n}$ **then return** $\perp$ **if** $\texttt{ev}(f_0, x_0) \neq \texttt{ev}(f_1, x_1)$ **then return** $\perp$ $(F, e, d) \leftarrow \texttt{Gb}(1^k, f_b)$; $X \leftarrow \texttt{En}(e, x_b)$; **return** (F, X, d)	**proc** $\texttt{GARBLE}(f, x)$ Game $\text{PrvSim}_{\mathcal{G},\Phi,\mathcal{S}}$ $b \leftarrow \{0,1\}$ **if** $x \notin \{0,1\}^{f.n}$ **then return** $\perp$ **if** $b = 1$ **then** $(F, e, d) \leftarrow \texttt{Gb}(1^k, f)$; $X \leftarrow \texttt{En}(e, x)$ **else** $y \leftarrow \texttt{ev}(f, x)$; $(F, X, d) \leftarrow \mathcal{S}(1^k, y, \Phi(f))$ **return** (F, X, d)
proc $\texttt{GARBLE}(f_0, f_1, x_0, x_1)$ Game $\text{ObvInd}_{\mathcal{G},\Phi}$ $b \leftarrow \{0,1\}$; **if** $\Phi(f_0) \neq \Phi(f_1)$ **then return** $\perp$ **if** $\{x_0, x_1\} \not\subseteq \{0,1\}^{f_0.n}$ **then return** $\perp$ $(F, e, d) \leftarrow \texttt{Gb}(1^k, f_b)$; $X \leftarrow \texttt{En}(e, x_b)$; **return** (F, X)	**proc** $\texttt{GARBLE}(f, x)$ Game $\text{ObvSim}_{\mathcal{G},\Phi,\mathcal{S}}$ $b \leftarrow \{0,1\}$ **if** $x \notin \{0,1\}^{f.n}$ **then return** $\perp$ **if** $b = 1$ **then** $(F, e, d) \leftarrow \texttt{Gb}(1^k, f)$; $X \leftarrow \texttt{En}(e, x)$ **else** $(F, X) \leftarrow \mathcal{S}(1^k, \Phi(f))$ **return** (F, X)
proc $\texttt{FINALIZE}(b, b')$ Game $\text{PrvInd}_{\mathcal{G},\Phi}$, Game $\text{PrvSim}_{\mathcal{G},\Phi,\mathcal{S}}$, Game $\text{ObvInd}_{\mathcal{G},\Phi}$, Game $\text{ObvSim}_{\mathcal{G},\Phi,\mathcal{S}}$ **return** $b = b'$	
proc $\texttt{GARBLE}(f, x)$ Game $\text{Aut}_{\mathcal{G}}$ $(F, e, d) \leftarrow \texttt{Gb}(1^k, f)$; $x \leftarrow \texttt{En}(e, x)$ **return** (F, X)	**proc** $\texttt{FINALIZE}(Y)$ Game $\text{Aut}_{\mathcal{G}}$ **return** $\texttt{De}(d, Y) \neq \perp$ **and** $Y \neq \texttt{Ev}(F, X)$

Table I: The games defining the different security notions

its GARBLE to $\mathcal{A}$, who then sends its answer b' to $\mathcal{B}$. The adversary $\mathcal{B}$ answers the same b' in its $PrvInd_{\Phi_b}$ game.

Let us now consider the winning probabilities and advantages of both adversaries in their games. The behavior of adversaries $\mathcal{A}$ and $\mathcal{B}$ are the same at every step of the game: the inputs are the same, and the answers are the same. Therefore the probability of the answer b' being the correct one must be the same in both games. Hence the advantages of both adversaries are also equal. Because $\mathcal{G} \in \texttt{GS}(prv.ind, \Phi_b)$ the advantage of $\mathcal{B}$ in $PrvInd_{\Phi_b}$ game is negligible. Thus, the advantage of $\mathcal{A}$ is also negligible, and $\mathcal{G} \in \texttt{GS}(prv.ind, \Phi_a)$, which proves the claim.

For the second part, let us assume that there exists an efficient conversion g from the side-information function Φ_a into Φ_b. Our objective is to prove under these assumptions that $\texttt{GS}(prv.sim, \Phi_b) \subseteq \texttt{GS}(prv.sim, \Phi_a)$.

To do this, assume that $\mathcal{G} \in \texttt{GS}(prv.sim, \Phi_b)$. This means that for every polynomial time adversary $\mathcal{A}'$ there exists a simulator $\mathcal{S}$ such that the advantage of $\mathcal{A}'$ is negligible in $PrvSim_{\mathcal{G},\Phi_b,\mathcal{S}}$ game.

Let $\mathcal{A}$ be an arbitrary adversary playing $PrvSim_{\mathcal{G},\Phi_a,\mathcal{S}}$ games. Similarly to the first part of the proof, let $\mathcal{B}$ be an adversary who plays $PrvSim_{\mathcal{G},\Phi_b,\mathcal{S}}$ games by emulating $\mathcal{A}$, i.e. behaving just like $\mathcal{A}$ would behave in corresponding $PrvSim_{\mathcal{G},\Phi_a,\mathcal{S}}$ games. More precisely, by emulation of $\mathcal{A}$ we mean the following. First, adversary $\mathcal{B}$ tells $\mathcal{A}$ to start its game. Adversary $\mathcal{B}$ receives the GARBLE input (f, x) from $\mathcal{A}$, after which $\mathcal{B}$ forwards this input to its own GARBLE. This procedure returns (F, X, d) or $\perp$ to $\mathcal{B}$, who now consults adversary $\mathcal{A}$ by giving this output to him. Now, $\mathcal{A}$ returns b' to $\mathcal{B}$, who chooses the same b' as its own return value.

The assumption $\mathcal{G} \in \texttt{GS}(prv.sim, \Phi_b)$ implies that there exists a simulator $\mathcal{S}_{hard}$ such that the advantage of $\mathcal{B}$ is negligible in $PrvSim_{\mathcal{G},\Phi_b,\mathcal{S}_{hard}}$ game. Now, we define another simulator $\mathcal{S}'_{hard}$ by $\mathcal{S}'_{hard}(1^k, y, \Phi_a(f)) = \mathcal{S}_{hard}(1^k, y, g(\Phi_a(f)))$. First of all, $\mathcal{S}'_{hard}$ is polynomial time, because the conversion g is efficient and $\mathcal{S}_{hard}$ is a polynomial time simulator. Secondly, the win probability of $\mathcal{B}$ in its own $PrvSim_{\mathcal{G},\Phi_b,\mathcal{S}_{hard}}$ game is the same as the win probability that $\mathcal{A}$ has in the $PrvSim_{\mathcal{G},\Phi_a,\mathcal{S}'_{hard}}$ game, which implies equal advantages. By assumption, the advantage of $\mathcal{B}$ was negligible, and so is the advantage of $\mathcal{A}$ by the above argument. Now we have found a simulator against which $\mathcal{A}$ has a negligible advantage. □

NOTE: For example, $\Phi_a = \Phi_{topo}$ and $\Phi_b = \Phi_{size}$ satisfy the condition $(*)$.

Theorem 2: Let Φ_a and Φ_b be two different side-information functions satisfying the above condition $(*)$. Then the following inclusion holds: $\texttt{GS}(obv.ind, \Phi_b) \subseteq \texttt{GS}(obv.ind, \Phi_a)$. If we additionally assume that there exists a polynomial time function g such that $g(\Phi_a(f)) = \Phi_b(f)$ then we have also $\texttt{GS}(obv.sim, \Phi_b) \subseteq \texttt{GS}(obv.sim, \Phi_a)$.

Proof: The proof is similar to that of the previous theorem. □

The next four theorems consider non-inclusions of the form $A \not\subseteq B$ between sets of garbling schemes. In all cases we make an assumption that the set A is non-empty. The following two propositions provide a generalization to Propositions 5 and 7 in paper [3].

Theorem 3: For all Φ and for $\texttt{ev} = \texttt{ev}_{circ}$, we have $\texttt{GS}(obv.sim, \Phi) \bigcap \texttt{GS}(\texttt{ev}) \not\subseteq \texttt{GS}(prv.ind, \Phi)$.

Proof: Let $\mathcal{G} = (\texttt{Gb}, \texttt{En}, \texttt{De}, \texttt{Ev}, \texttt{ev}) \in \texttt{GS}(obv.sim, \Phi) \bigcap \texttt{GS}(\texttt{ev})$. Let us construct another garbling scheme $\mathcal{G}' = (\texttt{Gb}', \texttt{En}, \texttt{De}', \texttt{Ev}, \texttt{ev})$ such that $\mathcal{G}' \in \texttt{GS}(obv.sim, \Phi) \bigcap \texttt{GS}(\texttt{ev})$ but $\mathcal{G}' \notin \texttt{GS}(prv.ind, \Phi)$. The construction is as follows: The function $\texttt{Gb}'(1^k, f)$ picks $(F, e, d) \leftarrow \texttt{Gb}(1^k, f)$ and returns $(F, e, d||e)$. Let $\texttt{De}'(d||e, Y) = \texttt{De}(d, Y)$. Including e in the description of the decoding function does not harm $obv.sim$ security, because the adversary is given only (F, X) by the GARBLE procedure in the $obv.sim$ game. Thus $\mathcal{G}'$ inherits the $obv.sim$ security from $\mathcal{G}$.

On the other hand, $\mathcal{G}'$ is not $prv.ind$ secure. Adversary $\mathcal{A}$ makes a query (f_0, f_1, x_0, x_1), where $f_0 = f_1 = \texttt{AND}$ and $x_0 = 00, x_1 = 01$. This choice is fine for the PrvInd game, since $\texttt{ev}(f_0, x_0) = 0 = \texttt{ev}(f_1, x_1)$. Now, the adversary computes $X_0 = \texttt{En}(e, x_0)$ and $X_1 = \texttt{En}(e, x_1)$, which must be different because of the non-degeneracy condition (see Section 2). Then he/she compares these two with the garbled

input X received from GARBLE. This comparison now reveals which of the inputs, x_0 or x_1, was used. □

Theorem 4: For all Φ and for $\mathtt{ev} = \mathtt{ev}_{circ}$, we have $\mathtt{GS}(aut) \bigcap \mathtt{GS}(\mathtt{ev}) \nsubseteq \mathtt{GS}(prv.ind, \Phi) \bigcup \mathtt{GS}(obv.ind, \Phi)$.

Proof: Let $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev}) \in \mathtt{GS}(aut) \bigcap \mathtt{GS}(\mathtt{ev})$. Let us construct a garbling scheme $\mathcal{G}' = (\mathtt{Gb}, \mathtt{En}', \mathtt{De}, \mathtt{Ev}', \mathtt{ev})$ such that $\mathcal{G}' \in \mathtt{GS}(aut, \Phi) \bigcap \mathtt{GS}(\mathtt{ev})$ but $\mathcal{G}' \notin \mathtt{GS}(prv.ind, \Phi) \bigcup \mathtt{GS}(obv.ind, \Phi)$. The construction is as follows: We define that $\mathtt{Ev}'(F, X||x)) = \mathtt{Ev}(F, X)$, $\mathtt{En}'(e, x) = \mathtt{En}(e, x)||x = X||x$.

The new encoding function $\mathtt{En}'$ and evaluation function $\mathtt{Ev}'$ do not harm $aut-$security, since the adversary has chosen the function f and its input x. On the other hand, appending x to the encoding harms both obliviousness and privacy: In both games the adversary chooses the function f in such a way that $ev(f, \cdot)$ is not injective. This is possible because it is assumed that $\mathtt{ev} = \mathtt{ev}_{circ}$.

In both PrvInd and ObvInd game the adversary chooses inputs x_0, x_1 such that $x_0 \neq x_1$ and $\mathtt{ev}(f, x_0) = \mathtt{ev}(f, x_1)$. Now, the encoding $X||x_b$ reveals which of the inputs was used. □

The following two results provide parallel results compared to Propositions 8 and 9 in [3].

Theorem 5: Let P be a one-way permutation in the set of all functions f. Then, for $\Phi_P(f) = P(f)$ and for any ev, $\mathtt{GS}(obv.ind, \Phi_P) \bigcap \mathtt{GS}(\mathtt{ev}) \nsubseteq \mathtt{GS}(obv.sim, \Phi_P)$.

Proof: Let $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev}) \in \mathtt{GS}(obv.ind, \Phi_P) \bigcap \mathtt{GS}(\mathtt{ev})$. We construct a new garbling scheme $\mathcal{G}' = (\mathtt{Gb}', \mathtt{En}, \mathtt{De}, \mathtt{Ev}', \mathtt{ev})$ such that $\mathcal{G}' \in \mathtt{GS}(obv.ind, \Phi_P) \bigcap \mathtt{GS}(\mathtt{ev})$ but $\mathcal{G}' \notin \mathtt{GS}(obv.sim, \Phi_P)$.

The construction is the following. The algorithm $\mathtt{Gb}'(1^k, f)$ picks $(F, e, d) \leftarrow \mathtt{Gb}(1^k, f)$ and returns $(F||f, e, d)$. Let $\mathtt{Ev}'(F||f, X)$ return $\mathtt{Ev}(F, X)$. First of all, we claim that the constructed garbling scheme is $obv.ind$ secure over Φ_P. The reasoning goes as follows. The adversary $\mathcal{A}$ sends (f_0, f_1, x_0, x_1) to its GARBLE. For the answer not being $\perp$ it must be that $\Phi_P(f_0) = \Phi_P(f_1)$, and hence $P(f_0) = P(f_1)$ by the definition of Φ_P. Since P is a one-way permutation, $f_0 = f_1$ must hold. Thus prepending f to the description of F does not harm $obv.ind$ security.

However, $\mathcal{G}'$ is not $obv.sim$ secure over Φ_P. We introduce an adversary $\mathcal{B}$ that breaks the $obv.sim$ security with respect to any PT simulator. The adversary chooses (f, x) to be sent to the GARBLE procedure in ObvSim game. Now, if the challenge bit b in the game is 0, the simulator $\mathcal{S}$ is called to produce $(F||f, X)$ from $(1^k, \Phi_P(f))$. However, the PT simulator manages to produce exactly the right function f with negligible probability, because $\Phi_P = P$ is a one-way permutation. In other words, this means that the adversary $\mathcal{B}$ will almost always detect from the parameter $F||f$ whether the simulator was used or not. □

Theorem 6: Let P be a one-way permutation in the set of all functions f and let $\Phi_P(f) = P(f)$ while ev is arbitrary. Assume that there exist x and y for which $\Phi_P(f) = P(f)$ is one-way even when restricted to functions f such that $y = \mathtt{ev}(f, x)$. Then $\mathtt{GS}(prv.ind, \Phi_P) \bigcap \mathtt{GS}(\mathtt{ev}) \nsubseteq \mathtt{GS}(prv.sim, \Phi_P)$.

Proof: Let $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev}) \in \mathtt{GS}(prv.ind, \Phi_P) \bigcap \mathtt{GS}(\mathtt{ev})$. We construct a new garbling scheme $\mathcal{G}' = (\mathtt{Gb}', \mathtt{En}, \mathtt{De}, \mathtt{Ev}', \mathtt{ev})$ such that $\mathcal{G}' \in \mathtt{GS}(prv.ind, \Phi_P) \bigcap \mathtt{GS}(\mathtt{ev})$ but $\mathcal{G}' \notin \mathtt{GS}(prv.sim, \Phi_P)$.

The construction is similar to that of the previous proof. The algorithm $\mathtt{Gb}'(1^k, f)$ picks $(F, e, d) \leftarrow \mathtt{Gb}(1^k, f)$ and returns $(F||f, e, d)$. Let $\mathtt{Ev}'(F||f, X)$ return $\mathtt{Ev}(F, X)$. First of all, the constructed garbling scheme is $prv.ind$ secure over Φ_P by exactly the same reasoning as in the previous proof.

However, $\mathcal{G}'$ is not $prv.sim$ secure over Φ_P. We prove this by introducing an adversary $\mathcal{B}$ having a non-negligible advantage in the $PrvSim_{\Phi_P}$ game. By the assumption, there exist x and y such that $\Phi_P(f)$ is still one-way, when restricted to f such that $y = \mathtt{ev}(f, x)$. Thus the adversary $\mathcal{B}$ can choose (f, x) satisfying $y = \mathtt{ev}(f, x)$ to be sent to the GARBLE procedure. Now, if the challenge bit b in the game is 0, the simulator $\mathcal{S}$ is called to produce $(F||f, X, d)$ from $(1^k, y, \Phi_P(f))$, where $y = \mathtt{ev}(f, x)$. However, the polynomial time simulator manages to produce exactly the right function f with negligible probability, because $\Phi_P = P$ is an injective one-way function. In other words, this means that the adversary $\mathcal{B}$ will almost always detect from $F||f$ whether the simulator was used or not. □

The following two propositions provide new results for garbling scheme classes in [3].

Theorem 7: If the function $h : (f, x) \mapsto (\Phi(f), \mathtt{ev}(f, x))$ is injective, then $\mathtt{GS}(\mathtt{ev}) \subseteq \mathtt{GS}(prv.ind, \Phi)$.

Proof: Let $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev})$ be an arbitrary garbling scheme over side-information function Φ. Let $\mathcal{B}$ be an adversary playing the $PrvInd_{\Phi}$ game. The adversary sends (f_0, f_1, x_0, x_1) to the GARBLE procedure of this game. For the output not being $\perp$ it must be that

$$\Phi(f_0) = \Phi(f_1), \mathtt{ev}(f_0, x_0) = \mathtt{ev}(f_1, x_1).$$

But by injectivity of h this implies

$$\begin{aligned} h(f_0, x_0) &= (\Phi(f_0), \mathtt{ev}(f_0, x_0)) \\ &= (\Phi(f_1), \mathtt{ev}(f_1, x_1)) = h(f_1, x_1) \\ &\Rightarrow (f_0, x_0) = (f_1, x_1). \end{aligned}$$

This in turn is equivalent to $f_0 = f_1$ and $x_0 = x_1$, meaning that the advantage of the adversary $\mathcal{B}$ in this game will be equal to 0. This completes the proof. □

Theorem 8: If the function ev is injective and efficiently invertible (i.e. given $y = \mathtt{ev}(f', x')$, f and x such that $\mathtt{ev}(f, x) = y$ can be found in polynomial time), then $\mathtt{GS}(\mathtt{ev}) \subseteq \mathtt{GS}(prv.sim, \Phi)$.

Proof: Let $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev})$ be an arbitrary garbling scheme over side-information function Φ. Let $\mathcal{B}$ be an adversary playing the $PrvSim_{\Phi}$ game. The adversary sends (f, x) to the GARBLE procedure of this game. But now, if the challenge bit $b = 0$, the simulator can always find the right f and x to be garbled because $y = \mathtt{ev}(f, x)$ can

be inverted efficiently and $\mathtt{ev}(f,x) = \mathtt{ev}(f',x')$ guarantees $f = f', x = x'$. This means that no matter what the challenge bit was, in both cases, $b = 0$ or $b = 1$, the pair (f, x) becomes garbled correctly because the simulator that knows f and x is able to use the normal garbling method. This means that the advantage of the adversary $\mathcal{B}$ in this game equals 0, proving the inclusion $\mathtt{GS}(\mathtt{ev}) \subseteq \mathtt{GS}(prv.sim, \Phi) \bigcap \mathtt{GS}(\mathtt{ev})$. □

IV. New classes of garbling schemes

In [3], the definitions and relations between different security types were, at least to some extent, based on intuition about what is meant by a garbling scheme that achieves privacy, obliviousness or authenticity, and the intuition was modeled as a game. In this section we consider the games defined in paper [3] from another point of view; we consider them purely as games, and try to achieve new results by modifying the existing game definitions in certain ways explained later.

The first modification we make is that in the indistinguishability model, the PrvInd game will be modified to the direction of ObvInd game by removing the decryption key d from the return value (F, X, d). The same end result can be obtained by tightening the ObvInd game by adding the evaluation test $\mathtt{ev}(f_0, x_0) \stackrel{?}{=} \mathtt{ev}(f_1, x_1)$ in it. In the absence of a better name we call the new class *ModInd*. The second modification concerns the PrvSim game, in which we again ease the requirements by removing the decryption key d from the return value (F, X, d). In ObvSim game, adding y to the input of the simulator $\mathcal{S}$ will lead to the same intermediate form as above. The new class shall be named *ModSim*.

Another modification is obtained by relaxing the PrvInd game by removing the evaluation test. This can also be achieved by adding d to the output (F, X) in ObvInd game. A similar modification in Sim side is to leave y out from the input of the simulator in PrvSim game, or add d to (F, X) in ObvSim game. The former modification is called *ModInd2* and the latter is called *ModSim2*.

The finalization procedure is not modified in any of these games.

A. Applications

Before proceeding to the descriptions of our modifications, it is convenient to discuss the possible applications that could utilize garbling schemes and more specifically, our modified security models. One typical example is outsourcing of a complex computation to a service in the cloud. In many cases the input data or the algorithm (or both of them) is privacy-sensitive data and should not be revealed to the party running the cloud service. With garbling schemes achieving different types of security, we can hide different amount of this information. In order to have an idea which type of security is most appropriate in different situations, let us take a closer look at which kind of information is revealed by a garbling scheme belonging to a specific security class.

Let the function f represent the algorithm, x represents the privacy-sensitive input data and $f(x) = y$ represents the output of the algorithm. These are all garbled with some garbling scheme, and the garbled function and garbled input are given to the server, which computes the garbled output. It depends on the garbling scheme how much the server is allowed to know about f, x and $f(x)$. It is worth noting that whatever the model of security is, the original function f is not known for the server, only the side-information function $\Phi(f)$ is. The following list provides the central differences between the models.

- **obv.sim**: Garbling does not reveal x, f or $f(x)$ to the server.
- **prv.sim**: The server is allowed to get $f(x)$ but not x or f.
- **mod.sim**: When computing $y_1 = f(x_1)$ and $y_2 = f(x_2)$, the server is allowed to find out whether $y_1 = y_2$ or not.

There are situations in which the output data is not sensitive and can be revealed to the party maintaining the cloud service. According to the list above, a prv.sim secure garbling scheme is then appropriate. Also garbling schemes of the two other types may fit the situation except if the server needs the output in further computations. The issue is that the output will remain garbled in the cloud. Of course, further computations could also be garbled but this arrangement would significantly and unnecessarily add the total complexity of computation.

If the output is sensitive data, an obv.sim secure scheme suits. Our modified model mod.sim is suitable as well except in some cases where the number and/or distribution of different output values may reveal too much information. On the other hand, mod.sim can actually be modified to apply to these cases as well. Instead of considering inputs x, the modified scheme would take inputs $x||i$ where i is for example an ever-increasing counter. The procedure then returns $\mathtt{ev}(f, x)||i$ as the output. The counter at the end of the evaluation result will now make sure that each output appears only once. According to the previous discussion, mod.sim secure garbling schemes can be used in the same applications as prv.sim or obv.sim secure schemes. In the following section we will prove that it is at least as easy to find a mod.sim secure scheme as it is to find an obv.sim or a prv.sim secure scheme. In conclusion, the modified security model mod.sim covers almost all applications except some esoteric cases.

B. Definitions and results

Next we give the formal definition of ModInd and ModSim games. Then we continue by proving some results concerning the new classes of garbling schemes that are secure with respect to these games.

The following proposition shows that mod.ind security is at least as easy to reach as prv.ind security or obv.ind security.

Theorem 9: $\mathtt{GS}(prv.ind, \Phi) \bigcup \mathtt{GS}(obv.ind, \Phi) \subseteq \mathtt{GS}(mod.ind, \Phi)$.

Proof: First suppose that $\mathcal{G}$ is a prv.ind secure garbling scheme. Dropping the decryption key d out of the output of GARBLE procedure does not increase the winning chances of any adversary.

proc $\texttt{GARBLE}(f_0, f_1, x_0, x_1)$ $\quad \text{ModInd}_{\mathcal{G},\Phi}$	**proc** $\texttt{GARBLE}(f, x)$ $\quad \text{ModSim}_{\mathcal{G},\Phi,\mathcal{S}}$
$b \leftarrow \{0,1\}$	$b \leftarrow \{0,1\}$
if $\Phi(f_0) \neq \Phi(f_1)$ **then return** $\perp$	**if** $x \notin \{0,1\}^{f.n}$ **then return** $\perp$
if $\{x_0, x_1\} \not\subset \{0,1\}^{f_0.n}$ **then return** $\perp$	**if** $b = 1$ **then** $(F, e, d) \leftarrow \texttt{Gb}(1^k, f); X \leftarrow \texttt{En}(e, x)$
if $\texttt{ev}(f_0, x_0) \neq \texttt{ev}(f_1, x_1)$ **then return** $\perp$	**else** $y \leftarrow \texttt{ev}(f, x)$; $(F, X) \leftarrow \mathcal{S}(1^k, y, \Phi(f))$
$(F, e, d) \leftarrow \texttt{Gb}(1^k, f_b)$; $X \leftarrow \texttt{En}(e, x_b)$	**return** (F, X)
return (F, X)	

Table II: The modified GARBLE procedures in Ind and Sim games

Secondly, suppose that $\mathcal{G}$ is an obv.ind secure scheme. Now, following the specification of ModInd GARBLE procedure, the adversary receives $\perp$ on all inputs whose evaluations $\texttt{ev}(f_0, x_0)$ and $\texttt{ev}(f_1, x_1)$ are not equal. However, this evaluation equality test is not a part of ObvInd game. Hence, even though GARBLE procedure in ObvInd game returns an output different from $\perp$, the corresponding procedure in ModInd game might return $\perp$. Otherwise the games are identical. Adversaries of both games are able to find out beforehand whether the GARBLE procedure returns $\perp$ and therefore the adversary in ModInd game does not receive . Therefore, the advantage of adversary playing the ModInd game cannot be better than the advantage of a corresponding adversary in ObvInd game. According to the assumption, the advantage in the ObvInd game is negligible, and thus the advantage in ModInd game must also be negligible. □

Theorem 10: $\texttt{GS}(prv.sim, \Phi) \bigcup \texttt{GS}(obv.sim, \Phi) \subseteq \texttt{GS}(mod.sim, \Phi)$.

Proof: First, suppose that garbling scheme $\mathcal{G}$ is prv.sim secure. As in the PrvInd case, omitting the decryption key d from (F, X, d) does not increase the winning probability of an adversary playing the modified game.

Secondly, suppose that the garbling scheme $\mathcal{G}$ belongs to the set of ObvSim secure schemes. In the ModSim game, the simulator's additional input y cannot make its work of producing a good output (F, X) more difficult. Let us explain in more details why this is the case.

Let $\mathcal{A}$ be an arbitrary adversary playing the ModSim game. Let $\mathcal{A}'$ be the corresponding adversary playing the ObvSim game: adversary $\mathcal{A}'$ behaves in ObvSim game exactly in the same way as $\mathcal{A}$ behaves in ModSim game. According to our assumption, there is a simulator $\mathcal{S}'$ such that the advantage of $\mathcal{A}'$ is negligible. Now, we construct a simulator $\mathcal{S}$ for the ModSim game. The simulator $\mathcal{S}$ will totally omit the additional input y and call simulator $\mathcal{S}'$ to produce an output to the adversary $\mathcal{A}$. Now, this simulator makes the advantage of adversary $\mathcal{A}$ negligible, because the adversary $\mathcal{A}$ behaves just like $\mathcal{A}'$ and also the simulators in both games behave identically. This completes the proof. □

As mentioned in the introductory part of this section, we have created four modifications to the prv.ind and prv.sim models in total, of which we have now covered two. In the rest of this section, we first give the descriptions of the two other modified games and provide some results concerning them. Finally, we give a diagram including the new models and their relations.

After these two definitions, we will now provide a result about mod.ind2 and mod.sim2.

Theorem 11: Assume that the following condition holds:

$$(\forall f_0, f_1)\ (\forall x_0, x_1): \qquad \textbf{(Condition } (**)\textbf{)}$$
$$\Phi(f_0) = \Phi(f_1) \Rightarrow \texttt{ev}(f_0, x_0) = \texttt{ev}(f_1, x_1).$$

Then $\texttt{GS}(mod.ind2, \Phi) = \texttt{GS}(prv.ind, \Phi)$. Otherwise $\texttt{GS}(mod.ind2, \Phi) = \emptyset$.

Proof: Suppose first that $(**)$ does not hold. Then the adversary can choose f_0, f_1, x_0, x_1 such that $\texttt{ev}(f_0, x_0) \neq \texttt{ev}(f_1, x_1)$ but still $\Phi(f_0) = \Phi(f_1)$ holds. In this case, the adversary will always win the game, because $b = 0$ if and only if $\texttt{ev}(f_0, x_0) = \texttt{De}(d, \texttt{Ev}(F, X))$, and thus the advantage would not satisfy the negligibility condition, and no garbling scheme is secure.

Now assume that $(**)$ holds. Then the the adversary in the ModInd2 game has no choice other than choosing f_0, f_1, x_0, x_1 such that $\Phi(f_0) = \Phi(f_1)$ for not receiving $\perp$ which now implies that $\texttt{ev}(f_0, x_0) = \texttt{ev}(f_1, x_1)$ must hold. It follows that the sets of PrvInd secure garbling schemes and ModInd2 secure garbling schemes must be equal. This completes the proof. □

Theorem 12: For any Φ, the inclusion $\texttt{GS}(mod.sim2, \Phi) \subseteq \texttt{GS}(prv.sim, \Phi)$ holds. If $(**)$ holds, and Φ is efficiently invertible (i.e. given $\phi = \Phi(f')$, a function f can be found in polynomial time such that $\Phi(f) = \phi$), then the equality $\texttt{GS}(mod.sim2, \Phi) = \texttt{GS}(prv.sim, \Phi)$ holds. Finally, if $(**)$ does not hold, then $\texttt{GS}(mod.sim2, \Phi) = \emptyset$.

Proof: The difference between the ModSim2 and PrvSim games is, that in PrvSim game the simulator gets $y = \texttt{ev}(f, x)$ as input, whereas the simulator in ModSim2 game does not. This means that simulator's task of creating a good output (F, X, d) in PrvSim game is not harder than the task of the simulator in the other game. Therefore, the advantage of an adversary in PrvSim game cannot be better than in ModSim2 game. This proves the first claim.

For the second part, suppose that $(**)$ holds and Φ is efficiently invertible. Even though y is not provided to the simulator, it still is able to produce (F', X', d') such that the adversary has no better chances than guessing to win the ModSim2 game. Namely, the simulator creates from $\Phi(f)$ such a function f' that $\Phi(f) = \Phi(f')$, and it then creates any suitable input x' to the function f'. Now, because of the condition $(**)$, the equality $\texttt{ev}(f, x) = \texttt{ev}(f', x')$ must hold and hence the simulator always learns the right y. This means that the setting in this new, modified game actually is exactly the same as in PrvSim game.

Finally suppose that $(**)$ does not hold. In the modified game, the adversary can choose f and x such that there

proc $\texttt{GARBLE}(f_0, f_1, x_0, x_1)$ $\quad$ ModInd2$_{\mathcal{G},\Phi}$	**proc** $\texttt{GARBLE}(f, x)$ $\quad$ ModSim2$_{\mathcal{G},\Phi,\mathcal{S}}$
$b \leftarrow \{0,1\}$	$b \leftarrow \{0,1\}$
if $\Phi(f_0) \neq \Phi(f_1)$ **then return** $\perp$	**if** $x \notin \{0,1\}^{f.n}$ **then return** $\perp$
if $\{x_0, x_1\} \not\subset \{0,1\}^{f_0.n}$ **then return** $\perp$	**if** $b = 1$ **then** $(F, e, d) \leftarrow \texttt{Gb}(1^k, f); X \leftarrow \texttt{En}(e, x)$
$(F, e, d) \leftarrow \texttt{Gb}(1^k, f_b); \quad X \leftarrow \texttt{En}(e, x_b)$	**else**$(F, X, d) \leftarrow \mathcal{S}(1^k, \Phi(f))$
return (F, X, d)	**return** (F, X, d)

Table III: Another modification of GARBLE procedure in Ind and Sim games

exists a function f' satisfying $f' \neq f$, $\Phi(f) = \Phi(f')$ and $\texttt{ev}(f, x) \neq \texttt{ev}(f', x')$ for some x'. Now the simulator has at most 50% chance to guess the correct f. If the guess was incorrect, distinguishing the simulated version from the actual garbled output is easy since the adversary is able to check if $\texttt{ev}(f, x) = \texttt{De}(d, \texttt{Ev}(F, X))$. Thus, no garbling scheme is ModSim2 secure. □

Corollary 1: The following inclusion holds: $\texttt{GS}(mod.sim2, \Phi) \subseteq \texttt{GS}(mod.ind2, \Phi)$.

Proof: The claim follows from Theorem 11 and Theorem 12 and Proposition 2 in [3]. □

NOTE: In practice, condition $(**)$ does not usually hold. Therefore, it is hard to imagine an application in which our second modification would have practical significance because of the above result.

The next theorem provides a relation between the modified simulation type and the modified indistinguishability type garbling schemes under our first modification.

Theorem 13: The following inclusion holds: $\texttt{GS}(mod.sim, \Phi) \subseteq \texttt{GS}(mod.ind, \Phi)$.

Proof: Let $\mathcal{G} = (\texttt{Gb}, \texttt{En}, \texttt{De}, \texttt{Ev}, \texttt{ev}) \in \texttt{GS}(mod.sim, \Phi)$. We need to prove that $\mathcal{G} \in \texttt{GS}(mod.ind, \Phi)$. Let $\mathcal{A}$ be the PT adversary playing the ModInd game. We construct a PT ModSim adversary $\mathcal{B}$ as follows. Let $\mathcal{B}$ run $\mathcal{A}$ as a subroutine. The latter makes its query f_0, f_1, x_0, x_1. Adversary $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$ if $\Phi(f_0) \neq \Phi(f_1)$ or $\{x_0, x_1\} \not\subseteq \{0,1\}^{f_0.n}$ or $\texttt{ev}(f_0, x_0) \neq \texttt{ev}(f_1, x_1)$.

Regardless of whether $\mathcal{B}$ returned $\perp$ to $\mathcal{A}$ or not, adversary $\mathcal{B}$ picks $c \in \{0,1\}$ at random and makes its query to GARBLE with input f_c, x_c getting back (F, X) which is sent to adversary $\mathcal{A}$ in case $\perp$ was not sent earlier. In any case, adversary $\mathcal{A}$ returns a bit b' to adversary $\mathcal{B}$. The latter adversary now returns 1 if $\Phi(f_0) = \Phi(f_1)$, $\{x_0, x_1\} \subseteq \{0,1\}^{f_0.n}$, $\texttt{ev}(f_0, x_0) = \texttt{ev}(f_1, x_1)$ and $b' = c$ and 0 otherwise. Let $\mathcal{S}$ be any PT algorithm representing the simulator. Then there are two possible outcomes of the game:

1) If $\Phi(f_0) = \Phi(f_1)$, $\{x_0, x_1\} \subseteq \{0,1\}^{f_0.n}$ and $\texttt{ev}(f_0, x_0) = \texttt{ev}(f_1, x_1)$, then the input to the simulator $\mathcal{S}$ is the same regardless of c, or
2) $\Phi(f_0) \neq \Phi(f_1)$, $\{x_0, x_1\} \not\subset \{0,1\}^{f_0.n}$ or $\texttt{ev}(f_0, x_0) \neq \texttt{ev}(f_1, x_1)$ then adversary $\mathcal{B}$ always answers 0 regardless of b' received from adversary $\mathcal{A}$.

Let's analyze the win probabilities of both adversaries. First consider the case 2. Adversary $\mathcal{B}$ always answers 0, and there is 50% chance of it being the right answer, and hence the win probability of $\mathcal{B}$ is one half. The win probability of adversary $\mathcal{A}$ is the same: $\mathcal{A}$ does not get any information linked to the challenge bit, and thus its answer is as good as guessing but there is always 50% chance of answering right.

Next consider case 1. Now, there are two possibilities for challenge bit b. Suppose first that $b = 1$. In this case, adversary $\mathcal{B}$ wins if and only if $\mathcal{A}$ wins. On the other hand, if the challenge bit b equals 0, adversary $\mathcal{A}$ does not have any information because it is getting the same input regardless of c, so its answer is no better than a guess. Thus the win probability equals $\frac{1}{2}$. Furthermore, the adversary $\mathcal{A}$ wins if and only if adversary $\mathcal{B}$ loses, therefore $\Pr[\mathcal{B}\ wins] = \Pr[\mathcal{A}\ loses] = \frac{1}{2}$.

This case analysis above shows that in all cases $\Pr[\mathcal{B}\ wins] = \Pr[\mathcal{A}\ wins]$. Now continuing with $\Pr[\mathcal{A}\ wins]$ we obtain

$$\begin{aligned}\Pr[\mathcal{A}\ wins] &= \frac{1}{2}\Pr[\mathcal{A}\ wins|b=1] + \frac{1}{2}\Pr[\mathcal{A}\ wins|b=0] \\ &= \frac{1}{2}\cdot\left(\frac{1}{2} + \frac{1}{2}\cdot\mathbf{Adv}_{\mathcal{A}}\right) + \frac{1}{2}\cdot\frac{1}{2} \\ &= \frac{1}{2} + \frac{1}{4}\cdot\mathbf{Adv}_{\mathcal{A}}.\end{aligned}$$

By the definition of advantage of adversary $\mathcal{B}$ we have $\Pr[\mathcal{B}\ wins] = \frac{1}{2}\cdot\mathbf{Adv}_{\mathcal{B}} + \frac{1}{2}$ and therefore we obtain $\mathbf{Adv}_{\mathcal{A}} = 2\cdot\mathbf{Adv}_{\mathcal{B}}$. Now, since the $\mathbf{Adv}_{\mathcal{A}}$ is negligible according to the assumption, $\mathbf{Adv}_{\mathcal{B}}$ is also negligible. □

For our last theorem, we introduce a new condition:

The decryption key d can be efficiently computed from the tuple (F, X). (**Condition** $(***)$)

Theorem 14: The following inclusions hold: $\texttt{GS}(mod.ind2, \Phi) \subseteq \texttt{GS}(obv.ind, \Phi)$ and $\texttt{GS}(mod.sim2, \Phi) \subseteq \texttt{GS}(obv.sim, \Phi)$. If condition $(***)$ holds, then the classes are equal.

Proof: The difference between $ModInd2$ and $ObvInd$ (respectively $ModSim2$ and $ObvSim$) is that in ModInd2 game (in ModSim2 respectively) the adversary receives the decryption key d as an output from GARBLE together with F and X. This auxiliary output does not make the advantage smaller to the adversary in the modified games. The claim follows from this observation. □

These results complete the considerations about the possible relations between the new and old classes of garbling schemes. Results are collected into Figure 3.

If in addition to $(***)$ we require that $(\Phi, \texttt{ev})$ is efficiently invertible (i.e. given $y = \texttt{ev}(f', x')$ and $\phi = \Phi(f')$, f and x such that $y = \texttt{ev}(f, x)$ and $\Phi(f) = \phi$ can be found in polynomial time) and condition $(**)$ also holds, then all the

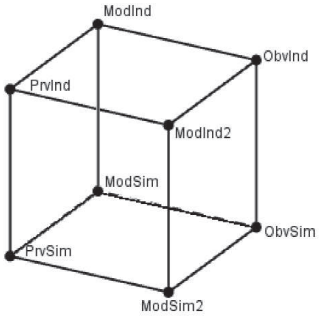


Figure 3: Inclusions between classes of garbling schemes

sets in the diagram collapse into one point: a garbling scheme that belongs to one security class will be secure also with respect to any other security model.

V. CONCLUSIONS

In this paper, we have considered different security classes of garbling schemes. Some of our results are obtained for the classes defined by Bellare, Hoang and Rogaway in [3]. We have also introduced new security classes and described their relation to the earlier classes. From these new classes, we see that the new classes $\mathtt{GS}(mod.ind, \Phi)$ and $\mathtt{GS}(mod.sim, \Phi)$ would be promising targets for future research - at least, it seems that these classes would have practical applications. Namely, our results show that all garbling schemes in the old obv-classes belong also to the new mod-classes, and therefore it is at least as easy to find a garbling scheme that is mod-secure. Moreover, it seems to be harder to find an application which would require obv-security but where mod-security would not suffice. The second new class sets too hard requirements for a secure garbling scheme and this class is practically always empty.

REFERENCES

[1] Y. Aumann and Y. Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. *Proc. of TCC 2007*, 4392 of LNCS:137–156, 2007.
[2] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. *Proc. of the* 22^{nd} *STOC*, pages 503–513, 1990.
[3] M. Bellare, V. T. Hoang, and P. Rogaway. Foundations of garbled circuits. *Proc. of ACM Computer and Communications Security (CCS'12)*, 2012.
[4] M. Bellare and P. Rogaway. Code-based game-playing proofs and the security of triple encryption. *Advances in Cryptology, Proc. of Eurocrypt 2006*, 4004 of LNCS:409–426, 2006.
[5] Y. Lindell and B. Pinkas. A proof of security of Yao's protocol for secure two-party computation. *Electronic Colloquium on Computational Complexity*, TR04-063, 2004.
[6] Y. Lindell and B. Pinkas. A proof of security of Yao's protocol for secure two-party computation. *Journal of Cryptology*, 22(2):161–188, 2009.
[7] A. Yao. How to generate and exchange secrets. *Proc. of* 27^{th} *FOCS, 1986.*, pages 162–167, 1986.

Tommi Meskanen had his PhD in 2005. Since that he has been working as a researcher and lecturer at University of Turku. His main research interests are cryptography and public choice theory. His email address is tommes@utu.fi

Valtteri Niemi is a Professor of Mathematics at the University of Turku, Finland. Between 1997 and 2012 he was with Nokia Research Center in various positions, based in Finland and Switzerland. Niemi was also the chairman of the security standardization group of 3GPP during 2003-2009. His research interests include cryptography and mobile security. Valtteri can be contacted at valtteri.niemi@utu.fi.

Noora Nieminen is a doctoral student at Turku Centre for Computer Science, Department of Mathematics and Statistics at the University of Turku. Her research interests include cryptography and its applications. Contact her at nmniem@utu.fi.

II

Paper II

Hierarchy for Classes of Garbling Schemes

T. Meskanen and V. Niemi and N. Nieminen (2015). *Studia Scientiarum Mathematicarium Hungarica*, 52(2):1-12

Studia Scientiarum Mathematicarum Hungarica 52 (2), 154–175 (2015)
DOI: 10.1556/012.2015.52.2.1315

HIERARCHY FOR CLASSES OF GARBLING SCHEMES

TOMMI MESKANEN[1], VALTTERI NIEMI[1] and NOORA NIEMINEN[1,2]

[1] Department of Mathematics and Statistics, University of Turku, 20014 Turun yliopisto, Finland
e-mails: {tommes, pevani, nmniem}@utu.fi

[2] Turku Centre for Computer Science (TUCS), Finland

Communicated by L. Csirmaz

(Received June 12, 2014; accepted April 30, 2015)

Abstract

The methods for secure outsourcing and secure one-time programs have recently been of great research interest. Garbling schemes are regarded as a promising technique for these applications while Bellare, Hoang and Rogaway introduced the first formal security notions for garbling schemes in [3, 4]. Ever since, even more security notions have been introduced and garbling schemes have been categorized in different security classes according to these notions. In this paper, we introduce new security classes of garbling schemes and build a hierarchy for the security classes including the known classes as well as classes introduced in this paper.

I. Introduction

Garbled circuits are a technique for secure multi-party computations, first introduced by Yao in [18]. Beaver, Micali and Rogaway introduced the term *garbled circuit* in [1] where they also introduced a way of performing secure multi-party computation with Yao's circuit garbling technique. Since then Yao's garbled circuits have been used for various purposes even though there was no formal definition what is meant by garbling, let alone a proof of security. Lindell and Pinkas introduced the first proof of security for a particular garbled circuit using a protocol assuming semi-honest adversaries [14].

2010 *Mathematics Subject Classification.* Primary 94A60.

Key words and phrases. Reusable garbled circuits, garbling schemes, secure multiparty computations, adaptive security.

Finally in 2012, Bellare, Hoang and Rogaway proposed the first formal definition of a garbling scheme in [4]. Not only is the garbling scheme formalized but also various security notions are defined in [4]. The notions, *privacy*, *obliviousness* and *authenticity*, are expressed via *code-based games*. These games are defined in such a way that they capture the intuition behind the different notions. Some of these games and notions are explained in more details in the following two sections.

Another seminal achievement in [4] are relations between different security notions. Bellare et al. also provide two concrete garbling schemes, one of which achieves not only privacy but also obliviousness and authenticity. This example assures that the defined security classes are not empty. Moreover, garbling schemes are also practical in the sense that they can be constructed and evaluated "at unprecedented speed" using the methods presented in [2].

The security notions presented in [4] have recently been extended by Meskanen, Niemi and Nieminen by two new notions *Mod1* and *Mod2* in [15]. The motivation of introducing new classes arises from the need for practicality in applications using garbled circuits. Meskanen et al. show in [15] that it is at least as easy to find a garbling scheme in the new class as it would be to find a garbling scheme that achieves either privacy or obliviousness, showing the practicality of the new notions. They also present the hierarchy for security classes of garbling schemes including the classes from [4, 15].

According to the terminology used by Bellare et al., the security notions defined in [4, 15] are *static*. In addition to static security notions for garbled circuits in [4], Bellare et al. define *adaptive* security notions in [3]. These adaptive notions are required for several important applications such as secure one-time programs and secure one-time outsourcing. Bellare et al. define two types, *coarse-grained* and *fine-grained*, adaptive security for the three security models privacy, obliviousness and authenticity. Using garbling schemes belonging to these different security models, they describe how to transform a garbling scheme of certain type into a secure one-time compiler. Moreover, they show how these adaptive notions can be applied to achieve a secure one-time outsourcing scheme. Further results about both notions are also presented in Hoang's PhD thesis [13].

The static as well as the adaptive security notions mentioned above provide an efficient way of securely evaluate function f on argument x. However, the definitions allow the same garbled circuit for f to be used only once. In other words, to evaluate the same function with different arguments requires function f to be garbled again. Garbling the same f over and over again is practical for small number of times but becomes impractical before long. Other ideas to construct reusable garbling schemes are needed.

Recently, Goldwasser, Kalai, Popa, Vaikuntanathan and Zeldovich proposed a method to construct reusable garbled circuits [9]. The building blocks for a reusable garbled circuit are *functional encryption* (FE) [6], *fully homomorphic encryption* (FHE) [8], *attribute-based encryption* (ABE) [11, 12] as well as garbled circuits [18]. The idea behind the construction is to

use FHE to compute the encrypted value of the function and use the garbled circuit of FHE decryption to recover the final value. The whole circuit cannot be given to the evaluator, because the reusable garbled circuit would not hide x as required by the security notions for garbled circuits. Therefore, ABE is used to provide the evaluator only the necessary information about the garbled circuit for FHE decryption. The functional encryption scheme constructed based on this idea does not hide the circuit. However, Goldwasser et al. show that by hiding the circuit together with the argument using the constructed functional encryption leads to desired goal – a reusable garbled circuit achieving argument and circuit privacy. The security of reusable garbled circuit relies on the security of the constructed FE. The existence of a secure FE scheme relies on the existence of secure FHE and ABE schemes. This requires the *subexponential Learning With Errors assumption* (see [16, 17] for more details about LWE). Moreover, current FHE schemes are not yet practical enough, so the construction of reusable garbled circuits is not yet practical using this method.

Our contributions: Current definitions support either practical garbled circuits for single-use or fully reusable garbled circuits at the cost of practicality. This paper defines new security classes that aim to garbled circuits somewhere between these two extremes, i.e. garbling schemes that provide at least some level of reusability while still being practical to construct.

The main result of this paper is an extended hierarchy for security classes of garbling schemes. This hierarchy includes most of the existing classes from [3, 4, 15] as well as the new security classes introduced in this paper.

This paper begins with the necessary definitions in Section II. The third section describes briefly the hierarchy of static garbling schemes. The fourth section provides descriptions of adaptive security classes. In the fifth section we prove the relations to build up the hierarchy as an extension to the existing hierarchy for classes of static garbling schemes. The complete hierarchy is presented in the sixth section, and conclusions are in Section VII.

II. Definitions

In this section, we provide the basic definitions and notations. As usual, $\mathbb{N}$ will be the set of positive integers. A *string* is a finite sequence of bits. In addition to the basic strings, there is a special symbol $\perp$, meaning of which is explained later where the context of usage will be clearer.

Let S be a finite set. Notation $y \twoheadleftarrow S$ means that an element is selected uniformly at random from the set S, and this element is assigned as a value to the variable y. An algorithm is denoted by A and notation $A(x_1, \ldots, x_n)$ refers to the output of the algorithm A on inputs $x_1, \ldots, x_n$.

As usual, we say that a function $f : \mathbb{N} \to \mathbb{R}$ is negligible if for every $c > 0$ there is an integer N_c such that $\left|f(n)\right| < n^{-c}$ for all $n > N_c$.

A. Code-based games

The proofs in this paper are heavily based on *code-based games.* Following the terminology presented in [5], a game is a collection of procedures called *oracles*. This collection contains three types of procedures: `INITIALIZE`, `FINALIZE` and other named oracles. We assume throughout this paper that procedures `INITIALIZE` and `FINALIZE` are compulsory in a game, whereas other procedures are optional.

The entity playing a game is called *adversary*. Every game starts with `INITIALIZE` procedure. Then the adversary may invoke other procedures before feeding its output to the `FINALIZE` procedure. Based on the input from the adversary `FINALIZE` creates a string that tells the outcome from the game typically consisting of one bit of information: whether the adversary has won or not.

B. Garbling scheme

Formally, a garbling scheme is a 5-tuple $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev})$ of algorithms, where algorithms `Gb` and `En` are probabilistic and the rest are deterministic. Let f denote the string that represents the original function that is to be garbled. The last component in the 5-tuple is the evaluation function $\mathtt{ev}(f, \cdot) : \{0,1\}^n \to \{0,1\}^m$ which we want to garble. Here, the values n and m represent the lengths of the argument x and the function value $y = \mathtt{ev}(f, x)$. The values n and m must also be efficiently computable from f.

The first component `Gb` is the garbling algorithm. It takes f and 1^k as inputs, where $k \in \mathbb{N}$ is a security parameter, and returns (F, e, d) as output. String e describes the particular encryption algorithm $\mathtt{En}(e, \cdot)$ which maps an initial argument x to a garbled argument $X = \mathtt{En}(e, x)$. String F determines the garbled function $\mathtt{Ev}(F, \cdot)$ which returns the garbled function value $Y = \mathtt{Ev}(F, X)$. Finally, string d describes the decryption algorithm $\mathtt{De}(d, \cdot)$ which on a garbled function value Y returns the final function value $y = \mathtt{De}(d, Y)$. We refer to Fig. 1 to get an idea of how a garbling scheme works.

Note that we have slightly generalized the definitions used in earlier papers [3,4,15] by allowing encryption algorithm to be probabilistic. The need for this generalization becomes apparent in Section 5.

C. Side-information function

By the concept of *a side-information function*, we capture the information revealed about f by the garbling process. More formally, a side-

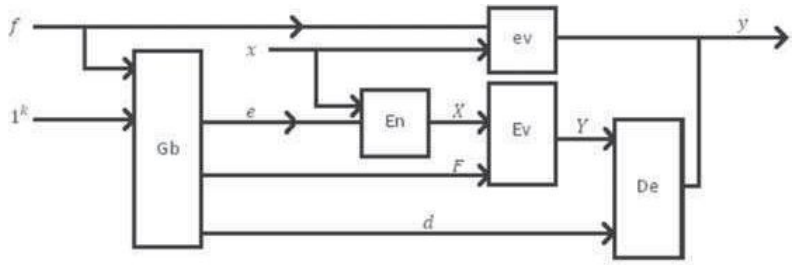


Fig. 1. Description of the technique behind garbling. The diagram also illustrates that the result of evaluation with garbling must coincide with the result obtained without garbling

information function Φ deterministically maps f to $\Phi(f)$. For example, in the case of computation model using circuits and the corresponding evaluation algorithm $\mathtt{ev}_{\mathrm{circ}}$, the value $\Phi(f)$ might be the size of the circuit that was garbled, the topology of the circuit or something else – even the whole initial circuit.

III. The static security notions of garbling schemes

In this section we present shortly the first security notions for a garbling scheme proposed by Bellare et al. in [4]. These notions were extended by new classes of garbling schemes in [15] where Meskanen et al. also presented the hierarchy for the classes garbling schemes that is illustrated in Fig. 3 later in this paper.

Bellare et al. defined three types of security notions: *privacy, obliviousness* and *authenticity*. Furthermore, the first two types have two distinct models: one based on *indistinguishability* and another based on *simulation*. In all cases, the security is defined through a code-based game consisting of initialization procedure `INITIALIZE`, a procedure named `GARBLE` and finalization procedure `FINALIZE`. The procedure `GARBLE` is not to be confused with the garbling function `Gb`: the garbling function `Gb` is a component of a garbling scheme $\mathcal{G}$, whose security the adversary tries to break via the procedure `GARBLE`.

Before the game starts, the garbling scheme $\mathcal{G}$ and the side-information function Φ are fixed for the game. Additionally in simulation model, also the simulator $\mathcal{S}$ is fixed although details of it are not assumed to be known to the adversary. Next we provide a brief description of the games, the complete descriptions are provided in [4] and [15].

PRIVACY: Privacy has two types of models, and we have two different games with distinct `GARBLE` procedures: $\text{PrvIndStat}_{\mathcal{G},\Phi}$ and $\text{PrvSimStat}_{\mathcal{G},\Phi,\mathcal{S}}$. The biggest difference between these two games is that the latter requires an auxiliary algorithm to be defined, namely the simulator $\mathcal{S}$.

Both games consist of procedures `INITIALIZE`, `FINALIZE` and `GARBLE` which are called by the adversary **exactly once** during one game. Informally the game PrvIndStat goes as follows: the `INITIALIZE` procedure first chooses the *challenge bit* uniformly at random. The adversary calls the `GARBLE` procedure having two appropriate functions and their arguments as the feed. Depending on the value of the challenge bit, the `GARBLE` procedure returns a garbled version of one of the functions and its argument, and the adversary guesses which of the functions got garbled. The `FINALIZE` procedure takes two inputs, value of challenge bit from `INITIALIZE` and adversary's guess and tells whether the answer given by the adversary was correct or not, and this will then be the outcome of the game.

The game PrvSimStat differs from PrvIndStat game in `GARBLE` procedure: the adversary provides only one pair of a function (f) and its argument; now the other function, from which the function f is to be distinguished, is devised by the simulator. The adversary must be able to tell the difference between an actual function and a simulated function.

We define the advantage of an adversary A in both games as follows:

$$\mathbf{Adv}(\mathcal{A}, k) = 2 \cdot \Pr[\mathcal{A} \text{ wins the game}] - 1.$$

If the advantage function $\mathbf{Adv}(\mathcal{A}, \cdot)$ is negligible for all `PT` adversaries $\mathcal{A}$ in $\text{PrvIndStat}_{\mathcal{G},\Phi}$ game then we say that the garbling scheme $\mathcal{G}$ is *prv.ind.stat secure over* Φ. Similarly, we define that a garbling scheme $\mathcal{G}$ is *prv.sim.stat secure over* Φ if for every `PT` adversary $\mathcal{B}$ there exists a `PT` simulator $\mathcal{S}$ such that the advantage $\mathbf{Adv}(\mathcal{B}, k)$ in the $\text{PrvSimStat}_{\mathcal{G},\Phi,\mathcal{S}}$ game is negligible.

OBLIVIOUSNESS: At first sight, the games for obliviousness look similar to the privacy games. The difference is that the decryption key d is given to the adversary in the privacy games but not in obliviousness games, and hence the adversary cannot compute the final output $y = \texttt{De}\big(d, \texttt{Ev}(F, X)\big)$. Informally, the adversary is asked to distinguish two functions and their arguments from each other without knowing the result of evaluation.

The adversary has an advantage which is calculated as in the privacy model. The obv.ind.stat and the obv.sim.stat security of a garbling scheme $\mathcal{G}$ are defined similarly as in the corresponding Prv-games.

MATCHABILITY-ONLY: No descriptive name was provided in [15] for the class Mod1. We use name *Matchability-only*, *Mao* for short, whenever we refer to the class Mod1 in paper [15]. The Mao game is a modification of Prv and Obv games. In Prv game, the `GARBLE` procedure returns the decoding function d which is not returned in Obv game. This holds for both security models, simulation-based and indistinguishability-based. In Mao game,

d is not given to the adversary. In the indistinguishability-based games, another difference is the evaluation test $\mathtt{ev}(f_0, x_0) = \mathtt{ev}(f_1, x_1)$ which is a part of Prv game but not Obv game. In MaoIndStat game this test is kept as a part of the game. In simulation-based game, the difference between the Prv and Obv game is the input to the simulator: in Prv game the simulator gets $y = \mathtt{ev}(f, x)$ as an input whereas in Obv game it does not. In the modified MaoSimStat game the simulator may use $y = \mathtt{ev}(f, x)$ to create simulated function.

The adversary has an advantage which is calculated as in the previous models. The mao.ind.stat and the mao.sim.stat security of a garbling scheme $\mathcal{G}$ are defined similarly as in the corresponding Prv and Obv games.

In [15], also a second type of modification to Prv and Obv games was presented. However, it was shown that these classes would almost always be empty, and therefore we do not consider them.

We denote $\mathtt{GS}(\text{xxx.yyy.stat}, \Phi)$ to be the class of all garbling schemes that are xxx.yyy.stat-secure over the side-information function Φ, where xxx denotes the security notion (prv, obv or mao) and yyy denotes the security model (ind or sim).

IV. Classes of adaptive garbling schemes

In the previous section, we introduced the static security classes. In the games defining these classes, the adversary needs to fix both the function(s) and its (their) argument(s) at the moment it calls the `GARBLE` procedure. There are applications, one-time programs [10] and secure outsourcing [7] for example, in which more adaptive notions of security are needed. In this section, we define the security classes of adaptive garbling schemes. These schemes allow the adversary to first fix only the function(s). The argument(s) to the function(s) is(are) fixed after adversary gets the garbled function.

The game definitions rely on the ones used by Bellare, Hoang and Rogaway in [3]. Our definitions generalize the definitions of [3] in such a way that the same garbled function can be used for more than one function evaluation. This extension seems to make sense from the practical point of view. We introduce a new parameter ℓ telling how many times the same garbling scheme can be used for secure function evaluation.

A. Games defining the adaptive security notions

The tables I and II define the games for *coarse-grained* security classes of garbling schemes. By *coarse-grained* we mean that the adversary first specifies the function he wants to garble and after receiving the garbled function

F the adversary sends the entire argument, x or (x_0, x_1) depending on the model, to be encrypted at one go.

All games start with procedure INITIALIZE, in which the challenge bit b in the game is chosen uniformly at random. Procedure GARBLE prepares garbled function F to the adversary and INPUT encrypts the argument x of the original function f. These two procedures are slightly different for the different security notions: GARBLE is the same for notions Mao and Obv, whereas INPUT is the same for Prv and Mao. FINALIZE procedure is again the same for all notions.

proc INITIALIZE() **Prv** **Mao** **Obv** $b \leftarrow \{0,1\}$		
proc GARBLE(f) **Prv** **if** $b = 1$ **then** $(F, e, d) \leftarrow \mathtt{Gb}(1^k, f)$ **else** $(F, d) \leftarrow \mathcal{S}(1^k, \Phi(f))$ **return** (F, d)	**proc** GARBLE(f) **Mao** **if** $b = 1$ **then** $(F, e, d) \leftarrow \mathtt{Gb}(1^k, f)$ **else** $F \leftarrow \mathcal{S}(1^k, \Phi(f))$ **return** F	**proc** GARBLE(f) **Obv** **if** $b = 1$ **then** $(F, e, d) \leftarrow \mathtt{Gb}(1^k, f)$ **else** $F \leftarrow \mathcal{S}(1^k, \Phi(f))$ **return** F
proc INPUT(x) **Prv** **if** $x \notin \{0,1\}^n$ **then** **return** $\perp$ **if** $b = 1$ **then** $X \leftarrow \mathtt{En}(e, x)$ **else** $X \leftarrow \mathcal{S}(\mathtt{ev}(f, x))$ **return** X	**proc** INPUT(x) **Mao** **if** $x \notin \{0,1\}^n$ **then** **return** $\perp$ **if** $b = 1$ **then** $X \leftarrow \mathtt{En}(e, x)$ **else** $X \leftarrow \mathcal{S}(\mathtt{ev}(f, x))$ **return** X	**proc** INPUT(x) **Obv** **if** $x \notin \{0,1\}^n$ **then** **return** $\perp$ **if** $b = 1$ **then** $X \leftarrow \mathtt{En}(e, x)$ **else** $X \leftarrow \mathcal{S}()$ **return** X
proc FINALIZE(b') **Prv** **Mao** **Obv** **return** $b = b'$		

Table I. The games defining the $\mathrm{PrvSimAdap}_\ell$, $\mathrm{MaoSimAdap}_\ell$ and $\mathrm{ObvSimAdap}_\ell$ security notions. The letter n in the game represents the length of the argument x. The game depends on the choice of $\mathcal{G}$, Φ and $\mathcal{S}$. Integer ℓ tells the upper limit for the number of INPUT queries

Next we briefly explain how the game progresses for different security notions in both models.

Let us start with the game Prv (the leftmost column in tables I and II). The game starts with a call to INITIALIZE procedure. It is followed by a query to GARBLE procedure. In simulation-based game, the adversary chooses one function f. Based on the value of the challenge bit b, either the actual function f becomes garbled or the simulator creates F that resembles a garbling of a real function. In indistinguishability-based game, the adversary chooses two functions, f_0 and f_1, and the challenge bit b determines which of these two functions is garbled. After this, the adversary may call procedure INPUT at most ℓ times. In simulation-based notion, INPUT procedure takes a single argument x as an input whereas in indistinguishability-based notion, INPUT takes two arguments (x_0, x_1), one for each function, as input. Each time, GARBLE returns one encryption of one argument or a simulated value

proc INITIALIZE()			**Prv** / **Mao** / **Obv**
$b \leftarrow \{0,1\}$			
proc GARBLE(f_0, f_1) **Prv**	**proc** GARBLE(f_0, f_1) **Mao**	**proc** GARBLE(f_0, f_1) **Obv**	
if $\Phi(f_0) \neq \Phi(f_1)$ **then**	**if** $\Phi(f_0) \neq \Phi(f_1)$ **then**	**if** $\Phi(f_0) \neq \Phi(f_1)$ **then**	
return $\perp$	**return** $\perp$	**return** $\perp$	
$(F, e, d) \leftarrow \mathtt{Gb}(1^k, f_b)$	$(F, e, d) \leftarrow \mathtt{Gb}(1^k, f_b)$	$(F, e, d) \leftarrow \mathtt{Gb}(1^k, f_b)$	
return (F, d)	**return** F	**return** F	
proc INPUT(x_0, x_1) **Prv**	**proc** INPUT(x_0, x_1) **Mao**	**proc** INPUT(x_0, x_1) **Obv**	
if $\Phi(f_0) \neq \Phi(f_1)$ **then**	**if** $\Phi(f_0) \neq \Phi(f_1)$ **then**	**if** $\Phi(f_0) \neq \Phi(f_1)$ **then**	
return $\perp$	**return** $\perp$	**return** $\perp$	
if $\{x_0, x_1\} \not\subseteq \{0,1\}^n$	**if** $\{x_0, x_1\} \not\subseteq \{0,1\}^n$	**if** $\{x_0, x_1\} \not\subseteq \{0,1\}^n$	
then return $\perp$	**then return** $\perp$	**then return** $\perp$	
if $\mathtt{ev}(f_0, x_0) \neq \mathtt{ev}(f_1, x_1)$	**if** $\mathtt{ev}(f_0, x_0) \neq \mathtt{ev}(f_1, x_1)$		
then return $\perp$	**then return** $\perp$		
return $\mathtt{En}(e, x_b)$	**return** $\mathtt{En}(e, x_b)$	**return** $\mathtt{En}(e, x_b)$	
proc FINALIZE(b')			**Prv** / **Mao** / **Obv**
return $b = b'$			

Table II. The games defining the PrvIndAdap$_\ell$, MaoIndAdap$_\ell$ and ObvIndAdap$_\ell$ games. The letter n represents the length of the argument x (which is the same for both functions f_0 and f_1). For each $\mathcal{G}$ and Φ we have a different game. Integer ℓ tells the upper limit for number of INPUT queries

depending on the challenge bit b. Now, based on the outputs from GARBLE and INPUT, the adversary must answer the challenge, i.e. try to determine whether $b = 0$ or $b = 1$. Adversary's answer is sent to procedure FINALIZE which finally tells, whether the adversary answered correctly.

The games for notion Obv (the rightmost column in tables I and II) are like the corresponding Prv games with the difference that the decryption function d is not provided to the adversary. Another difference to the Prv game appears in INPUT procedure. The evaluation test $\mathtt{ev}(f_0, x_0) \neq \mathtt{ev}(f_1, x_1)$ is not a part of INPUT procedure in indistinguishability-based game and in the simulation-based game the simulator needs to create a garbled argument X without knowing $y = \mathtt{ev}(f, x)$.

The notion Mao (the middle columns in tables I and II) appears to be an intermediate notion for Prv and Obv, because it has some features from both model, namely GARBLE in common with Obv, INPUT in common with Prv. In other words an adversary in a Mao game does not learn the decryption key d but it must still pass the evaluation test.

B. About fine-grained garbling schemes

In the earlier work [3] also the fine-grained adaptive privacy notions were defined. In these notions the level of adaptivity is increased by the chance

of providing the argument x to function f bit by bit instead of giving the whole argument at once as was done in coarse-grained games. With small modifications we could define also these notions in the non-deterministic setting. For all the results we prove later for coarse-grained classes it is possible to prove an analogous result for fine-grained classes. However, we will not consider the fine-grained adaptive privacy classes in this paper.

C. The security classes of adaptive garbling schemes

Now we have defined all the games for adaptive garbling schemes. Next we provide the concept of an advantage of an adversary. Furthermore, we give the definition for adaptive security of a garbling scheme.

Definition (Advantage of an adversary). Let adversary $\mathcal{A}$ be playing the adaptive game $\mathrm{XxxYyyAdap}_\ell$ where $\mathrm{Xxx} \in \{\mathrm{Prv}, \mathrm{Mao}, \mathrm{Obv}\}$ and $\mathrm{Yyy} \in \{\mathrm{Ind}, \mathrm{Sim}\}$. The advantage of adversary $\mathcal{A}$ in this game is

$$\mathbf{Adv}(\mathcal{A}, k) = 2 \cdot \Pr[\mathcal{A} \text{ wins } \mathrm{XxxYyyAdap}_\ell \text{ game}] - 1.$$

Definition (Adaptive security of a garbling scheme). We say that a garbling scheme $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev})$ is $\mathrm{xxx.ind.adap}_\ell$ secure over Φ, if for any $\mathtt{PT}$ adversary $\mathcal{A}$ the advantage $\mathbf{Adv}(\mathcal{A}, \cdot)$ is negligible in $\mathrm{XxxIndAdap}_\ell$ game. Respectively, $\mathcal{G}$ is $\mathrm{xxx.sim.adap}_\ell$ secure if for any $\mathtt{PT}$ adversary $\mathcal{A}$ there exists a $\mathtt{PT}$ simulator $\mathcal{S}$ such that $\mathbf{Adv}(\mathcal{A}, \cdot)$ is negligible in $\mathrm{XxxSimAdap}_\ell$ game.

V. Building up the hierarchy

In this section we prove relations between the classes of adaptive garbling schemes we defined in the previous section. As a result we obtain for each value of ℓ a hierarchy that is identical with the hierarchy for classes of static garbling schemes.

The next theorem tells us that the more `INPUT` queries are allowed the harder it is to find a secure garbling scheme.

Theorem 1. *Let $\mathcal{C}_\ell$ be a class of adaptively secure garbling scheme, i.e. $\mathcal{C}_\ell = \mathtt{GS}(\mathrm{xxx.yyy.adap}_\ell, \Phi)$ for $\mathrm{xxx} \in \{\mathrm{prv}, \mathrm{mao}, \mathrm{obv}\}$ and $\mathrm{yyy} \in \{\mathrm{ind}, \mathrm{sim}\}$ and $\ell \in \mathbb{N}$. Then $\mathcal{C}_{\ell+1} \subsetneqq \mathcal{C}_\ell$ or $\mathcal{C}_{\ell+1} = \mathcal{C}_\ell = \emptyset$.*

Proof. The claim $\mathcal{C}_{\ell+1} \subseteqq \mathcal{C}_\ell$ is fairly obvious, because the adversary playing the game related to class $\mathcal{C}_{\ell+1}$ can always use only ℓ `INPUT` calls instead of the maximum allowed number of calls.

Next we need to prove that in the case $\mathcal{C}_\ell \neq \emptyset$ there exist a garbling scheme that belongs to class $\mathcal{C}_\ell$ but does not belong to class $\mathcal{C}_{\ell+1}$. Let

$\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev})$ be a garbling scheme in $\mathcal{C}_\ell$. We construct another garbling scheme $\mathcal{G}' = (\mathtt{Gb}', \mathtt{En}', \mathtt{De}, \mathtt{Ev}', \mathtt{ev})$ by modifying three components of $\mathcal{G}$, namely Gb, En and Ev.

Intuitively the most important modification is the encrypting algorithm $\mathtt{En}'$. The modified encrypting algorithm $\mathtt{En}'$ consists of three parts. The first part contains the original encrypting algorithm En. The second component is a completely independent symmetric non-deterministic secure encryption scheme $\mathtt{En}_2$ that is used to encrypt x with an independent key e_2. Finally, the third component is a secret sharing scheme that creates t shares of key e_2 in such a way that $\ell + 1$ shares are needed to reconstruct the key e_2. The garbled argument X' is obtained with algorithm $\mathtt{En}'$ as follows: $\mathtt{En}'\big((e, e_2, t \text{ shares}), x\big) = \big(\mathtt{En}(e, x), \mathtt{En}_2(e_2, x), s\big) = (X, X_2, s)$ where s is a random share of e_2.

The garbling algorithm $\mathtt{Gb}'$ creates (F, e, d) in the similar manner as Gb would if scheme $\mathcal{G}$ had been used. In addition, it has to create e_2 by taking the security parameter k into account (the encryption with e_2 cannot be breakable in polynomial time with respect to k). Moreover, $\mathtt{Gb}'$ creates t shares of e_2. The output of algorithm $\mathtt{Gb}'$ is $\big((F, F_2), (e, e_2, t \text{ shares of key } e_2), d\big)$ where F is the garbled function and $F_2 = \mathtt{En}_2(e_2, f)$.

The garbled evaluation algorithm $\mathtt{Ev}'$ takes $\big((F, F_2), (X, X_2, s_i)\big)$ as its inputs. The algorithm omits the new parts F_2, X_2, s_i and computes $Y = \mathtt{Ev}'\big((F, F_2), (X, X_2, s)\big) = \mathtt{Ev}(F, X)$.

Our aim is now to prove that the constructed garbling scheme $\mathcal{G}'$ belongs to class $\mathcal{C}_\ell$ but not to class $\mathcal{C}_{\ell+1}$. The first claim follows from the fact that any adversary playing a game with at most ℓ INPUT calls obtains only ℓ random shares of key e_2 which are not useful in reconstructing the key and hence in discovering x and f. For the second part of the claim, let us consider an arbitrary adversary who is allowed to make $\ell + 1$ INPUT queries in its game. This adversary gets $\ell + 1$ shares of e_2 and there is a certain positive probability not depending on k that these shares can be used to reconstruct e_2. As a consequence, the adversary has a chance to discover f and x. Using that information the adversary would be able to find the challenge bit b and win the game. This leads to non-negligible advantage of the adversary. This shows that $\mathcal{G}'$ cannot belong to security class $\mathcal{C}_{\ell+1}$. □

Note that the assumption of encryption function being non-deterministic is relevant for Theorem 1. If the En function would be deterministic an adversary could find out b from the returned encrypted arguments by calling INPUT twice with the same input x in simulation model and changing one of the input arguments in indistinguishability model.

As a corollary we get that $\mathcal{C}_\ell \subsetneqq \mathcal{C}_1$ for $\ell \geqq 2$. The classes $\mathcal{C}_1$ are exactly the classes defined by Bellare, Hoang and Rogaway in [3]. The extension of non-deterministic encryption does not make a difference if INPUT is called only once.

Next we remove the constraint that the INPUT procedure can be called at most ℓ times, allowing the adversary playing the security game to call INPUT arbitrarily many times.

DEFINITION. Let xxx $\in \{\text{prv}, \text{mao}, \text{obv}\}$ and yyy $\in \{\text{ind}, \text{sim}\}$. We define a new class $\mathcal{C}_* = \mathtt{GS}(\text{xxx.yyy.adap}_*, \Phi)$ which is the class of secure garbling schemes when the adversary is allowed to call the INPUT procedure arbitrarily many times in the XxxYyyAdap game.

A garbling scheme belonging to some $*$-class allows the same garbled circuit to be used arbitrarily many times. This is exactly what the reusable garbled circuits introduced by Goldwasser et al. provide [9].

THEOREM 2. *Let us again denote* $\mathcal{C}_\ell = \mathtt{GS}(\text{xxx.yyy.adap}_\ell, \Phi)$ *for* xxx $\in \{\text{prv}, \text{mao}, \text{obv}\}$ *and* yyy $\in \{\text{ind}, \text{sim}\}$. *Then we have the following inclusion:*

$$\mathcal{C}_* \subsetneqq \bigcap_{\ell=1}^{\infty} \mathcal{C}_\ell \qquad \textit{or} \qquad \mathcal{C}_* = \bigcap_{\ell=1}^{\infty} \mathcal{C}_\ell = \emptyset$$

PROOF. It is fairly obvious that $\mathcal{C}_* \subseteqq \mathcal{C}_\ell$ for any $\ell \in \mathbb{N}$ because the adversary playing the game related to $\mathcal{C}_*$ can always make arbitrarily many calls to INPUT procedure, especially exactly ℓ calls. The claim $\mathcal{C}_* \subseteqq \bigcap_{\ell=1}^{\infty} \mathcal{C}_\ell$ follows.

The second part of the theorem claims that there is a garbling scheme that belongs to the intersection $\bigcap_{\ell=1}^{\infty} \mathcal{C}_\ell$ but does not belong to the class $\mathcal{C}_*$ assuming that the intersection is nonempty. To prove this claim, we start with a garbling scheme $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, Ev, \mathtt{ev}) \in \bigcap_{\ell=1}^{\infty} \mathcal{C}_\ell$ and construct another garbling scheme $\mathcal{G}' = (\mathtt{Gb}', \mathtt{En}', \mathtt{De}, \mathtt{Ev}', \mathtt{ev})$ in similar way as in the proof of the previous theorem. The modified garbling algorithm, as well as the modified evaluation function, is as in the previous proof. The encrypting function is mostly as in the proof of 1 with the exception that the number of secret shares needed to reconstruct the key e_2 is k, i.e. is the security parameter.

First we prove that $\mathcal{G}' \in \bigcap_{\ell=1}^{\infty} \mathcal{C}_\ell$. To do that, we first prove that $\mathcal{G}' \in \mathcal{C}_\ell$ for all $\ell \in \mathbb{N}$ from which the claim follows. Consider an arbitrary adversary playing the game related to class $\mathcal{C}_\ell$. The win probability of the adversary now depends on the parameters ℓ and k, because these two variables determine whether the adversary could have enough shares to reconstruct the second encryption key e_2. If $k > \ell$ then the adversary does not have enough shares to reconstruct e_2, and the advantage in the game will be the same as the adversary would have with respect to garbling scheme $\mathcal{G}$, i.e. the advantage is negligible.

Finally, we have the claim $\mathcal{G}' \notin \mathcal{C}_*$. An adversary playing the game related to $\mathcal{C}_*$ is allowed to call INPUT procedure arbitrarily many times, especially more than k times. If the adversary calls INPUT procedure k^2 times then the well-known approximations related to birthday paradox imply that

there is a non-zero probability not depending from k, which leads to non-negligible advantage of the adversary. □

As discussed above before the theorem, the garbling scheme constructed in [9] is a good candidate for a member of $\mathcal{C}_*$, at least in the case of prv.sim.adap. Therefore, the latter possibility of $\mathcal{C}_* = \emptyset$ seem not to hold in all cases.

The next theorem gives a relation between the length of the garbled argument X and the length of the original argument x in the case $\ell = 2$. The theorem also gives some estimate about how much non-determinism is needed in encryption algorithm.

THEOREM 3. *Let* $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev}) \in \mathtt{GS}(\text{prv.sim.adap}_2, \Phi)$ *be a* prv.sim.adap$_2$ *secure garbling scheme. For every constant* $c > 0$ *there exists a threshold* $k_c \in \mathbb{N}$ *for the security parameter* k *such that whenever* k *exceeds* k_c *the length of the garbled argument* X *must be at least* $c \cdot \log k + n$ *where* n *is the length* $|x|$ *of the original argument* x.

PROOF. Suppose on the contrary that there exists a constant $c > 0$ such that for all values k_c, $|X| \leqq c \cdot \log(k) + n$ for some $k > k_c$ and for some x. We give the proof only in case $|X|$ does not depend on the actual value of x. Now any argument x will have in average $\frac{2^{c \cdot \log(k)+n}}{2^n} = 2^{\log(k^c)} = k^c$ possible encodings. We also assume that it is possible to find two arguments x and x' having exactly k^c equally probable encodings. Let us design an adversary $\mathcal{A}$ who plays the PrvSimAdap$_\ell$ game. Our aim is to show that $\mathcal{A}$ has a strategy that will give her a non-negligible advantage and it follows that $\mathcal{G}$ is insecure.

Adversary $\mathcal{A}$ chooses her argument at random from $\{x, x'\}$ and calls `INPUT` two times using the randomly chosen arguments x_1 and x_2. Now, if $X_1 = X_2$ and $x_1 = x_2$ then the adversary answers $b' = 1$. If $X_1 = X_2$ but $x_1 \neq x_2$ then the adversary answers $b' = 0$. If $X_1 \neq X_2$ then the adversary just guesses b'. Let us now analyze the winning probability of adversary $\mathcal{A}$.

- Case $b = 1$: In this case, the actual garbling algorithm is used to create X_1 and X_2. Adversary $\mathcal{A}$ wins the game, if $X_1 = X_2$ and $x_1 = x_2$. The latter happens with probability $\frac{1}{2}$ whereas the previous with probability $\frac{1}{k^c}$ given $x_1 = x_2$. The case $X_1 = X_2$ while $x_1 \neq x_2$ is not possible, because X_1 and X_2 are created using the encryption algorithm `En`. Adversary wins also with probability $\frac{1}{2}$ if $X_1 \neq X_2$, because in this case $\mathcal{A}$ guesses and a guess is correct with probability $\frac{1}{2}$. Thus the winning probability of adversary $\mathcal{A}$ is $\Pr[\mathcal{A} \text{ wins} \mid b = 1] = \frac{1}{4} \cdot \frac{1}{k^c} + \frac{1}{2}$.
- Case $b = 0$: In this case, simulator $\mathcal{S}$ is used to create X_1 and X_2. Adversary $\mathcal{A}$ wins the game, if $X_1 = X_2$ and $x_1 \neq x_2$. On the other hand, if $X_1 = X_2$ and $x_1 = x_2$, $\mathcal{A}$ will lose the game. Adversary $\mathcal{A}$ will win with probability $\frac{1}{2}$ if $X_1 \neq X_2$ because $\mathcal{A}$ guesses in this case

and a guess is correct with this probability. Therefore we obtain as a summary that $\Pr[\mathcal{A} \text{ wins} \mid b = 0] = \frac{1}{2}$.

Now, combining these probabilities we get

$$\Pr[\mathcal{A} \text{ wins}] = \frac{1}{2}\Pr[\mathcal{A} \text{ wins} \mid b = 1] + \frac{1}{2}\Pr[\mathcal{A} \text{ wins} \mid b = 0]$$
$$= \frac{1}{2} \cdot \left(\frac{1}{4} \cdot \frac{1}{k^c} + \frac{1}{2}\right) + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8} \cdot \frac{1}{k^c} + \frac{1}{2}.$$

The advantage of adversary $\mathcal{A}$ is now

$$\mathbf{Adv}_{\mathcal{A}} = 2 \cdot \Pr[\mathcal{A} \text{ wins}] - 1 = 2 \cdot \left(\frac{1}{8} \cdot \frac{1}{k^c} + \frac{1}{2}\right) - 1 = \frac{1}{4k^c}.$$

Now we see that adversary $\mathcal{A}$ has a non-negligible advantage regardless of how good the simulator $\mathcal{S}$ in the game is. Therefore the garbling scheme $\mathcal{G}$ cannot belong to security class prv.sim.adap_2 which is a contradiction. □

Next we show that many known results for coarse-grained garbling scheme classes with $\ell = 1$ `INPUT` queries hold also for the classes allowing $\ell > 1$ `INPUT` queries. Therefore, also the classes of garbling schemes enjoy the same hierarchy for any $\ell \in \mathbb{N}$.

THEOREM 4. *Let $\ell \in \mathbb{N}$. We have the following inclusions:*

(i) $\texttt{GS}(\text{prv.sim.adap}_\ell, \Phi) \subsetneqq \texttt{GS}(\text{prv.ind.adap}_\ell, \Phi)$;

(ii) $\texttt{GS}(\text{obv.sim.adap}_\ell, \Phi) \subsetneqq \texttt{GS}(\text{obv.ind.adap}_\ell, \Phi)$;

(iii) $\texttt{GS}(\text{mao.sim.adap}_\ell, \Phi) \subsetneqq \texttt{GS}(\text{mao.ind.adap}_\ell, \Phi)$.

PROOF. Suppose that $\mathcal{G} \in \texttt{GS}(\text{xxx.sim.adap}_\ell, \Phi)$. Our aim is to prove that $\mathcal{G} \in \texttt{GS}(\text{xxx.ind.adap}_\ell, \Phi)$. Let $\mathcal{A}$ be an adversary playing the XxxIndAdap_ℓ game and let us construct a `PPT` adversary $\mathcal{B}$ for the XxxSimAdap_ℓ game. Let $\mathcal{B}$ run $\mathcal{A}$ as a subroutine.

The game XxxSimAdap_ℓ starts with `INITIALIZE` procedure in which the challenge bit is chosen uniformly at random. After that, the game tells the adversary $\mathcal{B}$ to start the game. Adversary $\mathcal{B}$ in turn challenges $\mathcal{A}$ to play a game XxxIndAdap_ℓ. Adversary $\mathcal{A}$ presumes that the challenge comes from a real XxxIndAdap_ℓ game, so $\mathcal{A}$ prepares its `GARBLE` input f_0, f_1 and sends them to $\mathcal{B}$. If $\Phi(f_0) \neq \Phi(f_1)$ $\mathcal{B}$ sends $\perp$ to $\mathcal{A}$. $\mathcal{B}$ lets $c \leftarrow \{0,1\}$ and sends f_c to its `GARBLE` regardless of what was the result of $\Phi(f_0) \stackrel{?}{=} \Phi(f_1)$. Up to this point the games progress similarly in all three cases, but from now on the progress is slightly different. The progress of the game PrvSimAdap_ℓ is illustrated in Fig. 2, the other cases follow the same idea.

In game PrvSimAdap_ℓ $\mathcal{B}$ gets (F, d) as a return. $\mathcal{B}$ sends (F, d) to $\mathcal{A}$ if $\Phi(f_0) = \Phi(f_1)$. Now $\mathcal{A}$ starts sending its arguments x_0, x_1 to $\mathcal{B}$. If

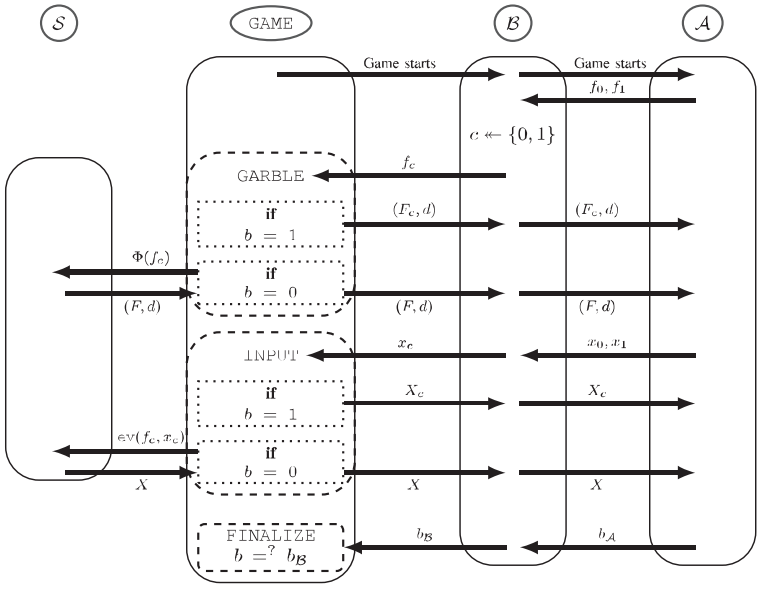


Fig. 2. In the above figure, $\mathcal{B}$ is an adversary playing the game PrvSimAdap_ℓ denoted in the diagram by `GAME`. $\mathcal{S}$ is the simulator in this game. $\mathcal{A}$ is an adversary presuming to play game PrvIndAdap_ℓ, but actually adversary $\mathcal{B}$ uses $\mathcal{A}$ as a subroutine by trying to emulate the actual game PrvIndAdap_ℓ

$\Phi(f_0) \neq \Phi(f_1)$, $\{x_0, x_1\} \not\subseteq \{0,1\}^n$ or $\mathsf{ev}(f_0, x_0) \neq \mathsf{ev}(f_1, x_1)$, $\mathcal{B}$ sends $\perp$ to $\mathcal{A}$. Regardless of possibly sending $\perp$ to $\mathcal{A}$, $\mathcal{B}$ sends x_c's to its own `INPUT`. Every time, $\mathcal{B}$ gets X as a return. $\mathcal{B}$ forwards X to $\mathcal{A}$ if none of the previous tests failed. This may be repeated at most ℓ times.

In game MaoSimAdap_ℓ adversary $\mathcal{B}$ gets F as a return. $\mathcal{B}$ sends F to $\mathcal{A}$ if $\Phi(f_0) = \Phi(f_1)$. Now $\mathcal{A}$ starts sending its arguments x_0, x_1 to $\mathcal{B}$. If $\Phi(f_0) \neq \Phi(f_1)$, $\{x_0, x_1\} \not\subseteq \{0,1\}^n$ or $\mathsf{ev}(f_0, x_0) \neq \mathsf{ev}(f_1, x_1)$, $\mathcal{B}$ sends $\perp$ to $\mathcal{A}$. Regardless of possibly sending $\perp$ to $\mathcal{A}$, $\mathcal{B}$ sends x_c's to its own `INPUT`. Every time, $\mathcal{B}$ gets X as a return and forwards it to $\mathcal{A}$ if none of the previous tests failed. This may be repeated at most ℓ times.

In ObvSimAdap_ℓ game $\mathcal{B}$ gets F as a return. $\mathcal{B}$ sends F to $\mathcal{A}$ if $\Phi(f_0) = \Phi(f_1)$. Now $\mathcal{A}$ starts sending its arguments x_0, x_1 to $\mathcal{B}$. If $\Phi(f_0) \neq \Phi(f_1)$ or $\{x_0, x_1\} \not\subseteq \{0,1\}^n$, $\mathcal{B}$ sends $\perp$ to $\mathcal{A}$. Regardless of possibly sending $\perp$ to $\mathcal{A}$, $\mathcal{B}$ sends x_c's to its own `INPUT`. Every time, $\mathcal{B}$ gets X as a return and forwards it to $\mathcal{A}$ if none of the previous tests failed. This may be repeated at most ℓ times.

When $\mathcal{A}$ returns its answer $b_{\mathcal{A}}$ to the challenge in any of the three games, $\mathcal{B}$ does the same answering $b_{\mathcal{B}} = 1$ if and only if none of the tests done for the input (f_0, f_1, x_0, x_1) lead to $\perp$ during the game for each input (f_0, f_1, x_0, x_1) and $b_{\mathcal{A}} = c$. $\mathcal{B}$ answers 0 otherwise.

Let us now consider the different scenarios of the game from the perspective of the adversary $\mathcal{B}$ and derive a win probability of $\mathcal{B}$ to each of these scenarios.

If $\Phi(f_0) \neq \Phi(f_1)$ then $\mathcal{B}$ answers 0 regardless of $\mathcal{A}$'s answer. This is the correct answer with probability $\frac{1}{2}$. If $\Phi(f_0) = \Phi(f_1)$, but for all inputs (x_0, x_1) given by adversary $\mathcal{A}$ some of the test for $\mathcal{A}$ fails. Also in this case $\mathcal{B}$ answers 0 regardless of $\mathcal{A}$'s answer. The win probability of $\mathcal{B}$ is therefore $\frac{1}{2}$. If $\Phi(f_0) = \Phi(f_1)$ and there is at least one $\mathcal{A}$'s `INPUT` query that does not result to $\perp$. There are two possibilities for b. First consider case $b = 0$. Then, $\mathcal{A}$ does not have any information about X or F, so its answer is no better than a guess. Guessing right still has probability $\frac{1}{2}$. Moreover, $\mathcal{B}$ loses its game if and only if $\mathcal{A}$ wins its game. Consider for example the following scenario. If $b_{\mathcal{A}} = 0$ and $c = 0$ then $b_{\mathcal{B}} = 1$ because $b_{\mathcal{A}} = c$. It follows that adversary $\mathcal{A}$ wins its game ($b_{\mathcal{A}} = c$), but adversary $\mathcal{B}$ loses its game ($b_{\mathcal{B}} \neq b$). The three other possibilities end up in a similar situation where the other adversary wins its game if and only if the other loses.

On the other hand, if $b = 1$ then $\mathcal{A}$ has a chance to figure out the correct $b_{\mathcal{A}}$ because F and X are created by the actual garbling algorithm from either f_0, x_0 or f_1, x_1. Therefore, $\mathcal{A}$ has an advantage $\mathbf{Adv}_{\mathcal{A}}$ of answering correctly to the challenge in its game. Adversary $\mathcal{B}$ wins the game only if $b_{\mathcal{A}} = c$ and $\mathcal{A}'s$ inputs pass all the tests made by adversary $\mathcal{B}$. The win probability of $\mathcal{B}$ is now $\frac{1}{2} + \frac{1}{2} \cdot \mathbf{Adv}_{\mathcal{A}}$.

This case analysis lets us derive the win probability of adversary $\mathcal{B}$ against any `PT` simulator $\mathcal{S}$ as follows.

$$\begin{aligned}\Pr[\mathcal{B} \text{ wins}] &= \frac{1}{2}\Pr[\mathcal{B} \text{ wins} \mid b = 1] + \frac{1}{2}\Pr[\mathcal{B} \text{ wins} \mid b = 0] \\ &= \frac{1}{2}\left(\frac{1}{2} + \frac{1}{2}\mathbf{Adv}_{\mathcal{A}}\right) + \frac{1}{2}\cdot\frac{1}{2} = \frac{1}{2} + \frac{1}{4}\cdot\mathbf{Adv}_{\mathcal{A}}\,.\end{aligned}$$

Now, using $\Pr[\mathcal{B} \text{ wins}] = \frac{1}{2} + \frac{1}{2} \cdot \mathbf{Adv}_B$, we obtain $\mathbf{Adv}_A = 2 \cdot \mathbf{Adv}_{\mathcal{B}}$. We assumed $\mathcal{G}$ to be prv.sim.adap$_\ell$ secure, so $\mathbf{Adv}_{\mathcal{B}}$ is negligible for some simulator $\mathcal{S}$. Therefore also $\mathbf{Adv}_{\mathcal{A}}$ is negligible which proves the claim. □

THEOREM 5. *The following inclusion holds for any $\ell \in \mathbb{N}$:*

$$\mathtt{GS}(\text{prv.ind.adap}_\ell, \Phi) \bigcup \mathtt{GS}(\text{obv.ind.adap}_\ell, \Phi) \subsetneq \mathtt{GS}(\text{mao.ind.adap}_\ell, \Phi).$$

PROOF. Let us start with the inclusion

$$\texttt{GS}(\text{prv.ind.adap}_\ell, \Phi) \subsetneqq \texttt{GS}(\text{mao.ind.adap}_\ell, \Phi).$$

Suppose that $\mathcal{G}$ is a prv.ind.adap$_\ell$ secure garbling scheme over Φ. Recall that the only difference in the two games is that in PrvIndAdap$_\ell$ game the adversary is allowed to get the decryption key d whereas in the PrvMaoAdap$_\ell$ it is not allowed to get d. Dropping d out of the output of `GARBLE` procedure does not increase the winning chances of any adversary.

Let's now consider the second inclusion

$$\texttt{GS}(\text{obv.ind.adap}_\ell, \Phi) \subsetneqq \texttt{GS}(\text{mao.ind.adap}_\ell, \Phi).$$

Suppose that $\mathcal{G} \in \texttt{GS}(\text{obv.ind.adap}_\ell, \Phi)$. Recall that evaluation test is a part of `INPUT` procedure in game MaoIndAdap$_\ell$ game but not in game ObvIndAdap$_\ell$, otherwise the games are identical. As a consequence, the same input (f_0, f_1, x_0, x_1) passing the other tests except the evaluation test leads to $\perp$ in MaoIndAdap$_\ell$ game whereas in game ObvIndAdap$_\ell$ (f_0, f_1, x_1, x_0) does not lead to $\perp$. Therefore, if `INPUT` procedure is not called in the game, adversaries in both games have an equal amount of information because of the identical `GARBLE` procedures and the adversaries have an equal winning chance in their games. On the other hand, if the `INPUT` procedure is called, the adversary in MaoIndAdap$_\ell$ game cannot have a better chance of winning than the adversary playing the ObvIndAdap$_\ell$ game, because the adversary in the previous game must pass the evaluation test every time `INPUT` is called and the procedure does not return $\perp$. □

THEOREM 6. *The following inclusion holds for any $\ell \in \mathbb{N}$:*

$$\texttt{GS}(\text{prv.sim.adap}_\ell, \Phi) \bigcup \texttt{GS}(\text{obv.sim.adap}_\ell, \Phi) \subsetneqq \texttt{GS}(\text{mao.sim.adap}_\ell, \Phi).$$

PROOF. First, suppose that garbling scheme $\mathcal{G}$ is prv.sim.adap$_\ell$ secure. The only difference in the two games is that mao.sim.adap$_\ell$ adversary does not get d whereas prv.sim.adap$_\ell$ adversary gets it as a return from `GARBLE` procedure. Omitting the decryption key d from (F, d) does not increase the winning probability of an adversary playing the MaoSimAdap$_\ell$ game.

Secondly, suppose that the garbling scheme $\mathcal{G}$ is obv.sim.adap$_\ell$ secure. Intuitively it is clear that in the MaoSimAdap$_\ell$ game, the simulator's additional input y cannot make its task of producing a good output (F, X) more difficult. Let $\mathcal{A}$ be an arbitrary adversary playing the MaoSimAdap$_\ell$ game and let $\mathcal{A}'$ be the corresponding adversary playing the ObvSimAdap$_\ell$ game. Adversary $\mathcal{A}'$ imitates the behavior of adversary $\mathcal{A}$. Because $\mathcal{G}$ is an obv.sim.adap$_\ell$ secure garbling scheme, there is a simulator $\mathcal{S}'$ such that it makes the advantage of $\mathcal{A}'$ is negligible. Our aim is to construct a simulator $\mathcal{S}$ for the MaoSimAdap$_\ell$ game that makes the advantage of $\mathcal{A}$ negligible.

On input $\left(1^k, \Phi(f)\right)$ the simulator $\mathcal{S}'$ simply calls $\mathcal{S}$ on the same input. On input y the simulator $\mathcal{S}'$ simply omits y and calls $\mathcal{S}()$ to get X. Now, the advantage of both adversaries is the same because both simulators and both adversaries behave identically. Now, since $\mathcal{A}'$ has a negligible advantage in its ObvSimAdap$_\ell$ game and the advantage of $\mathcal{A}$ in game MaoSimApad$_\ell$ is the same as the advantage of $\mathcal{A}'$, the advantage of $\mathcal{A}$ is negligible. □

Bellare et al. prove in [3] that prv.sim.adap$_\ell$ security implies prv.ind.adap$_\ell$ security, but the converse implication does not hold, even if we require (Φ, ev) to be *efficiently invertible*. This means, that if given $\Phi(f)$ and $y = \mathtt{ev}(f, x)$, one can find f' and x' such that $\Phi(f) = \Phi(f')$ and $\mathtt{ev}(f, x) = \mathtt{ev}(f', x')$ in polynomial time. However, the following proposition shows, that prv.ind.adap$_\ell$ security implies prv.sim.adap$_\ell$ security if we define *efficiently invertible* in slightly different way.

Definition. We say that $(\Phi, \mathtt{ev})$ is componentwise efficiently invertible, if there exists a polynomial time inverter M such that

1) first finds f' such that $\Phi(f') = \Phi(f)$ using only $\Phi(f)$,
2) then given $\mathtt{ev}(f, x)$ finds x' such that $\mathtt{ev}(f', x') = \mathtt{ev}(f, x)$.

Example. Let $\Phi = \Phi_{\text{size}}$. Let n be the length of x and m be the length of $y = \mathtt{ev}^*_{\text{circ}}(f, x)$. In addition, let q be the number of gates in circuit f. The evaluation function $\mathtt{ev}^*_{\text{circ}}(f, x)$ first counts the length of x, and computes $y = \mathtt{ev}_{\text{circ}}(f, x)$. If $n < m$, then $\mathtt{ev}^*_{\text{circ}}$ takes the first n bits of y and returns them. Otherwise $\mathtt{ev}^*_{\text{circ}}$ returns y. We define a machine M that tries to invert $(\Phi, \mathtt{ev}^*_{\text{circ}})$ efficiently as follows.

1) Machine M first chooses f' to be an identity function to $r = \min\{n, m\}$ first incoming wires. Then M creates q gates, and uses the incoming wires to create some arbitrary functionality that does not affect the r output bits. In this way, M ensures that condition $\Phi_{\text{size}}(f') = \Phi_{\text{size}}(f)$ is satisfied.
2) Then given $y = \mathtt{ev}^*_{\text{circ}}(f, x)$, M has to find x' such that

$$y' = \mathtt{ev}^*_{\text{circ}}(f', x') = y.$$

M sets $x' = y$ if $n < m$. Otherwise, M finds x' by appending arbitrary $n - m$ missing input bits at the end of y.

Machine M finds f', x' satisfying the conditions in polynomial time, which makes M a componentwise efficient $(\Phi_{\text{size}}, \mathtt{ev}^*_{\text{circ}})$-inverter.

Above definition gives a condition under which prv.ind.adap$_\ell$ security implies prv.sim.adap$_\ell$ security.

Theorem 7. *Let $\mathcal{G} \in (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev})$ be a* prv.ind.adap$_\ell$ *secure garbling scheme with $\ell \in \mathbb{N}$. Then $\mathcal{G}$ is also a* prv.sim.adap$_\ell$ *secure over Φ if $(\Phi, \mathtt{ev})$ is componentwise efficiently invertible.*

PROOF. Suppose that $\mathcal{G}$ is a prv.ind.adap$_\ell$ secure garbling scheme and assume that $\mathcal{B}$ is an adversary playing the PrvSimAdap$_\ell$ game. Let machine M be a componentwise efficient $(\Phi, \mathtt{ev})$-inverter. We construct another adversary $\mathcal{A}$ playing the PrvIndAdap$_\ell$ game. Adversary $\mathcal{A}$ uses $\mathcal{B}$ as its subroutine. We define the simulator $\mathcal{S}$ in PrvSimAdap$_\ell$ game as follows. On input $\big(1^k, \Phi(f)\big)$ the simulator lets $f' \leftarrow M\big(\Phi(f)\big)$ and then $(F, e, d) \leftarrow \mathtt{Gb}(1^k, f')$. Finally the simulator returns (F, d). On input y the simulator lets $x' \leftarrow M(y)$ and returns $X = En(e, x')$.

The PrvIndAdap$_\ell$ game starts when $\mathcal{A}$ is asked to send its input to `GARBLE` procedure. $\mathcal{A}$ does not send the input yet, instead it tells the adversary $\mathcal{B}$ to start its emulated PrvSimAdap$_\ell$ game. $\mathcal{B}$ presumes to play the real PrvSimAdap$_\ell$ and sends f to $\mathcal{A}$. Adversary $\mathcal{A}$ lets $f_1 \leftarrow f$ and $f_0 \leftarrow M\big(\Phi(f)\big)$. Then $\mathcal{A}$ sends (f_0, f_1) to its `GARBLE` and gets (F, d) as return because $\Phi(f_0) = \Phi(f_1)$. $\mathcal{A}$ forwards (F, d) to $\mathcal{B}$. Now, $\mathcal{B}$ starts sending its arguments x to $\mathcal{A}$ which may happen at most ℓ times. Every time $\mathcal{A}$ receives a new argument x it lets $x_1 \leftarrow x$ and uses machine M to create another argument $x_0 \leftarrow M(y)$ where $y = \mathtt{ev}(f, x)$. Then $\mathcal{A}$ sends (x_0, x_1) to its `INPUT` procedure getting X as return. $\mathcal{A}$ forwards this to $\mathcal{B}$ who either queries a new argument or answers the challenge with b'. $\mathcal{A}$ does the same, and if $\mathcal{B}$ answered the challenge, so does $\mathcal{A}$ with the same answer.

Let us now consider the possible outcomes of the game. If $b = 1$ then the actual function f is garbled. In PrvSimAdap$_\ell$ game this corresponds to the situation where the challenge bit has been chosen as 1. If $b = 0$ then in the game presented in this proof function f_0 becomes garbled. However, this garbling seems as it was created by $\mathcal{S}$ for adversary $\mathcal{B}$. Therefore, in a PrvSimAdap$_\ell$ game this would correspond to the situation in which the challenge bit was chosen as 0.

Let c represent the challenge bit in PrvSimAdap$_\ell$ game. Because adversaries $\mathcal{A}$ and $\mathcal{B}$ give the same answers to the challenge the win probabilities are as follows.

$$\Pr[\mathcal{A} \text{ wins} \mid b = 0] = \Pr[\mathcal{B} \text{ wins} \mid c = 0]$$

$$\Pr[\mathcal{A} \text{ wins} \mid b = 1] = \Pr[\mathcal{B} \text{ wins} \mid c = 1]$$

From these we obtain

$$\Pr[\mathcal{A} \text{ wins}] = \frac{1}{2}\Pr[\mathcal{A} \text{ wins} \mid b = 0] + \frac{1}{2}\Pr[\mathcal{A} \text{ wins} \mid b = 1]$$

$$= \frac{1}{2}\Pr[\mathcal{B} \text{ wins} \mid c = 0] + \frac{1}{2}\Pr[\mathcal{B} \text{ wins} \mid c = 1] = \Pr[\mathcal{B} \text{ wins}]$$

This lets us conclude that the advantages for both adversaries are equal as well. Our assumption implies that we assumed $\mathcal{A}$'s advantage must be

negligible, so must $\mathcal{B}$'s advantage as well be negligible against the constructed simulator $\mathcal{S}$. □

The results presented above together with the results in [3] give us a hierarchy for classes of adaptive garbling schemes. It appears to be similar to the hierarchy for classes of static garbling schemes (compare Fig. 3 to Fig. 3 in [15, p. 16]).

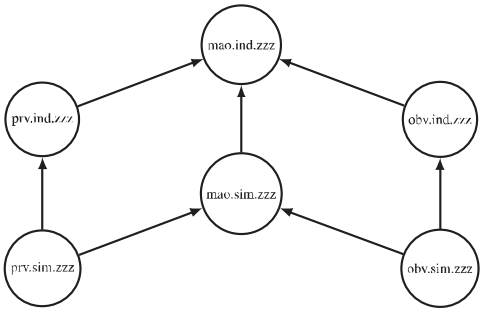


Fig. 3. Hierarchy of adaptive security classes for zzz $\in \{\text{stat}, \text{adap}_\ell\}$

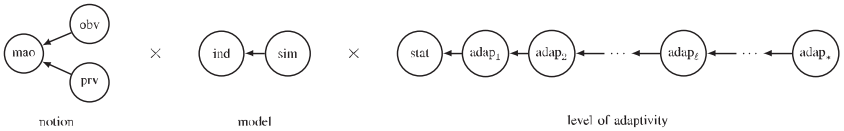


Fig. 4. Hierarchy of garbling scheme classes represented as a Cartesian product of graphs

VI. Hierarchy of garbling schemes

We have now completed describing the hierarchy for different types of security classes of garbling schemes. We have obtained a hierarchy that is similar for static and adaptive classes with a fixed threshold value ℓ (see Fig. 3). The result from Theorem 1 tells that we can combine the hierarchies for adaptive security classes with different threshold values ℓ into one common hierarchy. Moreover, the adaptive classes for threshold value $\ell = 1$ are included in the corresponding static classes, so also the hierarchy of static security classes can be included in the extended hierarchy.

To illustrate the hierarchy, we may use the notion of *directed graph Cartesian product*. By directed graph Cartesian product $G_1 \times G_2$ we mean the

directed graph having vertices in $V = V_1 \times V_2$. The directed edges of $G_1 \times G_2$ are found as follows. There is a directed edge from vertex $u = (u_1, u_2)$ to $v = (v_1, v_2)$ in $G_1 \times G_2$ whenever $u_1 = v_1$ and there is a directed edge from u_2 to v_2, or $u_2 = v_2$ and there is a directed edge from u_1 to v_1.

Using the graph Cartesian product, we can describe the hierarchy of security classes of garbling schemes as an Cartesian product of security notion $\{\mathrm{prv}, \mathrm{obv}, \mathrm{mao}\}$, security model $\{\mathrm{ind}, \mathrm{sim}\}$ and the level of adaptivity $\{\mathrm{stat}, \mathrm{adap}\}$ with the subscript ℓ telling how many times the `INPUT` procedure can be called for the same input. The hierarchy is shown in Fig. 4.

VII. Conclusions

In this paper, we have introduced new classes of garbling schemes by generalizing the adaptivity notion introduced by Bellare et al. in [3] by allowing the adversary to use the same garbling more than once. This is motivated by the fact that there exists a garbling scheme that allows the same garbled circuit to be used arbitrarily many times [9]. However, the construction is still impractical. On the other hand, we believe that there are applications in which it is sufficient to use the same garbled circuit only a limited number of times instead of arbitrarily many times, but still so many times that computing a new garbled circuit each time for the same function becomes impractical. Therefore, it makes sense to define classes where the adversary is allowed to use the same garbling at most ℓ times.

This paper investigates how these new security classes allowing the ℓ times usage of the same garbling are related to the existing classes. Our central result is the infinite hierarchy of garbling schemes presented in Fig. 4.

The two first theorems in this paper are essential in proving the infiniteness of the hierarchy. First of all, a security class with threshold value $\ell + 1$ is properly included in the corresponding class with threshold value ℓ, unless both classes are empty. Secondly, a security class allowing the same garbling to be used arbitrarily many times ($*$-class) is properly included in the corresponding security class with any threshold value $\ell \in \mathbb{N}$ (unless both are empty). Thirdly, the class allowing the same garbling to be used arbitrarily many times seems to be non-empty at least in some cases. These three facts together let us conclude that at least in some cases the hierarchy is proper and infinite.

Showing how the classes are related with each other with different threshold values ℓ is a waypoint towards the main goal – an extended hierarchy for classes of garbling schemes. As another waypoint we proved that adaptive security classes for a fixed ℓ have a hierarchy identical to the hierarchy of static security classes shown in Fig. 3.

These two results, the infiniteness of the hierarchy as well as the hierarchy for a specific threshold value ℓ, allows us to present the extended

hierarchy shown in Fig. 4. An area of future research would be to extend the results to the fine-grained security notions of [3]. Another interesting task would be to construct a practical garbling scheme that allows a limited usage of the same garbled circuit.

REFERENCES

[1] Beaver, D., Micali, S. and Rogaway, P., The round complexity of secure protocols, *Proc. of the* 22nd *STOC,* (1990), pp. 503–513.

[2] Bellare, M., Hoang, V. T., Keelveedhi, S. and Rogaway, P., Efficient garbling from a fixed-key blockcipher, *Proc. of Symposium on Security and Privacy 2013,* (2013), pp. 478–492.

[3] Bellare, M., Hoang, V. T. and Rogaway, P., Adaptively secure garbling scheme with applications to one-time programs and secure outsourcing, *Asiacrypt 2012,* 7685 of LNCS (2012), 134–153.

[4] Bellare, M., Hoang, V. T. and Rogaway, P., Foundations of garbled circuits, in: *Proc. of ACM Computer and Communications Security (CCS'12),* ACM (2012), pp. 784–796.

[5] Bellare, M. and Rogaway, P., Code-based game-playing proofs and the security of triple encryption, *Advances in Cryptology, Proc. of Eurocrypt 2006,* 4004 of LNCS (2006), 409–426.

[6] Boneh, D., Sahai, A. and Waters, B., Functional encryption: Definitions and challenges, *Proc. of TCC 2011,* 6597 of LNCS (2011), 253–273.

[7] Gennaro, R., Gentry, C. and Parno, B., Non-interactive verfiable computing: Outsourcing computation to untrusted workers, *CRYPTO 2010,* 6223 of LNCS (2010), 465–482.

[8] Gentry, C., *A Fully Homomorphic Encryption Scheme,* PhD thesis, Stanford University (2009). `crypto.stanford.edu/craig`.

[9] Goldwasser, S., Kalai, Y., Popa, R. A., Vaikuntanathan, V. and Zeldovich, N., Reusable garbled circuits and succinct functional encryption, *Proc. of the* 45th *STOC,* (2013), pp. 555–564.

[10] Goldwasser, S., Kalai, Y. and Rothblum, G., One-time programs, *CRYPTO 2008,* 5157 of LNCS (2008), 39–56.

[11] Gorbunov, S., Vaikuntanathan, V. and Wee, H., Attribute-based encryption for circuits, *Proc. of the* 45th *STOC* (2013), pp. 545–554.

[12] Goyal, V., Pandey, O., Sahai, A. and Waters, B., Attribute-based encryption for fine-grained access control of encrypted data, in: *Proc. of ACM Computer and Communications Security (CCS'06),* ACM (2006), pp. 89–98.

[13] Hoang, V. T., *Foundations of Garbled Circuits,* PhD thesis, University of California, Davis (2013).

[14] Lindell, Y. and Pinkas, B., A proof of security of Yao's protocol for secure two-party computation, *Journal of Cryptology,* **22(2)** (2009), 161–188.

[15] Meskanen, T., Niemi, V. and Nieminen, N., Classes of garbled schemes, *Infocommunications Journal,* **V(3)** (2013), 8–16.

[16] Peikert, C., Public-key cryptosystems from the worst-case shortest vector problem, *Proc. of the* 41st *STOC* (2009), pp. 333–342.

[17] Regev, O., On lattices, learning with errors, random linear codes, and cryptography, *Proc. of the* 37th *STOC* (2005), pp. 84–93.

[18] Yao, A., How to generate and exchange secrets, *Proc. of* 27th *FOCS, 1986* (1986), pp. 162–167.

Paper III

Hierarchy for Classes of Projective Garbling Schemes

T. Meskanen and V. Niemi and N. Nieminen (2014). In *International Conference on Information and Communications Technologies (ICT 2014)*, pages 1-8. IEEE

HIERARCHY FOR CLASSES OF PROJECTIVE GARBLING SCHEMES

Tommi Meskanen[*], Valtteri Niemi[*, §], Noora Nieminen[*]

[*]Department of Mathematics and Statistics, University of Turku, 20014 Turun yliopisto, FINLAND
§ Xidian University, Xi'an, CHINA

[*]{tommes, pevani, nmniem}@utu.fi

Keywords: garbled circuits, garbling schemes, secure multiparty computations, adaptive security, projectivity

Abstract

Recently, privacy has become one of the hottest topics in the world of the Internet-based communications and computation. Many applications, such as cloud computing and various cloud services, require information security and privacy. However, in many cases the level of security in current solutions is insufficient. Methods to ensure secure multiparty computation have been studied for decades, but because of the needs of the modern computation this topic has recently become increasingly popular. Yao's garbled circuit is one method to compute a function f on an argument x privately without revealing any information about f and x except $f(x)$. Unfortunately, current constructions for garbled circuits have limitations - either efficient construction at the cost of reusability or reusability at the cost of efficiency. In this paper we aim at somewhere between these two extremes by defining new security measures for garbling schemes. These definitions support design of reusable garbled circuits that achieve adaptive security even when the function argument is handled bitwise. Moreover, we show how the new classes are related to the existing security classes of garbling schemes. As a result, we obtain an extended hierarchy for the classes of garbling schemes.

1 Introduction

The technique of garbling circuits originates from Yao's seminal paper [15] after which garbled circuits have been used for various purposes even though exact measures for security had not been defined yet. The first step towards formal security definitions was taken by Lindell and Pinkas who gave the first proof of security for a custom protocol using garbled circuits in [10]. The next step was taken by Bellare, Hoang and Rogaway who proposed the first definition of garbling scheme, at the same time elevating the garbled circuits from a mere cryptographic technique into a cryptographic primitive by introducing several notions for security in [2] including the notions *privacy* and *obliviousness*. Bellare et al. also introduced two models for each notion, *indistinguishability-based* and *simulation-based* model. Afterwards, Bellare et al. have extended the definitions in order to achieve the needs of modern multiparty computations including one-time programs and secure outsourcing. In [1], they introduce the concept of *adaptive security* of a garbling scheme and two levels of adaptivity, *coarse-grained* and *fine-grained*. Descriptions of the classes of garbling schemes and their relations can be found from Hoang's PhD thesis [9].

The different security notions allow garbling schemes to be categorized in different classes. The relations between the classes have been investigated not only by Bellare et al. in [1, 2] but also by Meskanen, Niemi and Nieminen in [11, 12]. Meskanen et al. introduced a new security notion, *ModI*, and included the classes for this new notion in the hierarchy presented in [11]. Meskanen et al. also generalized the notion of adaptive security in [12] by allowing non-deterministic encryption as well as introducing a new parameter telling how many times the same garbled function can be reused. In the same paper, they also extended the hierarchy of [11] by including many new classes of garbling schemes.

As mentioned, adaptive security notions are important from the practical point of view. Another important aspect of practicality of garbled circuits is their reusability. The notions introduced by Bellare et al. support only single-use garbled circuits. If a user liked to evaluate the same function on different arguments, the user should compute a new garbled circuit every time. This kind of regarbling becomes soon inefficient. This deficiency has been tackled in a recent paper [6] from Goldwasser, Kalai, Popa, Vaikuntanathan and Zeldovich in which a technique to construct reusable garbled circuits is introduced. This construction is based on functional encryption [4] which in turn has used fully-homomorphic encryption (FHE) [5] and attribute-based encryption(ABE) [7, 8] as its building blocks. However, constructing a reusable garbled circuit in this way is impractical due to the impracticality of FHE. Moreover, the existence of reusable garbled circuits relies on Learning With Errors assumption (see [13, 14] on more information about this assumption).

An attempt to support design of both (somewhat) reusable and adaptive garbled circuits was initiated in [12]. In this paper we continue on the same topic by extending the hierarchy for classes of garbling schemes even further by introducing at least moderately reusable garbled circuits that provide fine-grained adaptive security. The extended hierarchy contains the classes from [1, 2, 11, 12] and new

classes presented in this paper. We start by defining the central concepts in Section 2. In Section 3 we define the new adaptive security classes. Section 4 is a collection of results about the relations between the new classes of garbling schemes. The extended hierarchy is presented in Section 5.

2 Definitions

This section contains the basic definitions and notations. Let us start with a concept of a *string* which is a finite sequence of bits. Let S be a finite set. Notation $y \leftarrow S$ means that an element is selected uniformly at random from the set S, and this element is assigned as a value to the variable y. If A denotes an algorithm, then notation $A(x_1, \dots, x_n)$ refers to the output of the algorithm A on inputs $x_1, \dots, x_n$. We say that a function $f: \mathbb{N} \to \mathbb{R}$ is negligible if for every $c > 0$ there is an integer N_c such that $|f(n)| < n^{-c}$ for all $n > N_c$.

2.1 Code-based games

The proofs in this paper are based on *code-based games*. Following the terminology presented in [3], a game is a collection of procedures containing three types of procedures: INITIALIZE, FINALIZE and other named procedures. We assume throughout this paper that procedures INITIALIZE and FINALIZE are compulsory in a game, whereas other procedures are optional.

The entity playing a game is called *an adversary*. Every game starts with INITIALIZE procedure. Then the adversary may invoke other procedures before feeding its output to the FINALIZE procedure. Based on this input from the adversary FINALIZE procedure creates a string that tells the outcome from the game typically consisting of one bit of information: whether the adversary has won or not.

2.2 Garbling scheme

Formally, a garbling scheme is a 5-tuple $\mathcal{G} = (Gb, En, De, Ev, ev)$ of algorithms, where algorithms Gb and En are probabilistic and the rest are deterministic. Let f denote the string representing the original function. Function f describes the evaluation function $ev(f,\cdot)\colon \{0,1\}^n \to \{0,1\}^m$ which is to be garbled. For example, if f is an encoding of a circuit then $ev(f,\cdot)$ is some circuit evaluation function. Here, the values n and m represent the lengths of the argument x and the function value $y = ev(f,x)$. The values n and m must also be efficiently computable from f.

The first component Gb is the garbling algorithm. It takes f and 1^k, where $k \in \mathbb{N}$ is a security parameter, as inputs and returns (F, e, d) as output. String e describes the particular probabilistic encryption algorithm $En(e,\cdot,r)$ which maps an initial argument x to a garbled argument $X = En(e,x,r)$. Here, r is an additional input describing the randomness of the encryption algorithm En. String F determines the garbled function $Ev(F,\cdot)$ which returns the garbled function value $Y = Ev(F,X)$. Finally, string d describes the decryption algorithm $De(d,\cdot)$ which on a garbled function value Y returns the final function value $De(d,Y) = y = ev(f,x)$. Note that, regardless of the choice of r, En gives the correct output X in the sense that the final function value is always y. Figure 1 illustrates how a garbling scheme works.

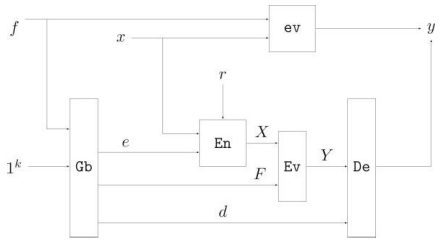


Figure 1: Idea behind garbling. The diagram shows that the final value obtained via garbling must coincide the final value obtained by direct evaluation, i.e. $\boldsymbol{ev(f,x) = y = De(d,Y)}$ where $\boldsymbol{F}$ is the garbled function, $\boldsymbol{Y = Ev(F,X)}$ the garbled value and $\boldsymbol{X = En(e,x,r)}$ the garbled argument.

2.3 Projectivity of a garbling scheme

In this paper, we concentrate on garbling schemes that allow the user to feed his argument to the function bit by bit. This sets additional requirements to the garbling scheme, namely the encryption algorithm needs to treat the argument in pieces instead of processing the whole input at once. This property is called *projectivity* which we next define more formally.

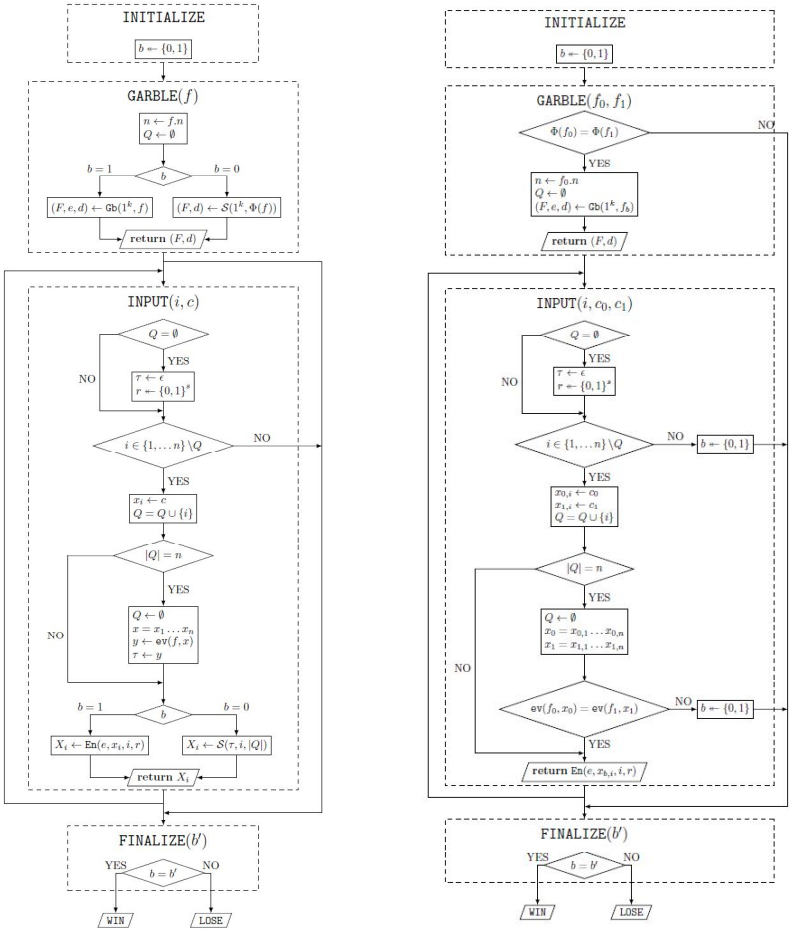
Let $\mathcal{G} = (Gb, En, De, Ev, ev)$ be a garbling scheme which is used to evaluate function f. Let $x = x_1 \dots x_n \in \{0,1\}^n$ and $x' = x'_1 \dots x'_n \in \{0,1\}^n$ be two arguments for f. Let e be the encryption key that is used to encrypt both inputs with the same randomness value r. We say that garbling scheme $\mathcal{G}$ is *projective* if two garbled arguments $X = En(e,x,r)$ and $X' = En(e,x',r)$ can be presented in form $X = (X_1, \dots, X_n)$, $X' = (X'_1, \dots, X'_n)$ in such way that the following condition holds: $x_i = x_i$, iff $X_i = X'_i$.

2.4 Side-information function

By the concept of *a side-information function*, we capture the information revealed about f by the garbling process. More formally, a side-information function Φ deterministically maps f to $\Phi(f)$. For example, in the case of computation model using circuits and the corresponding natural evaluation algorithm ev_{circ}, the value $\Phi(f)$ might be the size of the circuit that was garbled, the topology of the circuit or something else - even the whole initial circuit.

3 Classes of adaptively secure projective garbling schemes

According to [1, 2, 11, 12], garbling schemes can be organized into classes according to security notion, model and level of adaptivity. We shortly summarize these concepts before going to the definitions for the new classes of garbling schemes.



(a) (b)

Figure 2 Flowcharts for the games $\boldsymbol{PrvSimPadap_{\mathcal{E}}}$ (a) and $\boldsymbol{PrvIndPadap_{\mathcal{E}}}$ (b). The game $\boldsymbol{PrvSimPadap_{\mathcal{E}}}$ depends on the choice of $\mathcal{G}$, Φ and $\mathcal{S}$ whereas the game $\boldsymbol{PrvIndPadap_{\mathcal{E}}}$ depends on the choice of $\mathcal{G}$ and Φ.

The security model is either *indistinguishability-based* or *simulation-based*. Indistinguishability refers to the inability of distinguishing two garbled functions and their arguments from each other. Simulation-based model refers to inability of distinguishing a garbled function and its argument from a string that is created by the simulator to mimic the original garbled function and argument. The shorthand notion *ind* means that the model is indistinguishability-based whereas *sim* refers to the simulation-based model.

The security notions are *privacy*, *matchability-only* and *obliviousness*. The different security notions capture the idea of how much information is allowed to be leaked during the garbled evaluation. A garbling scheme achieving obliviousness does not allow f, x or $f(x)$ to be revealed. Matchability-only secure scheme is not allowed to leak f or x, but when computing $y_1 = f(x_1)$ and $y_2 = f(x_2)$, the scheme is allowed to reveal whether $y_1 = y_2$. Garbling schemes achieving privacy keep f and x secret, whereas $f(x)$ is allowed to be revealed. The notions are abbreviated as follows: *prv* refers to privacy, *mao* refers to matchability-only and *obv* refers to obliviousness.

The level of adaptivity is either *static* or *adaptive*. Garbling schemes that achieve static security assume that the function f and argument x are both fixed at the same time in the security game. Garbling schemes that achieve adaptive security allow the function f to be garbled before

the adversary fixes the argument x in the security game. If an adaptively secure garbling scheme is also projective, then we have the option that the argument x is not only handled separately from f but also bitwise in the security game. The adversary first fixes f and gets garbled function F. After this, the adversary fixes the argument bit by bit. Game returns garbled versions of argument bits to the adversary. The adversary gets a fully specified garbled argument only after specifying all the bits of the argument. We use abbreviation *stat* to denote static security, *adap* to denote adaptive security in general and *padap* to denote bitwise adaptive security. We also use words *coarse-grained adaptive security* when we refer to adaptive security and *fine-grained adaptive security* when we refer to bitwise adaptive security. For adaptive security we can define an additional parameter ℓ: the adversary is allowed to sequentially choose at most ℓ different values of x.

Every security class is specified by a combination of the above three components. Each combination and hence each class corresponds to its own security game. The games are named using the abbreviations given above as follows. Security game *XxxYyyZzz* refers to a game whose notion is $Xxx \in \{Prv, Mao, Obv\}$, model $Yyy \in \{Ind, Sim\}$ and level of adaptivity $Zzz \in \{Stat, Adap_\ell, Padap_\ell\}$. The corresponding security class is denoted by $GS(xxx.yyy.zzz)$ and a garbling scheme $\mathcal{G}$ in this class is said to be $xxx.yyy.zzz$ secure.

As mentioned before, in this paper we focus on projective garbling schemes achieving adaptive security. Next we describe the games which define the adaptive security of an projective garbling scheme. In fig. 2(a) we present the game that tests whether a garbling scheme achieves adaptive privacy in simulation-based model. The corresponding indistinguishability-based game is illustrated in fig. 2(b). The four other games, $MaoSimPadap_\ell$, $MaoIndPadap_\ell$, $ObvSimPadap_\ell$ and $ObvIndPadap_\ell$ are slightly different. The differences are described along the game descriptions below.

All six games consist of four procedures: INITIALIZE, GARBLE, INPUT, FINALIZE. The game always starts with INITIALIZE procedure, in which the challenge bit b is chosen randomly. The compulsory call to INITIALIZE is followed by a query to the next procedure called GARBLE which prepares garbled function F to the adversary. After this procedure an adversary calls procedure INPUT which prepares the garbled arguments to the adversary. Finally, the adversary calls FINALIZE, a procedure telling whether the adversary has won the security game. The procedures INITIALIZE and FINALIZE are the same for all six games. The two other procedures, GARBLE and INPUT, depend on the game. Next we describe these two procedures in more detail. First, we describe procedure GARBLE in simulation- and indistinguishability-based games. Along the descriptions we also describe the differences between the security notions prv, mao and obv. The procedure INPUT is explained in similar manner.

In simulation-based games, procedure GARBLE prepares the garbled function based on the challenge bit b: either the garbled function F and decryption key d are created by the garbling algorithm ($b = 1$) Gb or by a simulator $\mathcal{S}$ ($b = 0$). Finally, GARBLE returns pair (F, d) in $PrvSimPadap_\ell$ game, whereas in $MaoSimPadap_\ell$ and $ObvSimPadap_\ell$ games the procedure returns F without d.

In indistinguishability-based games, procedure GARBLE takes two functions, f_0 and f_1 as its inputs. First, the procedure checks whether the two functions are compatible with each other, i.e. whether $\Phi(f_0) = \Phi(f_1)$. If the functions do not share the same side-information, then procedure GARBLE is exited without no return value. The adversary proceeds directly to FINALIZE and is forced to guess the challenge bit b. Otherwise, the GARBLE procedure prepares a garbled function based on the challenge bit, i.e. returns the garbled version of f_b. In the game $PrvIndPadap_\ell$ the adversary additionally gets the decryption key d, while in the two other games the adversary doesn't get d.

The INPUT procedure in simulation-based games takes an index i and the i^{th} bit of argument x as its input. First the procedure checks whether the index i is already processed. If i^{th} bit of the argument has been processed earlier, then an adversary playing the game does not receive the garbled argument bit and the procedure is exited. The adversary is forced to proceed directly to FINALIZE and has to give the answer based on the output from GARBLE procedure. On the other hand, if i has not been processed earlier, then the procedure creates a garbling of the argument bit based on the choice of the challenge bit: if $b = 1$ then X_i is the encryption of x_i, and if $b = 0$ then X_i is created by the simulator. Every time the whole argument $x = x_1 \dots x_n$ becomes specified, the list of processed indices Q is emptied for the next round. In games $\{Prv, Mao\}SimPadap_\ell$ the simulator is allowed to use $y = ev(f, x)$ to create the garbling of the last missing argument bit whereas in $ObvSimPadap_\ell$ game it is not allowed to know $y = ev(f, x)$.

The procedure INPUT in indistinguishability-based games takes index i as well as the i^{th} bit of arguments x_0 and x_1 as its inputs. The procedure first checks that index i has not yet been processed. If it has been processed, the procedure ends and the adversary proceeds to FINALIZE after a new challenge bit is chosen. This is done to ascertain that the adversary would not get excessive advantage by knowing F (and d) for such functions that are doomed to fail the other tests in INPUT procedure. Namely, chosing functions f_0 and f_1 wisely (F_0 and F_1 are distinguishable) and purposefully failing in INPUT the adversary is always able to win the game. If the index i had not yet been processed, the procedure proceeds to the next check. If the entire argument becomes specified, meaning that $|Q| = n$, then Q is emptied in all three games. In games $\{Prv, Mao\}IndPadap_\ell$ there is an additional check $ev(f_0, x_0) \stackrel{?}{=} ev(f_1, x_1)$. If this check is not passed, the

procedure is over and the adversary needs to guess the challenge bit based on the information it is already possessing. However, also in this case the challenge bit is chosen newly and knowing F (or F and d) does not really help the adversary to win the game. If the check is passed or $|Q| < n$, INPUT procedure prepares a garbling $X_{b,i}$ of the argument bit $x_{b,i}$ to be returned to the adversary.

Procedure INPUT may be called several times. However, in every game there is *a threshold* for the maximum number of INPUT queries: The parameter ℓ in all games describes how many times the adversary is allowed to get a fully specified garbled argument by calling INPUT procedure. In other words, the adversary is allowed to call INPUT at most $n \cdot \ell$ times in total.

In both simulation- and indistinguishability-based games the adversary needs to predict the challenge bit chosen by the game. In simulation-based game this means that the adversary tries to predict, based on the information from procedures GARBLE and INPUT, whether F and the various garbled argument bits were created by the actual garbling algorithm or were they devised by a simulator. In indistinguishability-based game the task of the adversary is to determine which of the functions, f_0 or f_1, got garbled based on the information it has received during the game. The procedure FINALIZE tells whether the adversary answered the correct challenge bit. If the adversary answered correctly, it wins the security game.

An adversary playing a security game for an adaptively secure projective garbling scheme has a certain probability of winning a game. If the information gathered during the game makes the win probability of the adversary greater than $\frac{1}{2}$ then the adversary has an *advantage* over pure guessing. This gives the intuition behind the concept of advantage, a formal definition is given below.

Definition (*Advantage of an adversary*) Let adversary $\mathcal{A}$ be playing the adaptive game $XxxYyyPadap_\ell$ where $Xxx \in \{Prv, Mao, Obv\}$ and $Yyy \in \{Ind, Sim\}$. The advantage of adversary $\mathcal{A}$ in this game is

$$Adv(\mathcal{A}, k) = 2 \cdot Pr[\mathcal{A}\ wins\ XxxYyyPadap_\ell\ game] - 1.$$

The advantage is a measure that is used to assess the security of a garbling scheme. The definition of adaptive security of a projective garbling scheme is given below.

Definition (*Fine-grained adaptive security of a garbling scheme*) We say that a garbling scheme $\mathcal{G} = (Gb, En, De, Ev, ev)$ is $xxx.ind.padap_\ell$ secure over Φ, if for any polynomial time (PT) adversary $\mathcal{A}$ the advantage $Adv(\mathcal{A},\cdot)$ is negligible in $XxxIndPadap_\ell$ game. Respectively, $\mathcal{G}$ is $xxx.sim.padap_\ell$ secure if for any PT adversary $\mathcal{A}$ there exists a PT simulator $\mathcal{S}$ such that $Adv(\mathcal{A},\cdot)$ is negligible in $XxxSimPadap_\ell$ game.

4 Relations between the adaptive security classes of projective garbling schemes

In this section we consider relations between the adaptive security classes. First we study how the fine-grained adaptive security classes are related to each other when varying the threshold value ℓ. Then we consider the relation between the fine-grained and coarse-grained adaptive security - we show that fine-grained security implies coarse-grained security but not vice versa. After that we consider the relations between the fine-grained security classes for a fixed threshold value ℓ. We show how different models and notions are related to each other.

Theorem 1 Let $\mathcal{F}_\ell$ be a class of adaptively secure projective garbling schemes, i.e. $\mathcal{F}_\ell = GS(xxx.yyy.padap_\ell, \Phi)$ for $xxx \in \{prv, mao, obv\}$, $yyy \in \{sim, ind\}$ and $\ell \in \mathbb{N}$. Then $\mathcal{F}_\ell \subsetneq \mathcal{F}_{\ell+1}$ or $\mathcal{F}_{\ell+1} = \mathcal{F}_\ell = \emptyset$.

Proof: The inclusion $\mathcal{F}_{\ell+1} \subseteq \mathcal{F}_\ell$ is obvious, because the adversary playing the game related to class $\mathcal{F}_{\ell+1}$ can always use only $n \cdot \ell$ INPUT calls instead of the maximum allowed number of calls which in this case is $n \cdot (\ell + 1)$.

Next we need to prove that under the condition $\mathcal{F}_\ell \neq \emptyset$ there exists a garbling scheme that belongs to class $\mathcal{F}_\ell$ but does not belong to class $\mathcal{F}_{\ell+1}$. Let $\mathcal{G} = (Gb, En, De, Ev, ev)$ be a garbling scheme in $\mathcal{F}_\ell$. We use Gb, En and Ev of garbling scheme $\mathcal{G}$ and modify them to construct another garbling scheme $\mathcal{G}' = (Gb', En', De, Ev', ev)$ in the following way.

The modified encrypting algorithm En' consists of three parts. The first part contains the original encrypting algorithm En from garbling scheme $\mathcal{G}$. The second part is a completely independent symmetric non-deterministic secure encryption scheme $\widehat{En}$ that is used to encrypt x_i with an independent key $\hat{e}$. Finally, the third part is a secret sharing scheme that creates t shares of key $\hat{e}$ in such a way that $n \cdot \ell + 1$ shares are needed to reconstruct the key $\hat{e}$. The garbled argument bit X_i' is obtained with algorithm En' as follows: $En'\big((e, \hat{e}, t\ shares), x_i\big) = (En(e, x_i), \widehat{En}(\hat{e}, x_i), s) = (X_i, \hat{X}_i, s)$ where s is a random share of $\hat{e}$.

The garbling algorithm Gb' creates (F, e, d) in the similar manner as Gb would if the scheme $\mathcal{G}$ had been used. In addition, it has to create $\hat{e}$ by taking the security parameter k into account (the encryption with $\hat{e}$ cannot be breakable in polynomial time with respect to k). Moreover, Gb' creates t shares of $\hat{e}$. The output of algorithm Gb' is $((F, \hat{F}), (e, \hat{e}, t\ shares\ of\ key\ \mathrm{e}), d)$ where F is the garbled function and $\hat{F} = \widehat{En}(\hat{e}, f)$.

The garbled evaluation algorithm Ev' takes $((F, \hat{F}), (X, \hat{X}, s))$ as its inputs. Here $X = X_1 \ldots X_n$ is the fully specified garbled input which is obtained by using En of garbling scheme $\mathcal{G}$ whereas $\hat{X} = \widehat{X_1} \ldots \widehat{X_n}$ is the fully specified argument encrypted with the independent key $\hat{e}$. Finally, $s = \{s_1, \ldots s_n\}$ is the set of n random shares of

secret $\hat{e}$. The algorithm omits the new parts $\hat{F}, \hat{X}, s$ and computes $Y = Ev'((F, \hat{F}), (X, \hat{X}, s)) = Ev(F, X)$.

The constructed garbling scheme $\mathcal{G}'$ belongs to class $\mathcal{F}_\ell$ because any adversary playing a game with at most $n \cdot \ell$ INPUT calls obtains at most $n \cdot \ell$ random shares of key $\hat{e}$ which are not useful in reconstructing the key and hence in discovering x_i and f. To prove that $\mathcal{G} \notin \mathcal{F}_{\ell+1}$, consider an arbitrary adversary who is allowed to make $n \cdot (\ell + 1)$ INPUT queries in its game. This adversary gets $n \cdot (\ell + 1) = n \cdot \ell + n$ shares of $\hat{e}$ and there is a certain positive probability not depending on k that these shares can be used to reconstruct $\hat{e}$. As a consequence, the adversary has a chance to discover f and x and using that information the adversary would be able to find the challenge bit b and win the game. This in turn leads to a non-negligible advantage of the adversary. This shows that $\mathcal{G}'$ cannot belong to security class $\mathcal{F}_{\ell+1}$. ■

Definition Let $xxx \in \{prv, mao, obv\}$ and $yyy \in \{ind, sim\}$. We define a new class $\mathcal{F}_* = GS(xxx.yyy.padap_*, \Phi)$ which is the class of secure garbling schemes when the adversary is allowed to call the INPUT procedure arbitrarily many times in the $xxxYyyPadap$ game.

Theorem 2 Let us again denote $\mathcal{F}_\ell = GS(xxx.yyy.padap_\ell, \Phi)$ for $xxx \in \{prv, mao, obv\}$ and $yyy \in \{ind, sim\}$. Then we have the following inclusion:

$$\mathcal{F}_* \subsetneq \bigcap_{\ell=1}^{\infty} \mathcal{F}_\ell \quad \text{or} \quad \mathcal{F}_* = \bigcap_{\ell=1}^{\infty} \mathcal{F}_\ell = \emptyset$$

Proof: Omitted in this version due to lack of space. ■

Theorem 3 Let $xxx \in \{prv, mao, obv\}$ and $yyy \in \{ind, sim\}$. Let $\mathcal{F}_\ell = GS(xxx.yyy.padap_\ell)$ be a class of adaptively secure projective garbling schemes and $\mathcal{C}_\ell = GS(xxx.yyy.adap_\ell)$ the corresponding class of (not necessarily projective) adaptively secure garbling schemes. Then, $\mathcal{F}_\ell \subsetneq \mathcal{C}_\ell$ or $\mathcal{F}_\ell \subseteq \mathcal{C}_\ell \subseteq GS(proj)$, where $GS(proj)$ is the class of projective garbling schemes.

Proof: Omitted in this version due to lack of space. ■

Theorem 4 Let again $xxx \in \{prv, mao, obv\}$ and $yyy \in \{ind, sim\}$. Let $\mathcal{F}_\ell = GS(xxx.yyy.padap_\ell)$ be a class of adaptively secure projective garbling schemes and $\mathcal{C}_{\ell+1} = GS(xxx.yyy.adap_{\ell+1})$ the corresponding class of adaptive garbling schemes. Then $\mathcal{F}_\ell \not\subseteq \mathcal{C}_{\ell+1}$ or $\mathcal{F}_\ell = \mathcal{C}_{\ell+1} = \emptyset$.

Proof: Omitted in this version due to lack of space. ■

Theorem 5 We have the following inclusions for any $\ell \in \mathbb{N}$:

1. $GS(prv.sim.padap_\ell, \Phi) \subseteq GS(prv.ind.padap_\ell, \Phi)$
2. $GS(mao.sim.padap_\ell, \Phi) \subseteq GS(mao.ind.padap_\ell, \Phi)$
3. $GS(obv.sim.padap_\ell, \Phi) \subseteq GS(obv.ind.padap_\ell, \Phi)$

Proof: Suppose that $\mathcal{G}$ is a garbling scheme in class $GS(xxx.sim.padap_\ell, \Phi)$ where $xxx \in \{prv, obv, mao\}$. Our aim is to prove that $\mathcal{G} \in GS(xxx.ind.padap_\ell, \Phi)$. Let $\mathcal{A}$ be an adversary playing the $xxxIndPadap_\ell$ game and let us construct a PT adversary for the $xxxSimPadap_\ell$ game. Let $\mathcal{B}$ run $\mathcal{A}$ as a subroutine.

Procedure INITIALIZE starts the game $XxxSimPadap_\ell$ game, telling the adversary $\mathcal{B}$ to prepare f to be given to GARBLE procedure. However, $\mathcal{B}$ does not immediately send a function f of its choice to GARBLE, instead it tells adversary $\mathcal{A}$ to start the $XxxIndPadap_\ell$ game. $\mathcal{A}$ presumes to play a real $XxxIndPadap_\ell$ game and sends its GARBLE input f_0, f_1 to $\mathcal{B}$. If $\Phi(f_0) \neq \Phi(f_1)$ $\mathcal{B}$ tells $\mathcal{A}$ that GARBLE procedure has been finished without an output. $\mathcal{B}$ lets $d \leftarrow \{0,1\}$ and sends f_d to its GARBLE regardless of what was the result of $\Phi(f_0) =^? \Phi(f_1)$. From this point on, the games for different notions progress slightly differently. Figure 3 illustrates the case where $\mathcal{B}$ plays $PrvSimPadap_\ell$ game and $\mathcal{A}$ presumes to play $PrvIndPadap_\ell$ game which actually is mimicked by $\mathcal{B}$. Cases (ii) and (iii) follow the same idea.

In Prv game, $\mathcal{B}$ gets (F, d) as a return from GARBLE. $\mathcal{B}$ sends (F, d) to $\mathcal{A}$ if $\Phi(f_0) = \Phi(f_1)$. If $\Phi(f_0) \neq \Phi(f_1)$, $\mathcal{A}$ proceeds to FINALIZE and so does $\mathcal{B}$. Otherwise, $\mathcal{A}$ starts sending its arguments x_0, x_1 to $\mathcal{B}$ bit by bit, thus sending (i, c_0, c_1) to $\mathcal{B}$. Here i tells the index of the bit, c_0 is the i^{th} bit of argument x_0 and c_1 is the i^{th} bit of argument x_1. First, $\mathcal{B}$ performs the check $i \in^? \{1, \dots, n\} \setminus Q$. If $\mathcal{A}$ fails the test, $\mathcal{B}$ tells $\mathcal{A}$ that the query for (i, c_0, c_1) has failed, the procedure is exited and no output is provided to $\mathcal{A}$. Adversary $\mathcal{A}$ is now forced to proceed to FINALIZE. Otherwise, $\mathcal{B}$ proceeds to the next check $|Q| =^? n$. If $|Q| < n$, $\mathcal{B}$ calls its own INPUT with (i, c_w) getting X_i as an output. $\mathcal{B}$ sends X_i to $\mathcal{A}$. On the other hand, if $|Q| = n$ then $\mathcal{B}$ first empties Q, chooses the new randomness value r and collects all the bits for x_0 and x_1. $\mathcal{B}$ then checks whether $ev(f_0, x_0) = ev(f_1, x_1)$. If $ev(f_0, x_0) \neq ev(f_1, x_1)$ then $\mathcal{B}$ tells to $\mathcal{A}$ that INPUT procedure is exited with no output. Regardless of possible failure of $\mathcal{A}$, $\mathcal{B}$ sends (c_w, i) to its own INPUT getting X_i as an output. $\mathcal{B}$ sends X_i to $\mathcal{A}$ only if no previous test has failed.

In Mao game, $\mathcal{B}$ gets F as a return from GARBLE. $\mathcal{B}$ forwards F to $\mathcal{A}$ if $\Phi(f_0) = \Phi(f_1)$. If the side-informations of f_0 and f_1 do not coincide, both adversaries proceed to FINALIZE. Otherwise, the game continues like described above (recall that INPUT procedure is the same for games $PrvYyyPadap_\ell$ and $MaoYyyPadap_\ell$).

In Obv game, $\mathcal{B}$ gets F as a return and forwards it to $\mathcal{A}$ if $\Phi(f_0) = \Phi(f_1)$. Again, the game proceeds similarly as before. The only difference to the two other games, Prv and Mao, is that $\mathcal{B}$ does not perform the evaluation test in the case $|Q| = n$.

In all three cases, $\mathcal{A}$ and $\mathcal{B}$ may make at most $n \cdot \ell$ queries to INPUT after which $\mathcal{A}$ gives its answer $b_{\mathcal{A}}$ to $\mathcal{B}$. $\mathcal{B}$ answers $b_{\mathcal{B}} = 1$ if none of $\mathcal{A}'s$ previous queries

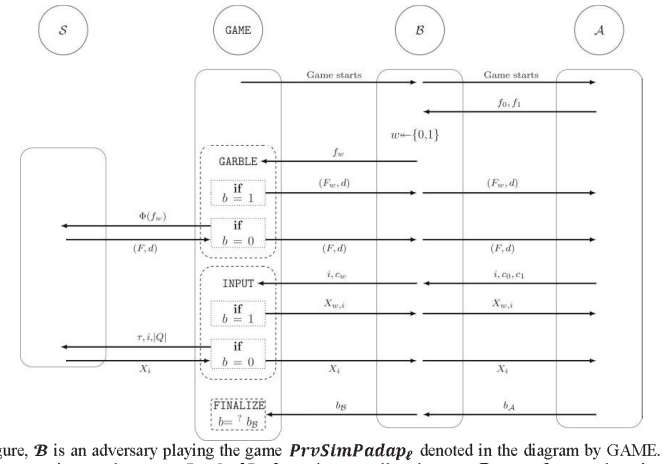


Figure 3 In the above figure, $\mathcal{B}$ is an adversary playing the game $\boldsymbol{PrvSimPadap_\ell}$ denoted in the diagram by GAME. $\mathcal{S}$ is the simulator in this game. $\mathcal{A}$ is an adversary presuming to play game $\boldsymbol{PrvIndPadap_\ell}$, but actually adversary $\mathcal{B}$ uses $\mathcal{A}$ as a subroutine by trying to emulate the actual game $\boldsymbol{PrvIndPadap_\ell}$. Note that every time $|\boldsymbol{Q}| = \boldsymbol{n}$, the simulator gets $\boldsymbol{y} = \boldsymbol{ev}(\boldsymbol{f_w}, \boldsymbol{x_{w,1}} \ldots \boldsymbol{x_{w,n}})$ as its input whereas for other values of $|\boldsymbol{Q}|$ the output $(\boldsymbol{F}, \boldsymbol{d})$ from simulator may depend only on $\boldsymbol{i}$ and $|\boldsymbol{Q}|$.

failed and $b_{\mathcal{A}} = c$. Otherwise $\mathcal{B}$ answers $b_B = 0$.

The possible scenarios in the games are the following. If $\Phi(f_0) \neq \Phi(f_1)$ then adversary $\mathcal{A}$ is forced to exit GARBLE with no output. Using other words, $\mathcal{A}$ does not get any information about the garbled function and garbled arguments, and therefore $\mathcal{A}$ does not have better strategy than guessing $b_{\mathcal{A}}$. Regardless of $\mathcal{A}$'s answer $\mathcal{B}$ answers $b_B = 0$ which is the correct answer with probability $\frac{1}{2}$. The second possibility is that $\mathcal{A}$ made a successful query to GARBLE but some of $\mathcal{A}$'s INPUT queries fails. In this case $\mathcal{A}$ has some information on which $\mathcal{A}$ can base her answer $b_{\mathcal{A}}$. However, adversary $\mathcal{B}$ ignores $\mathcal{A}$'s answer and answers $b_B = 0$.

The third possibility is that none of $\mathcal{A}$'s queries has failed. Let us now consider the two possibilities for b separately. First consider case $b = 0$. Then, $\mathcal{A}$ does not have any information about X_i's or F, so its answer is no better than a guess and guessing right always has probability $\frac{1}{2}$. Moreover, $\mathcal{B}$ loses its game if and only if $\mathcal{A}$ wins its game. As an example, consider the situation in which $\mathcal{A}$ answers $b_A = 0$. If $c = 0$, then $\mathcal{A}$ wins its game. However, $\mathcal{B}$ loses its game, because $\mathcal{B}$ answers $b_B = 1$ because $c = b_B$. The claim for three other cases can be proven in similar manner. In any of these cases, $\mathcal{B}$ wins the game with probability $\frac{1}{2}$. Now consider case $b = 1$. Then $\mathcal{A}$ has an advantage of answering the right b', because f and argument bits were created with the actual garbling scheme. The win probability of $\mathcal{B}$ is now $\frac{1}{2} + \frac{1}{2} \cdot Adv_{\mathcal{A}}$.

This case analysis lets us derive the win probability of adversary $\mathcal{B}$ against any PT simulator $\mathcal{S}$ as follows.

$$Pr[\mathcal{B}\ wins] = \frac{1}{2} Pr[\mathcal{B}\ wins | b = 1] + \frac{1}{2} Pr[\mathcal{B}\ wins | b = 0]$$

$$= \frac{1}{2}\left(\frac{1}{2} + \frac{1}{2} Adv_{\mathcal{A}}\right) + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2} + \frac{1}{4} \cdot Adv_{\mathcal{A}}.$$

Now, using $Pr[\mathcal{B}\ wins] = \frac{1}{2} + \frac{1}{2} \cdot Adv_B$, we obtain $Adv_A = 2 \cdot Adv_B$. We assumed $\mathcal{G}$ to be $xxx.sim_\ell$ secure against any PT simulator $\mathcal{S}$, so Adv_B is negligible. Therefore also $Adv_{\mathcal{A}}$ is negligible which proves the claim. ■

Theorem 6
$GS(prv.ind.padap_\ell, \Phi) \cup GS(obv.ind.padap_\ell, \Phi) \subseteq GS(mao.ind.padap_\ell, \Phi)$ holds for any $\ell \in \mathbb{N}$.

Proof: Omitted in this version due to lack of space. ■

Theorem 7
$GS(prv.sim.padap_\ell, \Phi) \cup GS(obv.sim.padap_\ell, \Phi) \subseteq GS(mao.sim.padap_\ell, \Phi)$ holds for any $\ell \in \mathbb{N}$.

Proof: Omitted in this version due to lack of space. ■

The results presented in this section allow us to present a hierarchy for adaptive security classes of projective garbling schemes. The hierarchy is similar to the hierarchies of static and adaptive garbling schemes [11, 12], as illustrated in fig. 4.

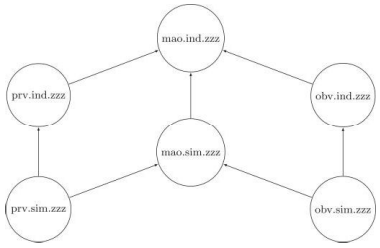


Figure 4 Hierarchy of security classes for $\boldsymbol{zzz} \in \{\boldsymbol{stat}, \boldsymbol{adap_\ell}, \boldsymbol{padap_\ell}\}$

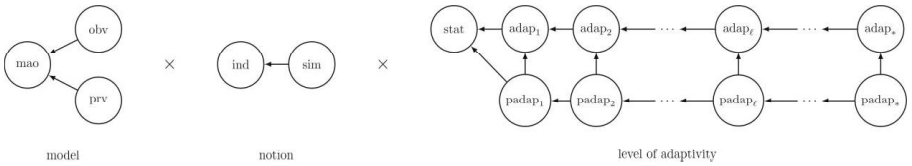


Figure 5 Hierarchy of garbling scheme classes represented as a Cartesian product of graphs

5 Conclusions

Garbled circuits are a useful tool for secure multiparty computation. Current constructions provide either efficiently constructable garbled circuits that are not reusable or reusable garbled circuits that can not be constructed efficiently. Our aim is to investigate the possibility to achieve at least some level of reusability while still contructing the garbled circuit efficiently. To reach this goal, we provide new definitions for classes of garbling schemes which in some sense extend the currently known classes of adaptively secure garbling schemes. Namely, we extend the results concerning the adaptive garbling schemes to adaptively secure projective garbling schemes. Compared to [1], we have modified the concept of fine-grained security by allowing non-deterministic encryption as well as introduced a new parameter ℓ acting as a threshold for the maximum number of fully specified garbled arguments.

We obtain several relations concerning the new security classes as well as the previously known classes. One of the main results is that the security classes defined in this paper form an infinite hierarchy. This hierarchy starts from the class of statically secure garbling schemes and ends to the class of adaptively secure garbling schemes allowing the same garbling to be used arbitrarily many times, if none of the classes between these two is empty. The infiniteness follows from two results. The first result tells that the class of garbling schemes allowing the same garbling to be used $\ell + 1$ times is properly included in a class with threshold value ℓ, unless the both are empty. The second result shows that the class allowing arbitrary reuse of the garbling is properly included in the security class with any threshold value ℓ. As another component of the hierarchy, we show how the different notions and models are related to each other when the level of adaptivity is fixed.

The results in [1, 2, 11, 12] and in this paper allow us to present the hierarchy for classes of garbling schemes extended with the classes of projective garbling schemes presented in this paper. We present the hierarchy of garbling schemes as a *directed graph Cartesian product* similarly as in [12]. The components of the product are the security notion (prv, mao, obv), the model (ind, sim) and the level of adaptivity (stat, adap, padap). The adaptive classes are additionally classified according to the threshold value ℓ telling how many fully specified garbled arguments may be produced with the same garbling scheme. The hierarchy is illustrated in fig.5.

References

[1] M. Bellare, V. T. Hoang, and P. Rogaway. Adaptively secure garbling scheme with applications to one-time programs and secure outsourcing. *Asiacrypt 2012*, 7685 of LNCS:134–153, 2012.

[2] ---. Foundations of garbled circuits. In *Proc. of ACM Computer and Communications Security (CCS'12)*, pages 784–796. ACM, 2012.

[3] M. Bellare and P. Rogaway. Code-based game-playing proofs and the security of triple encryption. *Advances in Cryptology, Proc. of Eurocrypt 2006*, 4004 of LNCS:409–426, 2006.

[4] D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. *Proc. of TCC 2011*, 6597 of LNCS:253–273, 2011.

[5] C. Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.

[6] S. Goldwasser, Y. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. *Proc. of the* 45^{th} *STOC*, pages 555–564, 2013.

[7] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute-based encryption for circuits. *Proc. of the* 45^{th} *STOC*, pages 545–554, 2013.

[8] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proc. of ACM Computer and Communications Security (CCS'06)*, pages 89–98. ACM, 2006.

[9] V. T. Hoang. *Foundations of Garbled Circuits*. PhD thesis, University of California, Davis, 2013.

[10] Y. Lindell and B. Pinkas. A proof of security of Yao's protocol for secure two-party computation. *Journal of Cryptology*, 22(2):161–188, 2009.

[11] T. Meskanen, V. Niemi, and N. Nieminen. Classes of garbled schemes. *Infocommunications Journal*, V(3):8–16, 2013.

[12] ---. Hierarchy for classes of garbling schemes. Submitted.

[13] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem. *Proc. of the* 41^{st} *STOC*, pages 333–342, 2009.

[14] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *Proc. of the* 37^{th} *STOC*, pages 84–93, 2005.

[15] A. Yao. How to generate and exchange secrets. *Proc. of* 27^{th} *FOCS, 1986.*, pages 162–167, 1986

Paper IV

IV

On Reusable Projective Garbling Schemes

T. Meskanen and V. Niemi and N. Nieminen (2014). In *2014 IEEE International Conference on Computer and Information Technology (CIT 2014)*, pages 315-322. IEEE

2014 IEEE International Conference on Computer and Information Technology

On Reusable Projective Garbling Schemes

Tommi Meskanen*, **Valtteri Niemi***†, **Noora Nieminen***‡

*Department of Mathematics and Statistics, University of Turku, FINLAND
† Xidian University, Xi'an, CHINA
‡ TUCS - Turku Centre for Computer Science, FINLAND
{tommes, pevani, nmniem}@utu.fi

***Abstract*—The popularity of network-based computation has increased during the past years. Storing data and performing computations using for example cloud services is commonplace nowadays. However, the information security in network-based computation rarely is satisfactory. One technique to improve information security for networked computations is based on *secure multiparty computations*. The idea of multiparty computation originates from Yao, who considered the secure computation with two parties by introducing the technique called *garbled circuits*. The technique was then generalized, but still the exact definitions for different security measures were not established until Bellare, Hoang and Rogaway proposed a definition of *a garbling scheme* and several security notions for garbling schemes. This gave inspiration to study the security properties of garbling schemes from different perspectives, including *adaptivity* and *reusability*. In this paper, we study the adaptive security for a certain type of reusable garbling schemes. First, we consider general constraints to achieve security for this specific type of garbling schemes. Then we analyze the adaptive security classes of reusable garbling schemes. More specifically, we prove a relation between security classes that has been left open in earlier research.**

***Keywords*—secure multiparty computations, garbling schemes, reusability, adaptive security, projectivity**

I. Introduction

During the past decade, the popularity of distributed computing has increased enormously. Distributed computing projects like SETI@Home [1], Folding@Home [19] and Mersenne Prime Search [15] are just a few examples where multiple clients together participate over the Internet on complex computing managed by a central server. Also other services, Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) among many others, can be run on network-based environments called clouds. Despite of the wide use of cloud computing, there still are some major problems, for example lack of proper privacy [13], [22]. Many cryptographic methods have been proposed to overcome the privacy issues - one such method is *secure multiparty computation*.

The history of multiparty computations can be said to have started from Yao's paper [23], where the idea of computation with two parties was first introduced. A technique to perform 2-party computations was developed by Yao in [24]. This technique was named as *garbled circuits* by Beaver, Micali and Rogaway, who also generalized the technique to multiparty setting in [2]. Ever since, garbled circuits have been used for various purposes, despite of the fact that exact measures for security had not been defined yet. The first proof of security was given by Lindell and Pinkas in [14]. This proof shows that a particular protocol using garbled circuits is secure rather than showing that garbling as a technique is secure. This latter aspect was first considered by Bellare, Hoang and Rogaway who proposed the first definition of garbling scheme, at the same time elevating the garbled circuits from a mere cryptographic technique into a cryptographic primitive by introducing several notions for security in [4].

The security of garbling schemes has been considered from different perspectives after the security definitions for garbling schemes were established. The first definitions of Bellare et al. describe only *static* security, meaning that the choice of function f and its argument x have to happen simultaneously. Some applications, for example one-time programs [10] and secure outsourcing [7] require a more adaptive setting. Therefore, Bellare et al. extended their security definitions to adaptive security in [3]. While still being adaptive, the garbling schemes defined by Bellare et al. do not support the reusability of the garbled function. In other words, performing a garbled evaluation for function f with different arguments x requires that the garbled function is constructed anew every time a new argument comes into play. This kind of regarbling becomes inefficient if the amount of different argument values is big. This leaves an open question, whether the same garbled function could be reused somehow. This question was answered in a paper [9] from Goldwasser et al. in which a technique to construct reusable garbled circuits is introduced. The construction is based on functional encryption [6] which in turn has used fully-homomorphic encryption (FHE) [8] and attribute-based encryption (ABE) [11], [12] as its building blocks. However, constructing a reusable garbled circuit in this way is impractical due to the impracticality of FHE. Also, the existence of reusable garbled circuits relies on Learning With Errors assumption (see [20], [21] on more information about this assumption). As an attempt to overcome these problems, we proposed a new way of combining the reusability property and adaptive security in [17]. We introduced a new parameter, *reusability parameter* ℓ, telling how many garbled evaluations can be securely performed using the same garbled function. In [18], we also introduced the reusability parameter in the case of bitwise adaptive security, which generalizes the *fine-grained adaptive security* defined by Bellare et al. in [3]. In both papers [17], [18], we considered relations between different security notions. These results together allow the classes of garbling schemes to be categorized into an infinite hierarchy.

978-1-4799-6239-6/14 $31.00 © 2014 IEEE
DOI 10.1109/CIT.2014.63

To demonstrate the applicability of the concepts presented above, consider the following scenario. A health center treats patients, whose medical records are collected and stored on a cloud. Also all other services that the health center uses are outsourced into cloud environment. One of these services is a medical diagnostics software suite that is used to analyze the medical records in order to diagnose the symptoms and help in choosing the appropriate treatment. However, the cloud might be untrustworthy and does not provide an appropriate level of privacy to the medical records that must be kept absolutely private. Absolute privacy must also be applied to the medical diagnosis. Furthermore, there are privacy issues in the use of diagnostics software suite itself. The diagnostics software suite contains several tools to analyze different symptoms - one of them may be targeted at neurological symptoms and another at cardiac symptoms. The tool used by the doctor reveals patient's possible disease which undesirably compromises patient's privacy. This indicates that also the diagnostics software suite should be kept secret. The implementation details of software suites usually are business secrets which gives another reason to hide the diagnostics software suite. All in all, to securely run the diagnostics software on medical records, the health center and the cloud service provider need a method for secure function evaluation. Suppose that the health center has decided to use garbling technique as the method to achieve the demanded security. We may consider the medical records together with the choice of the diagnostic tool acting as an argument to the diagnostics software suite. The diagnostics software suite in turn is interpreted as the function whose evaluation we want to keep secret.

How does the garbled evaluation of the diagnostics software suite work on the cloud? The first step is that the health center garbles the diagnostics software suite before giving it to the cloud. This can be done before any medical records exist. When a new patient comes for treatment, the doctor in the health center garbles the medical records and sends them to the cloud. The cloud then performs the garbled evaluation and sends the garbled result to the health center. The doctor in the health center then applies ungarbling to get the actual result of the diagnostics program. After finishing the session with this patient, the doctor meets the next patient and analyzes the medical records of the second patient in similar manner. Efficient analyzing of the medical data requires that the re-garbling of the diagnostics software suite is avoided. To avoid garbling the software suite anew, the garbling scheme must be reusable. To summarize, the garbling scheme used by the health center and the cloud must have the following properties. First, the garbling scheme must keep the medical data as well as the diagnose private (the garbling scheme must keep the function, argument and value private). Secondly, the medical data of patients is garbled independently after garbling the diagnostics program (the garbling scheme must be adaptive). Finally, the diagnostics program is used for several different medical records (the garbling scheme must be reusable).

Our contributions. In this paper, we study the adaptively secure reusable garbling schemes by filling the gaps of [18]. There are still relations that have not yet been covered for the classes of adaptively secure garbling schemes supporting reusability. Especially we concentrate on the bitwise adaptive security and the relation between the two security models, *indistinguishability-based* and *simulation-based* model. In [18], we proved that security in simulation-based model always implies security in indistinguishability-based model, but the converse relation is left open. In this paper, we provide a condition under which security in indistinguishability-based model implies simulation-based security. We also provide a lower limit for the length of the encrypted version of the argument x in a bitwise adaptively secure reusable garbling scheme.

II. Definitions

This section contains the basic definitions and notations. Let us start with a concept of a *string* which is a finite sequence of bits. Let S be a finite set. Notation $y \xleftarrow{} S$ means that an element is selected uniformly at random from the set S, and this element is assigned as a value to the variable y. If A denotes an algorithm, then notation $A(x_1, \ldots, x_n)$ refers to the output of the algorithm A on inputs $x_1, \ldots, x_n$. We say that a function $f : \mathbb{N} \to \mathbb{R}$ is negligible if for every $c > 0$ there is an integer N_c such that $|f(n)| < n^{-c}$ for all $n > N_c$. Next we give the formal definition of a garbling scheme as well as the central concepts needed to describe the security of garbling schemes.

A. Garbling scheme

Formally, a garbling scheme is a 5-tuple $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev})$ of algorithms, where algorithms $\mathtt{Gb}$ and $\mathtt{En}$ are probabilistic and the rest are deterministic. The last component in the 5-tuple is the evaluation algorithm $\mathtt{ev}$ which is used to evaluate the function f on argument x. In other words, evaluation algorithm takes f and x on inputs and computes the final function value $y = \mathtt{ev}(f, x)$. Here f is treated as a string representing the original function. The computation of $\mathtt{ev}(f, x)$ is exactly what is to be garbled with the four other algorithms $\mathtt{Gb}$, $\mathtt{En}$, $\mathtt{De}$ and $\mathtt{Ev}$.

The first component $\mathtt{Gb}$ is the garbling algorithm. It takes f and 1^k, where $k \in \mathbb{N}$ is a security parameter, as inputs and returns (F, e, d) as output. The first component is the garbled function F. String e is encryption key that is used when the argument x is encrypted to garbled argument X using the probabilistic encryption algorithm $\mathtt{En}$. Using other words, the algorithm $\mathtt{En}$ takes the key e and argument x as inputs and outputs the garbled argument $X = \mathtt{En}(e, x, r)$ where r is an additional input that provides the randomness for the encryption algorithm $\mathtt{En}$. The garbled function F is evaluated on garbled argument X with the garbled evaluation function F, i.e. the garbled evaluation function computes the garbled function value $Y = \mathtt{Ev}(F, X)$. Finally, the garbled function value is decrypted with decryption key d using the decryption algorithm $\mathtt{De}$. The decryption algorithm returns the final function value $\mathtt{De}(d, Y) = y = \mathtt{ev}(f, x)$. Note that, regardless of the choice of r, $\mathtt{En}$ gives the correct output X in the sense that the final function value is always y. Figure 1 illustrates how a garbling scheme works.

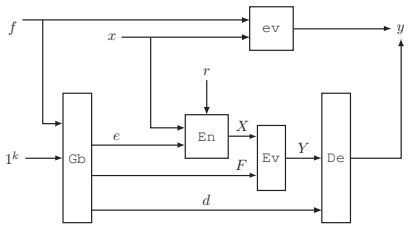


Figure 1. Idea behind garbling. The diagram shows that the final value obtained via garbling must coincide the final value obtained by direct evaluation, i.e. $\mathtt{ev}(f,x) = y = \mathtt{De}(d,Y)$ where F is the garbled function, $Y = \mathtt{Ev}(F,X)$ the garbled value and $X = \mathtt{En}(e,x,r))$ the garbled argument.

B. Projectivity of a garbling scheme

Some applications, like one-time programs [10], allow the user to feed his argument to the function bit by bit. This sets additional requirements to the garbling scheme, namely the encryption algorithm needs to treat the argument in pieces instead of processing the whole input at once. This property is called *projectivity*. We next define the concept more formally.

Let $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev})$ be a garbling scheme which is used to evaluate function f. Let $x = x_1 \dots x_n \in \{0,1\}^n$ and $x' = x'_1 \dots x'_n \in \{0,1\}^n$ be any two arguments for f. Let us consider the specific case where the same encryption key e and the same randomness value r are used in encrypting both x and x'. We say that garbling scheme $\mathcal{G}$ is *projective* if the two garbled arguments $X = \mathtt{En}(e,x,r)$ and $X' = \mathtt{En}(e,x',r)$ can be presented in form $X = (X_1, \dots, X_n)$, $X' = (X'_1, \dots, X'_n)$ in such way that the following condition holds: $x_i = x'_i$ iff $X_i = X'_i$.

C. Side-information function

By the concept of *a side-information function*, we capture the information revealed about f by the garbling process. More formally, a side-information function Φ deterministically maps f to $\Phi(f)$. For example, in the case of computation model using circuits and the corresponding natural evaluation algorithm denoted by $\mathtt{ev}_{circ}$, the value $\Phi(f)$ might be the size of the circuit that was garbled, the topology of the circuit or something else - even the whole initial circuit.

D. Code-based games

The security notions for garbling schemes are based on *code-based games*. Following the terminology presented in [5], a game is a collection of procedures containing three types of procedures: `INITIALIZE`, `FINALIZE` and other named procedures. We assume throughout this paper that procedures `INITIALIZE` and `FINALIZE` are compulsory in a game, whereas other procedures are optional.

The entity playing a game is called *an adversary*. Every game starts with `INITIALIZE` procedure. Then the adversary may invoke other procedures before feeding its output to the `FINALIZE` procedure. Based on this input from the adversary `FINALIZE` procedure creates a string that tells the outcome from the game typically consisting of one bit of information: whether the adversary has won or not.

E. Security concepts

In this section we shortly describe the security concepts for garbling schemes following the terminology used in [3], [4], [17]. According to [17], garbling schemes can be characterized by three components: *security notion*, *security model* and *level of adaptivity*. Next paragraphs contain a short description from each of these three components.

Security notion tells how much information is allowed to be leaked during the garbled evaluation. The notions are *privacy*, *oblviousness* and *matchability-only*. A garbling scheme achieving **privacy (prv)** does not allow f or x to be leaked, but $f(x)$ can be leaked. A garbling scheme achieving **obliviousness (obv)** keeps all, f, x and $f(x)$ secret. A **matchability-only (mao)** secure garbling scheme hides f, x and $f(x)$ but when computing $y_1 = f(x_1)$ and $y_2 = f(x_2)$ the scheme is allowed to leak whether $y_1 = y_2$.

Security model is either *simulation-* or *indistinguishability-based*. **Indistinguishability (ind)** refers to inability to distinguish garblings of two similar functions from each other. In **simulation-based model (sim)**, the task of the adversary is to distinguish a real garbling from simulated information mimicking the real garbling.

Level of adaptivity describes how function f and argument x may be treated during the garbled evaluation. There are several levels of adaptivity: *static*, *adaptive* and *bitwise adaptive* security. In **static (stat)** security, both f and x need to be fixed at once and must not be altered afterwards. **Adaptive (adap)** security, also known as *coarse-grained security*, allows f and x to be handled separately from each other, meaning that f can be fixed before x. **Bitwise adaptive (padap)** security, also known as *fine-grained security*, refines adaptive security by allowing the argument x to be handled bitwise.

Notations. Every security class is determined by these three components: security notion, security model and level of adaptivity. We use notation $xxx.yyy.zzz$, where $xxx \in \{\text{prv, obv, mao}\}$, $yyy \in \{\text{sim, ind}\}$ and $zzz \in \{\text{stat, adap}_\ell, \text{padap}_\ell\}$, for a $xxx.yyy.zzz$ secure garbling scheme. The corresponding security class of garbling schemes is denoted by $\mathtt{GS}(xxx.yyy.zzz)$.

To determine whether a garbling scheme belongs to a certain security class, we need a security game, which is denoted by $XxxYyyZzz$ for $Xxx \in \{\text{Prv, Obv, Mao}\}$, $Yyy \in \{\text{Sim, Ind}\}$ and $Zzz \in \{\text{Stat, Adap}_\ell, \text{Padap}_\ell\}$. The game is played by an adversary, who wins the game with certain probability. The games are designed in such manner that simple guessing strategy gives win probability $\frac{1}{2}$. If the win probability is greater than $\frac{1}{2}$ then the adversary has an advantage over pure guessing. If this advantage happens to be non-negligible, then the garbling scheme is not secure in the sense determined by the game. More formal definitions of advantage and security are given below.

Advantage of an adversary. We define the advantage of an adversary A in both games as follows:

$$\mathbf{Adv}(\mathcal{A}, k) = 2 \cdot \Pr[\mathcal{A} \text{ wins the game }] - 1.$$

Security of a garbling scheme. If the advantage function $\mathbf{Adv}(\mathcal{A}, \cdot)$ is negligible for all PT adversaries $\mathcal{A}$ in XxxIndZzz$_{\mathcal{G},\Phi}$ game then we say that the garbling scheme $\mathcal{G}$ is *xxx.ind.zzz secure over* Φ. Similarly, we define that a garbling scheme $\mathcal{G}$ is *xxx.sim.zzz secure over* Φ if for every PT adversary $\mathcal{B}$ there exists a PT simulator $\mathcal{S}$ such that the advantage $\mathbf{Adv}(\mathcal{B}, k)$ in the XxxSimZzz$_{\mathcal{G},\Phi,\mathcal{S}}$ game is negligible.

III. Classes of adaptively secure projective garbling schemes

As explained above, the security of a garbling scheme can be characterized by three properties: notion, model and level of adaptivity. In this paper, we concentrate on the bitwise adaptive security, i.e. the padap$_\ell$ security for all three notions and in both security models. We start by defining the security games PrvSimPadap$_\ell$,MaoSimPadap$_\ell$,ObvSimPadap$_\ell$, PrvIndPadap$_\ell$, MaoIndPadap$_\ell$ and ObvIndPadap$_\ell$. We do this by presenting games PrvSimPadap$_\ell$ and PrvIndPadap$_\ell$ as flowcharts (see figs. 2(a) and 2(b)) and explaining the differences between all six different notions and models.

All six games consist of four procedures: INITIALIZE, GARBLE, INPUT, FINALIZE. The game always starts with INITIALIZE procedure, in which the challenge bit b is chosen randomly. The compulsory call to INITIALIZE is followed by a query to the next procedure called GARBLE which prepares garbled function F to the adversary. After this procedure an adversary may call procedure INPUT which prepares the garbled arguments to the adversary. Finally, the adversary calls FINALIZE, a procedure telling whether the adversary has won the security game. The procedures INITIALIZE and FINALIZE are the same for all six games. The two other procedures, GARBLE and INPUT, depend on the game. Next we describe these two procedures in more detail. First, we describe procedure GARBLE in simulation- and indistinguishability-based games. Along the descriptions we also describe the differences between the security notions prv, mao and obv. The procedure INPUT is explained in similar manner.

In simulation-based games, procedure GARBLE prepares the garbled function based on the challenge bit b: either the garbled function F and decryption key d are created by the garbling algorithm ($b = 1$) $\mathtt{Gb}$ or by a simulator $\mathcal{S}$ ($b = 0$). Finally, GARBLE returns pair (F, d) in PrvSimPadap$_\ell$ game, whereas in MaoSimPadap$_\ell$ and ObvSimPadap$_\ell$ games the procedure returns F without d.

In indistinguishability-based games, procedure GARBLE takes two functions, f_0 and f_1 as its inputs. First, the procedure checks whether the two functions are compatible with each other, i.e. whether $\Phi(f_0) = \Phi(f_1)$. If the functions do not share the same side-information, then procedure GARBLE is exited without no return value. The adversary proceeds directly to FINALIZE and is forced to guess the challenge bit b. Otherwise, the GARBLE procedure prepares a garbled function based on the challenge bit, i.e. returns the garbled version of f_b. In the game PrvIndPadap$_\ell$ the adversary additionally gets the decryption key d, while in the two other games the adversary doesn't get d.

The INPUT procedure in simulation-based games takes an index i and the i^{th} bit of argument x as its input. First the procedure checks whether the index i is already processed. If i^{th} bit of the argument has been processed earlier, then an adversary playing the game does not receive the garbled argument bit and the procedure is exited. The adversary is forced to proceed directly to FINALIZE and has to give the answer based on the output from GARBLE procedure. On the other hand, if i has not been processed earlier, then the procedure creates a garbling of the argument bit based on the choice of the challenge bit: if $b = 1$ then X_i is the encryption of x_i, and if $b = 0$ then X_i is created by the simulator. Every time the whole argument $x = x_1 \dots x_n$ becomes specified, the list of processed indices Q is emptied for the next round. In games $\{$Prv, Mao$\}$SimPadap$_\ell$ the simulator is allowed to use $y = \mathtt{ev}(f, x)$ to create the garbling of the last missing argument bit whereas in ObvSimPadap$_\ell$ game it is not allowed to know $y = \mathtt{ev}(f, x)$.

The procedure INPUT in indistinguishability-based games takes index i as well as the i^{th} bit of arguments x_0 and x_1 as its inputs. The procedure first checks that index i has not yet been processed. If it has been processed, the procedure ends and the adversary proceeds to FINALIZE. It is worth noting that the game chooses a new challenge bit so that the win probability of an adversary is always the same as in the case of pure guessing. Otherwise, the procedure proceeds to the next check. If the entire argument becomes specified, meaning that $|Q| = n$, then Q is emptied in all three games. In games $\{$Prv, Mao$\}$IndPadap$_\ell$ there is an additional check $\mathtt{ev}(f_0, x_0) \stackrel{?}{=} \mathtt{ev}(f_1, x_1)$. If this check is not passed, the procedure is over and the adversary needs to guess the challenge bit based on the information it is already possessing. Also in this case the challenge bit is chosen afresh, and the win probability of an adversary is the same as in pure guessing. If the check is passed or $|Q| < n$, INPUT procedure prepares a garbling $X_{b,i}$ of the argument bit $x_{b,i}$ to be returned to the adversary.

Procedure INPUT may be called several times. However, in every game there is *a threshold* for the maximum number of INPUT queries: The parameter ℓ in all games describes how many times the adversary is allowed to get a fully specified garbled argument by calling INPUT procedure. In other words, the adversary is allowed to call INPUT at most $n \cdot \ell$ times in total.

In both simulation- and indistinguishability-based games the adversary tries to find out the challenge bit chosen by the game. In simulation-based game this means that the adversary tries to find out, based on the information from procedures GARBLE and INPUT, whether F and the various garbled argument bits were created by the actual garbling algorithm or were they devised by a simulator. In indistinguishability-based game the task of the adversary is to determine which of the functions, f_0 or f_1, got garbled based on the information it has received

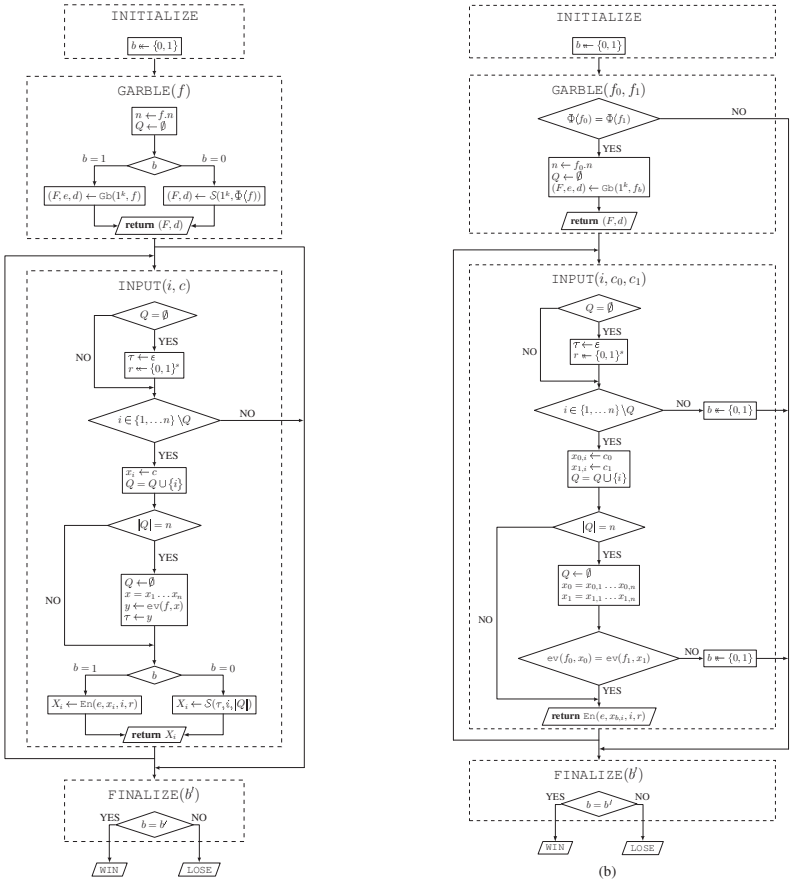


Figure 2. Flowcharts for the games PrvSimPadap$_\ell$ (a) and PrvindPadap$_\ell$ (b). The game PrvSimPadap$_\ell$ depends on the choice of $\mathcal{G}$, Φ and $\mathcal{S}$ whereas the game PrvIndPadap$_\ell$ depends on the choice of $\mathcal{G}$ and Φ.

during the game. The procedure FINALIZE tells whether the adversary answered the correct challenge bit. If the adversary answered correctly, it wins the security game.

Next we consider the role of the reusability parameter ℓ for the security classes defined above. In [18] it is shown, that the classes of bitwise adaptive security form an infinite hierarchy: If $\mathcal{F}_{\ell+1}$ denotes the fine-grained security class with reusability parameter $\ell+1$ and $\mathcal{F}_\ell$ the corresponding security class with reusability parameter ℓ, then $\mathcal{F}_{\ell+1}$ is properly included in $\mathcal{F}_\ell$. This gives the inspiration to consider the case in which arbitrary many garbled evaluations using the same garbled function. In [18] this case $\ell = *$ is also discussed.

It is worth noting that for the cases $\ell > 2$ the encryption algorithm En of the garbling scheme must be non-deterministic. The next theorem shows that the length of the garbled argument X must not be too short in a bitwise adaptively secure garbling scheme with reusability parameter $\ell \geq 2$.

Theorem 1: Let $\mathcal{G} = (\mathrm{Gb}, \mathrm{En}, \mathrm{De}, \mathrm{Ev}, \mathrm{ev}) \in \mathrm{GS}(\text{xxx.sim.padap}_2, \Phi)$ be a xxx.sim.padap$_2$ secure garbling scheme, where xxx $\in \{$prv, mao, obv$\}$. For every constant

$c > 0$ there exists a threshold $k_c \in \mathbb{N}$ for the security parameter k such that whenever k exceeds k_c the length of the garbled argument X must be at least $nc\log k + n$ where n is the length $|x|$ of the original argument x.

Proof: To prove the theorem, we need the following lemma.

Lemma *For every constant $c > 0$ there exists a threshold $k_c \in \mathbb{N}$ for the security parameter k such that whenever k exceeds k_c the length of the i^{th} garbled argument bit X_i must be at least $c\log k + 1$ for all $i \in \{1, \ldots, n\}$.*

Proof of lemma Suppose on the contrary that there exists a constant $c > 0$ such that for all values k_c the following holds: for some $i \in \{1, \ldots, n\}$ and $k > k_c$ the length of X_i is $|X_i| \leq c \cdot \log k + 1$. This implies that the argument $x_i \in \{0,1\}$ will have in average at most $\frac{2^{c\cdot\log(k)+1}}{2} = 2^{\log(k^c)} = k^c$ possible encryptions. For simplicity of the presentation we assume that both possible argument bits 0 and 1 have exactly k^c equally probable encryptions. (If this is not the case then the task of the adversary becomes easier.) Let us design an adversary $\mathcal{A}$ who plays the PrvSimPadap$_\ell$ game. Our aim is to show that $\mathcal{A}$ has a strategy that will give her a non-negligible advantage and it follows that $\mathcal{G}$ is insecure.

Adversary $\mathcal{A}$ uses all its possible $2n$ queries to INPUT. Adversary $\mathcal{A}$ makes its i^{th} and $(n+i)^{th}$ query to INPUT using the argument bits at random from $\{0,1\}$. Let these bits be denoted by b_1 and b_2. Both times, adversary gets the encryption of the bit, let those be denoted by $B_1 = \mathrm{En}(e, b_1, i, r_1)$ and $B_2 = \mathrm{En}(e, b_2, i, r_2)$. Now, if $B_1 = B_2$ and $b_1 = b_2$, then the adversary answers $b_{\mathcal{A}} = 1$. If $B_1 = B_2$, but $b_1 \neq b_2$ then the adversary answers $b_{\mathcal{A}} = 0$. Otherwise the adversary guesses $b_{\mathcal{A}}$. The other indices than i do not affect $\mathcal{A}$'s strategy. Let us now analyze the win probability of the adversary $\mathcal{A}$ using the above strategy in game PrvSimPadap$_2$.

Consider first the case in which the challenge bit is $b = 1$. In this case, the actual encryption algorithm is used to create B_1 and B_2. Adversary $\mathcal{A}$ wins the game, if $B_1 = B_2$ and $b_1 = b_2$. The latter happens with probability $\frac{1}{2}$ whereas the former happens with probability $\frac{1}{k^c}$ given $b_1 = b_2$. The case $B_1 = B_2$ while $b_1 \neq b_2$ is not possible, because B_1 and B_2 are created using the encryption algorithm En. Adversary wins also with probability $\frac{1}{2}$ if $B_1 \neq B_2$, because in this case $\mathcal{A}$ guesses and a guess is correct with probability $\frac{1}{2}$. In particular, $\mathcal{A}$ wins with probability $\frac{1}{2}$ if $b_1 \neq b_2$. Thus the winning probability of adversary $\mathcal{A}$ is

$$\begin{aligned}
&\Pr[\mathcal{A} \text{ wins } |b = 1] \\
&= \frac{1}{2} \cdot \frac{1}{k^c} \cdot 1 + \frac{1}{2} \cdot (1 - \frac{1}{k^c}) \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} \\
&= \frac{1}{2}\left(\frac{1}{k^c} + \left(1 - \frac{1}{k^c}\right) \cdot \frac{1}{2} + \frac{1}{2}\right) \\
&= \frac{1}{4} \cdot \frac{1}{k^c} + \frac{1}{2}.
\end{aligned}$$

Then consider the other choice for the challenge bit, i.e. $b = 0$. In this case, simulator $\mathcal{S}$ is used to create B_1 and B_2. Adversary $\mathcal{A}$ wins the game, if $B_1 = B_2$ and $b_1 \neq b_2$. On the other hand, if $B_1 = B_2$ and $b_1 = b_2$, $\mathcal{A}$ will lose the game. Adversary $\mathcal{A}$ will win with probability $\frac{1}{2}$ if $B_1 \neq B_2$ because $\mathcal{A}$ guesses in this case and a guess is correct with this probability. Therefore we obtain as a summary that $\Pr[\mathcal{A} \text{ wins } |b = 0] = \frac{1}{2}$.

Now, combining these probabilities we get

$$\begin{aligned}
&\Pr[\mathcal{A} \text{ wins }] = \tfrac{1}{2}\Pr[\mathcal{A} \text{ wins}|b = 1] + \tfrac{1}{2}\Pr[\mathcal{A} \text{ wins}|b = 0] \\
&= \frac{1}{2} \cdot \left(\frac{1}{4} \cdot \frac{1}{k^c} + \frac{1}{2}\right) + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8} \cdot \frac{1}{k^c} + \frac{1}{2}
\end{aligned}$$

The advantage of adversary $\mathcal{A}$ is now

$$\mathbf{Adv}_{\mathcal{A}} = 2 \cdot \Pr[\mathcal{A} \text{ wins }] - 1 = 2 \cdot \left(\frac{1}{8} \cdot \frac{1}{k^c} + \frac{1}{2}\right) - 1 = \frac{1}{4k^c}.$$

Now we see that adversary $\mathcal{A}$ has a non-negligible advantage regardless of the simulator $\mathcal{S}$. Therefore the garbling scheme $\mathcal{G}$ cannot belong to security class xxx.sim.adap$_2$ which is a contradiction. This completes the proof of the lemma.

The original claim follows now from the lemma as follows. Let $x = x_1 \ldots x_n$ the argument consisting of n bits and $X = X_1 \ldots X_n$ where $X_i = \mathrm{En}(e, x_i, i, r)$ is the encryption of the i^{th} bit. Then for each index i we have the condition $|X_i| > c\log k+1$, so $|X| = \sum_{i=1}^{n} |X_i| > n \cdot (c\log k+1) = nc\log k + n$, which was to be proved. □

The above result sheds more light on a bitwise adaptively secure garbling scheme. Now we continue by considering the relations between the different security models and notions. In [18] many relations are already studied. For example, it was proven for all notions, privacy, matchability-only and obliviousness, that security in simulation-based model implies indistinguishability-based security (Theorem 4, [18]). Another result in [18] was about the relation between the notions, showing that matchability-only is at least as easily achieved as privacy or obliviousness. One relation not discussed in [18] is the potential implication from indistinguishability-based security to simulation-based security. This is our topic in the next section.

IV. From indistinguishability-based security to simulation-based security

It has been shown for every security notion enjoying the certain level of adaptivity that the security in simulation model implies security in indistinguishability model [4], [16]–[18]. Also in some cases, the indistinguishability-based security implies simulation-based security under some conditions. In static security, the concept of *efficient invertibility of* (ev, Φ), where ev is the evaluation function and Φ the side-information function, is the condition for indisinguishability-based security to imply simulation-based [4]. We say that (Φ, ev) is efficiently invertible if there is a polynomial time algorithm that on input (ϕ, y) , where $\phi = \Phi(f')$ and $y = \mathrm{ev}(f', x')$ for some f' and $x' \in \{0,1\}^{f'.n}$, returns (f, x) satisfying $\Phi(f) = \phi$ and $\mathrm{ev}(f, x) = y$.

In the adaptive setting, the same condition does not suffice anymore and a new way of defining efficient invertibility of

$(\mathtt{ev}, \Phi)$ is needed - this is called componentwise effecient invertibility in [17]. Componentwise invertibility means that there is an algorithm that can first find a function f based on side-information $\phi = \Phi(f')$ for some function f' and then an argument x based on $y = \mathtt{ev}(f', x')$ for the function f' and for some argument x'. Again, (f, x) needs to satisfy $\Phi(f) = \phi = \Phi(f')$ and $\mathtt{ev}(f, x) = y = \mathtt{ev}(f', x')$. The inversion needs to be twofold, because the function f and argument x are handled separately in adaptive security. It is also apparent that for bitwise adaptivity, the componentwise invertibility is not sufficient, because the arguments are now handled bitwise and separately from f. Therefore, the concept of efficient invertibility needs one more modification.

We provide now definition for *bit- and componentwise invertibility*. After the definition, we provide an example of $(\Phi, \mathtt{ev})$ which is efficiently invertible in this new sense. Then we show that under this condition of bit- and componentwise efficient invertibility, indistinguishability-based bitwise adaptive security implies simulation-based bitwise adaptive security.

Definition We define that $(\Phi, \mathtt{ev})$ is *bit- and componentwise efficiently invertible*, if there exists a polynomial time algorithm A such that

1) algorithm A first finds f using $\phi = \Phi(f')$ for some f'
2) after constructing f, A then finds the i^{th} argument bit as follows. If $|Q| < n$ then the algorithm constructs c_i based on i and $|Q|$. If $|Q| = n$ then the algorithm A uses i, $|Q|$ and y, where $y = \mathtt{ev}(f', x')$ for f' and for some x'. The bit c_i must satisfy the condition: if $x = c_1 \dots c_i \dots c_n$ and f are created by the algorithm A , then $\mathtt{ev}(f, x) = y = \mathtt{ev}(f', x')$ for given f' and $x' = c'_1 \dots c'_i \dots c'_n$.

Example: Let us consider an example of (Φ, ev) that is bit- and componentwise efficiently invertible. Let $\Phi = \Phi_{size}$ and $\mathtt{ev} = \mathtt{ev}_{circ+trunc}$. On input f, x the evaluation function $\mathtt{ev}_{circ+trunc}$ computes y_1 where y_1 is obtained by computing $y_1 \dots y_m = \mathtt{ev}_{circ}(f, x)$ and truncating the output to its first bit. Recall that side-information function Φ_{size} leaks (n, m, q) where m is the length of x, n the length of y and q the number of gates in the circuit for f.

Let $\phi = \Phi(f')$ for some function f'. Given $\phi = (n, m, q)$ the inverter constructs f as follows. In the circuit for function f, we place $n-1$ gates in such way that they compute AND of all input bits. The result is then the first output bit. If $q > n-1$ then the rest of the gates are placed arbitrarily.

Then, the algorithm needs to invert the argument bits one by one based on $|Q|$, i and possibly y. The algorithm does this as follows. If $|Q| < n$, the inverter simply chooses $c_i = 1$ regardless of i and $|Q|$. If $|Q| = n$, then the inverter gets to know $y = y_1 = \mathtt{ev}_{circ+trunc}(f, x)$ and can choose c_i accordingly. The inverter always chooses the right last bit, because the function f was designed in such a way that all input bits affect to the q^{th} gate, the output gate for y_1. ►

Theorem 2: Let $\mathcal{G} \in (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev})$ be a PrvIndPadap$_\ell$ secure garbling scheme with $\ell > 1$. Then $\mathcal{G}$ is also a PrvSimPadap$_\ell$ secure garbling scheme over Φ if $(\Phi, \mathtt{ev})$ is bit- and componentwise efficiently invertible.

Proof: Suppose that $\mathcal{G}$ is a prv.ind.padap$_\ell$ secure garbling scheme. We will prove that $\mathcal{G}$ is also prv.sim.padap$_\ell$ secure. Let $\mathcal{B}$ be an arbitrary adversary playing the PrvSimPadap$_\ell$ game with simulator $\mathcal{S}$. We construct an adversary $\mathcal{A}$ for the game PrvIndAdap$_\ell$.

Let us start with the construction of the simulator for adversary $\mathcal{B}$ in game PrvSimPadap$_\ell$. Here we use the fact that $(\Phi, \mathtt{ev})$ is efficiently invertible. Namely, then there exists an algorithm A that is as an efficient, bit- and componentwise (Φ, ev)-inverter. The simulator uses algorithm A as follows. On input $(1^k, \Phi)$ simulator $\mathcal{S}$ lets $f \leftarrow A(\phi)$ and then $(F, e, d) \leftarrow \mathtt{Gb}(1^k, f)$. Finally, the simulator outputs (F, d). If $|Q| < n$, then the simulator is called with input $(\epsilon, i, |Q|)$. The simulator in turn calls A with input $(i, |Q|)$ getting x_i as a return. If $|Q| = n$ then the simulator is called with input $y, i, |Q|$, where $y = \mathtt{ev}(f', x')$. The triplet $(y, i, |Q|)$ is also the input to the algorithm A, which returns now x_i such that the condition $\mathtt{ev}(f, x) = y = \mathtt{ev}(f', x')$ holds. Then the simulator encrypts x_i with e, getting $X_i = \mathtt{En}(e, x_i, i, r)$ which is then returned to the adversary. The reason to construct the simulator $\mathcal{S}$ in this way becomes apparent later in the proof.

Let us now construct the adversary $\mathcal{A}$ that uses adversary $\mathcal{B}$ as a subroutine. The game starts by the choice of challenge bit for $\mathcal{A}$'s PrvIndPadap$_\ell$ game. Now $\mathcal{A}$ is told to send functions f_0, f_1 to its GARBLE procedure. Instead of sending them immediately, $\mathcal{A}$ challenges adversary $\mathcal{B}$ to play PrvSimPadap$_\ell$ game. $\mathcal{B}$ presumes playing an original PrvSimPadap$_\ell$ game even though it is mimicked by adversary $\mathcal{A}$. Adversary $\mathcal{B}$ sends f to $\mathcal{A}$. Adversary $\mathcal{A}$ first chooses challenge bit $b_{\mathcal{A}}$ randomly for game PrvSimPadap$_\ell$. Then $\mathcal{A}$ lets $f_1 \leftarrow f$ and uses A to find $f_0 \leftarrow A(\Phi(f))$. Then $\mathcal{A}$ sends f_0 and f_1 to its own GARBLE who returns (F, d). $\mathcal{A}$ sends (F, d) to $\mathcal{B}$.

Now, adversary $\mathcal{B}$ starts providing its argument bit by bit to adversary $\mathcal{A}$. When $\mathcal{A}$ is given argument bit x_j, it first sets $x_{1,j} \leftarrow x_j$. Then it uses algorithm A to determine the other argument bit $x_{0,j}$. After finding an appropriate $x_{0,j}$ $\mathcal{A}$ sends $x_{0,j}, x_{1,j}$ to its INPUT and gets $X_j^{x_{b,j}} = \mathtt{En}(e, x_{b,i}, i, r)$ as return. This is sent to $\mathcal{B}$ who now either gives a new argument bit or returns its answer $b_{\mathcal{B}}$ to the challenge. $\mathcal{A}$ does the same as $\mathcal{B}$, so it continues providing argument bits as long as $\mathcal{B}$ provides them or gives the answer to the challenge if $\mathcal{B}$ does so. The answer $b_{\mathcal{B}}$ is also the answer of adversary $\mathcal{A}$.

Let us now analyze the win probabilities of both adversaries. If $b = 1$, then function $f_1 = f$ becomes garbled. This is exactly the behavior we want, because in PrvSimPadap$_\ell$ game we garble the original function f in this case. If $b = 0$, then function f_0 becomes garbled. For adversary $\mathcal{B}$, this function seems simulated, as it should be if we had followed the actual PrvSimPadap$_\ell$ game. Because the answers of both adversaries are equal, also the win probabilities for both adversaries are equal, i.e.

$$\Pr[\mathcal{A} \text{ wins } | b = 1] = \Pr[\mathcal{B} \text{ wins } | b_{\mathcal{A}} = 1]$$
$$\Pr[\mathcal{A} \text{ wins } | b = 0] = \Pr[\mathcal{B} \text{ wins } | b_{\mathcal{A}} = 0]$$

From these we obtain

$$\Pr[\mathcal{A} \text{ wins }] = \tfrac{1}{2}\Pr[\mathcal{A} \text{ wins } | b = 1] + \tfrac{1}{2}\Pr[\mathcal{A} \text{ wins } | b = 0]$$
$$= \tfrac{1}{2}\Pr[\mathcal{B} \text{ wins} | b_{\mathcal{A}} = 1] + \tfrac{1}{2}\Pr[\mathcal{B} \text{ wins} | b_{\mathcal{A}} = 0] = \Pr[\mathcal{B} \text{ wins}].$$

This lets us derive that also the advantages for both adversaries are the same. Because adversary $\mathcal{A}$ has only negligible advantage, so has $\mathcal{B}$ against the constructed simulator $\mathcal{S}$. This completes the proof. □

V. Conclusion

Garbled circuits, or garbling schemes more generally, are a powerful tool to achieve secure multiparty computation, which is one of the methods applied in cloud computing. Garbling techniques must achieve a proper level of security and usability before they become practical enough to be used by masses. These both aspects of garbling schemes, different security measures and reusability, have recently gained a lot of attention in research. Several different security measures have been proposed, all of which can be presented as a combination of notion, model, level of adaptivity and level of reusability. The combination of the four properties determines the security class of a garbling scheme. Furthermore, the relations between different security classes build up a hierarchy which is currently quite comprehensively understood.

In this paper, we have studied the bitwise adaptive security of reusable garbling schemes. Adaptive security of reusable garbling schemes is indispensable for many applications, especially for secure outsourcing. However, this is not powerful enough for all applications. For example, one-time programs require not only adaptive security but also bitwise handling of the function argument, which is not a property of the adaptive security. Bitwise adaptive security has been first studied for one-time usable garbled functions and then extended to reusable garbling schemes. Still, there were some open questions related to classes of adaptively secure reusable garbling schemes which we have now solved.

First solved open question is related to the general existence of bitwise adaptively secure reusable garbling shemes: how long must the garbled argument be so that the scheme can belong to certain security class? In order to achieve a secure garbling scheme in bitwise adaptive setting, there are certain conditions which must be fulfilled. One of these requirements is related to the encryption algorithm $\mathtt{En}$. Namely, we prove in this paper that the length of the encrypted version of the argument x must be at least $nc \log k + n$, where n is the length of x, c a constant and k the security parameter, under the condition that a garbling scheme $\mathcal{G}$ is xxx.sim.padap$_2$ secure for xxx $\in \{$prv, mao, obv$\}$.

The second solved open question concerns the relations between the classes of bitwise adaptively secure reusable garbling schemes. Several results related to bitwise adaptively secure garbling schemes were obtained in [18]. However, it was left open whether there are conditions under which indistinguishability-based security could imply simulation-based security. In this paper, we provide such a condition by introducing a new variant for efficient invertibility of side-information function Φ and evaluation algorithm $\mathtt{ev}$ called bit- and componentwise efficient invertbility of $(\Phi, \mathtt{ev})$. The existence of a bit- and componentwise efficient $(\Phi, \mathtt{ev})$-inverter then ascertains, that privacy in indistingushability model implies privacy in simulation model.

References

[1] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "Seti@home: An experiment in public-resource computing," *Communications of the ACM*, vol. 45, no. 11, pp. 333–342, 2009.

[2] D. Beaver, S. Micali, and P. Rogaway, "The round complexity of secure protocols," *Proc. of the* 22^{nd} *STOC*, pp. 503–513, 1990.

[3] M. Bellare, V. T. Hoang, and P. Rogaway, "Adaptively secure garbling scheme with applications to one-time programs and secure outsourcing," *Asiacrypt 2012*, vol. 7685 of LNCS, pp. 134–153, 2012.

[4] ——, "Foundations of garbled circuits," in *Proc. of ACM Computer and Communications Security (CCS'12)*. ACM, 2012, pp. 784–796.

[5] M. Bellare and P. Rogaway, "Code-based game-playing proofs and the security of triple encryption," *Advances in Cryptology, Proc. of Eurocrypt 2006*, vol. 4004 of LNCS, pp. 409–426, 2006.

[6] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," *Proc. of TCC 2011*, vol. 6597 of LNCS, pp. 253–273, 2011.

[7] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verfiable computing: Outsourcing computation to untrusted workers." *CRYPTO 2010*, vol. 6223 of LNCS, pp. 465–482, 2010.

[8] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, 2009, crypto.stanford.edu/craig.

[9] S. Goldwasser, Y. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich, "Reusable garbled circuits and succinct functional encryption," *Proc. of the* 45^{th} *STOC*, pp. 555–564, 2013.

[10] S. Goldwasser, Y. Kalai, and G. Rothblum, "One-time programs," *CRYPTO2008*, vol. 5157 of LNCS, pp. 39–56, 2008.

[11] S. Gorbunov, V. Vaikuntanathan, and H. Wee, "Attribute-based encryption for circuits," *Proc. of the* 45^{th} *STOC*, pp. 545–554, 2013.

[12] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. of ACM Computer and Communications Security (CCS'06)*. ACM, 2006, pp. 89–98.

[13] W. A. Jansen, "Cloud hooks: Security and privacy issues in cloud computing," *Proc. of* 44^{th} *Hawaii International Conference on System Sciences (HICSS)*, pp. 1–10, 2011.

[14] Y. Lindell and B. Pinkas, "A proof of security of Yao's protocol for secure two-party computation," *Journal of Cryptology*, vol. 22(2), pp. 161–188, 2009.

[15] Mersenne Research Inc., "GIMPS home," http://www.mersenne.org/, accessed June 25, 2014.

[16] T. Meskanen, V. Niemi, and N. Nieminen, "Classes of garbling schemes," *Infocommunications Journal*, vol. V, no. 3, pp. 8–16, 2013.

[17] ——, "Hierarchy for classes of garbling schemes," *Proc. of Central European Conference on Cryptology (CECC'14)*, 2014.

[18] ——, "Hierarchy for classes of projective garbling schemes," *Proc. of International Conference on Information and Communications Technologies (ICT 2014)*, 2014.

[19] V. Pande, "Folding@home," http://folding.stanford.edu/, accessed June 25, 2014.

[20] C. Peikert, "Public-key cryptosystems from the worst-case shortest vector problem," *Proc. of the* 41^{st} *STOC*, pp. 333–342, 2009.

[21] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Proc. of the* 37^{th} *STOC*, pp. 84–93, 2005.

[22] H. Takabi, J. B. D. Joshi, and G.-J. Ahn, "Security and privacy challenges in cloud computing environments," *Security & Privacy*.

[23] A. Yao, "Protocols for secure computations," *Proc. of* 23^{rd} *SFCS, 1982*, pp. 160–164, 1982.

[24] ——, "How to generate and exchange secrets," *Proc. of* 27^{th} *FOCS, 1986.*, pp. 162–167, 1986.

Paper V

V

Garbling in Reverse Order

T. Meskanen and V. Niemi and N. Nieminen (2014). In *The 13th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-14)*, pages 53-60. IEEE

2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications

Garbling in reverse order

Tommi Meskanen*, Valtteri Niemi*†, Noora Nieminen*

*Department of Mathematics and Statistics, University of Turku, 20014 Turun yliopisto, FINLAND
†Xidian University Xi'an, CHINA
{tommes, pevani, nmniem}@utu.fi

***Abstract*—Modern computing has increasingly been moved into network-based solutions such as clouds. The security in these solutions is not always at satisfactory level. Ideas to improve the security of networked computing include the technique called *garbled circuits* or *garbling*, a technique first introduced by Yao and then formalized by Bellare, Hoang and Rogaway. The security of garbling has been considered from different perspectives, including *adaptivity* and *reusability*. In this paper, we improve the practicality of garbling by presenting a new type of adaptivity with support to reusability. We also show how this new type of adaptivity is related to the known security concepts.**

***Keywords*—secure multiparty computations, garbling schemes, adaptive security, reusability**

I. Introduction

Recent trends in computer science show that computations become more and more network-based. We use Internet-based solutions for storing huge amounts of data and running programs on clouds - there are even anti-virus software that are entirely run on cloud [1]. Naturally, all this sets requirements to the solutions used in networked computation. Especially information security and privacy are important aspects in the design of network-based computation. Cryptography provides powerful tools to achieve information security and privacy. In the context of networked computation, methods for secure multiparty computations provide a solution to reach the necessary security goals.

Secure multiparty computation has been a vivid research area in cryptography since Yao introduced the concept of *secure computation* in [16]. Yao also took the next step towards secure multiparty computations by proposing a method called *garbled circuits* [17] in the case of two parties. Beaver, Micali and Rogaway generalized Yao's protocol into a multiparty protocol with constant round complexity in [2]. Since then Yao's garbled circuits have been used for various purposes in spite of the fact that formal definition of what is meant by garbling has been missing, let alone a proof of security. The first proof of security related to garbled circuits was introduced by Lindell and Pinkas in [13]. However, this proof showed a custom protocol using garbled circuits to be secure against semi-honest adversaries, not the security of garbling itself.

The tools to study the security of garbling were established when Bellare, Hoang and Rogaway proposed a formal definition of *a garbling scheme* which is a collection of algorithms run in certain order to model garbled evaluation [4]. More formally, a garbling scheme is a 5-tuple $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev})$ of algorithms, where algorithm $\mathtt{Gb}$ is probabilistic and the other four are deterministic. The last component in the 5-tuple is the evaluation algorithm $\mathtt{ev}$. The evaluation algorithm takes f and x as inputs, where f is the original function represented as a string and x is the argument for f. Algorithm $\mathtt{ev}$ computes the function value $\mathtt{ev}(f, x)$ and this process is what we want to garble. The four first algorithms take care of the garbling. The first component $\mathtt{Gb}$ is the garbling algorithm. It takes f and 1^k as inputs, where $k \in \mathbb{N}$ is a security parameter, and returns (F, e, d) as output. The encryption algorithm $\mathtt{En}$ maps an initial argument x to a garbled argument $X = \mathtt{En}(e, x)$ by using the key e. The garbled evaluation function $\mathtt{Ev}$ takes the garbled function F and the garbled argument X as inputs and returns the garbled function value $Y = \mathtt{Ev}(F, X)$. Finally, the decryption algorithm $\mathtt{De}$ decrypts Y using the key d to final function value $\mathtt{De}(d, Y) = y = \mathtt{ev}(f, x)$.

Using this definition for a garbling scheme, Bellare et al. propose several notions for security of a garbling scheme in [4]. These notions share a common characteristic - they are all static in the sense that both the function f to be evaluated and the argument x are to be fixed at the same time before garbling. However, many applications, including one-time programs [10] and secure outsourcing [7], require a more adaptive setting - it should be possible to choose the function f independently before the argument x. To respond to these new requirements of adaptivity, Bellare et al. extended their security notions to adaptive setting in [3]. They also showed how to efficiently transform garbling schemes achieving static security to garbling schemes achieving adaptive security.

There is still one possible variant of adaptivity which has not yet been considered. In many realistic situations, the argument x is known before f. As an example, consider the following scenario. Suppose that some amount of personal data is collected into a database, denoted by x. Then, following the cloud computing paradigm, different operations, e.g. searches, comparisons, optimizations, are applied to x by the cloud. Now function f changes between operations but the argument remains the same. If the contents of x is private then garbling is needed. However, the established garbling techniques do not recognize this new setting where we want to evaluate the same argument x using different functions f. In this paper, our aim is to fill this gap in the definition of adaptivity by defining the concept of a garbling scheme newly. This new definition then allows us to generalize the definition for adaptive security. We call this new variant *reverse-order adaptivity*.

As explained above, Bellare et al. define a garbling scheme as a 5-tuple $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev})$ of algorithms [4].

978-1-4799-6513-7/14 $31.00 © 2014 IEEE
DOI 10.1109/TrustCom.2014.11

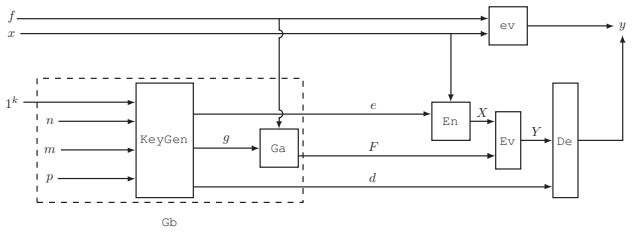


Figure 1. Generalizing the concept of a garbling scheme. The diagram shows that the final value obtained via garbling must coincide the final value obtained by direct evaluation, i.e. $\mathtt{ev}(f,x) = y = \mathtt{De}(d,Y)$ where F is the garbled function, $Y = \mathtt{Ev}(F,X)$ the garbled value and $X = \mathtt{En}(e,x)$ the garbled argument. Compared to the defintion in [4], the diagram is the same except of the garbling algorithm Gb, which in our model is split to two separate algorithms KeyGen and Ga.

Now we analyze the algorithms in more details, especially the garbling algorithm Gb. Algorithm Gb consists of two functionalities. Firstly, Gb prepares the garbled function F based on the security parameter and f. Secondly, Gb generates encryption and decryption keys (e,d) which are used to encrypt the argument x and decrypt the garbled function value Y. When considering static or adaptive security, it is natural to treat these both functionalities at the same time in one algorithm: in both cases, the function f is first garbled to F together with creation of keys (e,d) and only after this we encrypt x and proceed to garbled evaluation Ev. However, this is not the case in the new version of adaptivity. The garbling scheme should first be able to generate the key to encrypt x and only then prepare the garbled function F. This gives the motivation to split the two functionalities in the garbling algorithm Gb to two separate algorithms KeyGen and Ga.

Let us first describe the key generation algorithm KeyGen. KeyGen takes the security parameter 1^k and three other parameters, m, n and p as its inputs and outputs a triplet of keys (g,e,d). The parameter $m = |x|$ is the length of the argument x, $n = |y|$ the length of the function value y and $p = |f|$ the length of the string representation of function f. The reason to give these parameters as input to KeyGen is motivated as follows. First of all, the garbling algorithm Gb in [4] takes f as input, and, according to [4], parameters m, n and p must be efficiently computable from f. Moreover, the garbling scheme of [4] must fulfill the *non-degeneracy* property. Non-degeneracy property tells that (e,d) may depend only on k, m, n and p, not on f. In particular, the garbling algorithm Gb in [4] is allowed to use m, n and p to create (F,e,d). This property is now built in our generalized model by explicitly using m, n and p as inputs to the key generation algorithm KeyGen. All four parameters are also necessary to KeyGen in order to create valid encryption and decryption keys in the sense that the garbled evaluation gives the correct function value $\mathtt{De}(d,\mathtt{Ev}(F,\mathtt{En}(e,x))) = y = \mathtt{ev}(f,x)$.

The other algorithm Ga takes the first key g from the triplet (g,e,d) generated by KeyGen as its input. Based on g and the function f, the algorithm Ga generates the garbled function F. The two other keys, e and d created by KeyGen are used by algorithms En and De. e to encrypt x to $X = \mathtt{En}(e,x)$. Then garbled evaluation algorithm is called to prepare the garbled function value $Y = \mathtt{Ev}(F,X)$. Finally, d is used to decrypt the garbled function value Y to the final function value $y = \mathtt{De}(d,Y)$.

To conclude, we present the garbling scheme as 6-tuple of algorithms $(\mathtt{KeyGen},\mathtt{Ga},\mathtt{En},\mathtt{De},\mathtt{Ev},\mathtt{ev})$. Here, the algorithm KeyGen is probabilistic. Algorithms Ga and En can be either deterministic or probabilistic. The four other algorithms, En, De, Ev and ev are deterministic. Note, that everything after the garbling phase (Gb split to KeyGen and Ga) in our definition remains the same as in the definition of [4]. Moreover, following a basic assumption in [3], parameters m, n, p can be found from f. Therefore, a garbling scheme according to our new definition is a garbling scheme according to definitions of [3], [4]. On the other hand, any garbling scheme according to old definition (of [3], [4]) can be converted to a garbling scheme according to the new definition by running the Gb algorithm twice in sequel. From the first instance of Gb we get e and d while using any function f' as input such that m, n. p are the same as for f. Because of non-degeneracy property (e and d are independent from the function), e and d are the same as when using f as input. This first instance of Gb becomes now the algorithm KeyGen. This algorithm is non-deterministic, hence it uses some additional random input r. This r together with 1^k forms now the parameter g. The second run of Gb uses r for non-determinism and it produces F as output. This second instance of Gb is now algorithm Ga.

Our contributions. In all currently existing security notions, the function to be evaluated has to be fixed at the same time or even before the argument. In many practical situations, the order of fixing f and x is reversed, i.e. argument is known before the corresponding function. Garbling schemes using currently known security notions do not recognize the situation described above. To overcome this problem, we generalize the adaptive security notions for garbling schemes by allowing the evaluator to fix argument x before function f. This requires generalization of the concept of garbling scheme as explained above.

Using these new ideas we are able to define a new type of adaptivity which we call *reverse-order* adaptivity. We define several new security classes achieving this new type of adaptivity. In section II, we start by basic definitions related to garbling schemes. In section III, we present the definition

of the new type of adaptive security. In section IV, we provide results showing the relations between the new security classes as well as the relation to the known security classes. Section V is the summary of results presented in this paper.

II. Definitions

In this section, we provide the basic definitions and notations. A *string* is a finite sequence of bits. Let S be a finite set. Notation $y \leftarrow S$ means that an element is selected uniformly at random from the set S, and this element is assigned as a value to the variable y. An algorithm is denoted by A and notation $A(x_1, \ldots, x_n)$ refers to the output of the algorithm A on inputs $x_1, \ldots, x_n$. We say that a function $f : \mathbb{N} \to \mathbb{R}$ is negligible if for every $c > 0$ there is an integer N_c such that $|f(n)| < n^{-c}$ for all $n > N_c$.

A. Code-based games

The proofs in this paper are heavily based on *code-based games*. Following the terminology presented in [5], a game is a collection of procedures called *oracles*. This collection contains three types of procedures: `INITIALIZE`, `FINALIZE` and other named oracles. We assume throughout this paper that procedures `INITIALIZE` and `FINALIZE` are compulsory in a game, whereas other procedures are optional.

The entity playing a game is called *adversary*. Every game starts with `INITIALIZE` procedure. Then the adversary may invoke other procedures before feeding its output to the `FINALIZE` procedure. Based on the input from the adversary `FINALIZE` creates a string that tells the outcome from the game typically consisting of one bit of information: whether the adversary has won or not.

B. Side-information function

A side-information function is intended to model what is revealed about evaluation (with f and x) even when garbling is used. More formally, a side-information function Φ deterministically maps f to $\Phi(f)$. For example, in the case of computation model using circuits and the corresponding evaluation algorithm $\mathtt{ev}_{circ}$, the value $\Phi(f)$ might be the size of the circuit that was garbled, the topology of the circuit or something else - even the whole initial circuit.

C. Security concepts for garbling schemes

In this section we shortly describe the security concepts for garbling schemes following the terminology used in [4], [3], [14], [15]. According to [15], garbling schemes can be characterized by three components: *security notion*, *security model* and *level of adaptivity*. Next paragraphs contain a short description from each of these three components.

Security notion tells how much information is allowed to be leaked during the garbled evaluation. The notions are *privacy*, *oblviousness* and *matchability-only*. A garbling scheme achieving **privacy (prv)** does not allow f or x to be leaked, but $f(x)$ can be leaked. A garbling scheme achieving **obliviousness (obv)** keeps all, f, x and $f(x)$ secret. A **matchability-only (mao)** secure garbling scheme hides f, x and $f(x)$ but when computing $y_1 = f(x_1)$ and $y_2 = f(x_2)$ the scheme is allowed to leak whether $y_1 = y_2$.

Security model is either *simulation-* or *indistinguishability-based*. **Indistinguishability (ind)** refers to inability to distinguish garblings of two similar functions from each other. In **simulation-based model (sim)**, refers to inability to distinguish a real garbling from simulated information mimicking the real garbling.

Level of adaptivity describes how function f and argument x may be treated during the garbled evaluation. There are several levels of adaptivity: *static*, *adaptive*, *bitwise adaptive* and *reverse-order adaptive* security. In **static (stat)** security, both f and x need to be fixed at once and must not be altered afterwards. **Adaptive (adap)** security, also known as *coarse-grained security*, allows f and x to be handled separately from each other, meaning that f can be fixed before x. **Bitwise adaptive (padap)** security, also known as *fine-grained security*, refines adaptive security by allowing the argument to be handled bitwise. The new flavor af adaptivity presented in this paper, **reverse-order adaptivity (radap)** allows f and x be handled separately so that x can be fixed before f.

The adaptivity levels $stat$ and $adap$ have been considered in more detail by Bellare et al. in [3], [4]. Unfortunately, the notions used by Bellare et al. have one drawback - they allow only single-use of the garbled function. To perform a garbled evaluation for the same function f for a new argument requires every time a new garbling. This regarbling becomes inefficient before long. To overcome this deficiency, garbling schemes supporting reusable garbled functions have been investigated by several researchers. Goldwasser, Kalai, Popa Vaikunathan and Zeldovich were the first to propose a reusable garbled circuit in [9]. The reusable garled circuit uses *fully homomorphic encryption* (FHE, [8]), *attribute-based encryption* [12], [11] and *functional encryption* [6] as its building blocks. While being reusable, the garbled circuit constructed by Goldwasser et al. fails to be practical due to the impracticality of FHE. To fill the gap of practical garbling schemes achieving at least some level of reusability, Meskanen, Niemi and Nieminen have proposed several new notions for adaptive security for reusable garbling schemes in [14], [15]. These security notions include an additional parameter ℓ telling the level of reusability: The parameter ℓ tells how many garbled arguments can be securely obtained using the same garbled function. This parameter can get values $\ell \in \mathbb{N}$ or $\ell = *$, where the latter means that arbitrarily many garbled arguments can be processed using the same garbled function. Using this reusability parameter ℓ, we build the possibility of reusable garbled functions in to the new $radap$ security classes. Using reusability parameter value $\ell \geq 2$ sets additional requirements for the garbling scheme. In the case of adaptive security, at least the encryption algorithm `En` needs to be non-deterministic [14]. In the case of reverse-order adaptive security, at least the garbling algorithm `Ga` needs to be non-deterministic.

Notations. Every security class is determined by these three components: security notion, security model and level of adaptivity. We use notation $xxx.yyy.zzz$, where $xxx \in \{\text{prv, obv, mao}\}$, $yyy \in \{\text{sim, ind}\}$ and $zzz \in \{\text{stat, adap}_\ell, \text{padap}_\ell, \text{radap}_\ell\}$, for a $xxx.yyy.zzz$ secure gar-

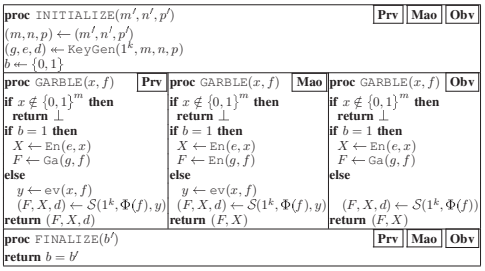


Table I. THE GAMES DEFINING THE PrvSimRstat, MaoSimRstat AND ObvSimRstat SECURITY NOTIONS. THE GAME DEPENDS ON THE CHOICE OF $\mathcal{G}$, Φ,$\mathcal{S}$.

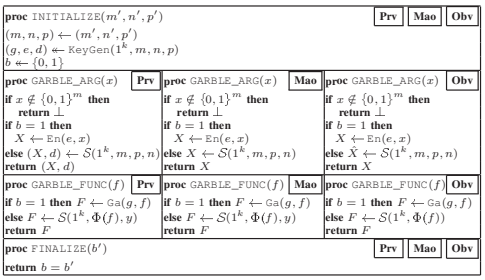


Table III. THE GAMES DEFINING THE PrvSimRadap$_\ell$, MaoSimRadap$_\ell$ AND ObvSimRadap$_\ell$ SECURITY NOTIONS. THE GAME DEPENDS ON THE CHOICE OF $\mathcal{G}$, Φ AND $\mathcal{S}$. INTEGER ℓ TELLS THE UPPER LIMIT FOR THE NUMBER OF GARBLE_FUNC QUERIES.

bling scheme. The corresponding security class of garbling schemes is denoted by $\mathtt{GS}(xxx.yyy.zzz)$.

To determine whether a garbling scheme belongs to a certain security class, we need a security game, which is denoted by $XxxYyyZzz$ for $Xxx \in \{\text{Prv, Obv, Mao}\}$, $Yyy \in \{\text{Sim, Ind}\}$ and $Zzz \in \{\text{Stat, Adap}_\ell, \text{Padap}_\ell, \text{Radap}_\ell\}$. The game is played by an adversary, who wins the game with certain probability. If the win probability is greater than $\frac{1}{2}$ then the adversary has an advantage over pure guessing. If this advantage in turn happens to be non-negligible, then the garbling scheme cannot be secure in the sense determined by the game. More formal definitions of advantage and security are given below whereas the games themselves are explained in the next section.

Advantage of an adversary. We define the advantage of an adversary A in all games as follows:

$$\mathbf{Adv}(\mathcal{A}, k) = 2 \cdot \Pr[\mathcal{A} \text{ wins the game }] - 1.$$

Security of a garbling scheme. If the advantage function $\mathbf{Adv}(\mathcal{A}, \cdot)$ is negligible for all polynomial-time (PT) adversaries $\mathcal{A}$ in XxxIndZzz$_{\mathcal{G},\Phi}$ game then we say that the garbling scheme $\mathcal{G}$ is *xxx.ind.zzz secure over* Φ. Similarly, we define that a garbling scheme $\mathcal{G}$ is *xxx.sim.zzz secure over* Φ if for every PT adversary $\mathcal{B}$ there exists a PT simulator $\mathcal{S}$ such that the advantage $\mathbf{Adv}(\mathcal{B}, k)$ in the XxxSimZzz$_{\mathcal{G},\Phi,\mathcal{S}}$ game is negligible.

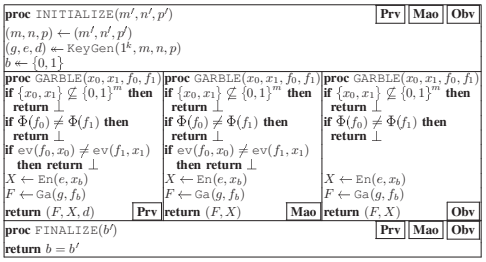


Table II. THE GAMES DEFINING THE PrvIndRstat, MaoIndRstat AND ObvIndRstat GAMES. THE GAME DEPENDS ON THE CHOICE OF $\mathcal{G}$ AND Φ.

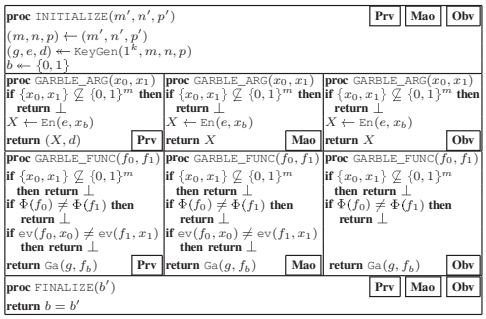


Table IV. THE GAMES DEFINING THE PrvIndRadap$_\ell$, MaoIndRadap$_\ell$ AND ObvIndRadap$_\ell$ GAMES. THE GAME DEPENDS ON THE CHOICE OF $\mathcal{G}$ AND Φ. INTEGER ℓ TELLS THE UPPER LIMIT FOR NUMBER OF GARBLE_FUNC QUERIES.

III. GAMES FOR REVERSE-ORDER ADAPTIVE SECURITY

In previous section we described the central concepts related to security of garbling schemes in general. From now on, we concentrate on the new security classes inspired by the new type of security, *reverse-order adaptive* security. We first define the security games related to these security classes and describe how the games are used to achieve a certain type of security.

Tables I to IV define the games for *reverse-order adaptive* security classes of garbling schemes. By *reverse-order* we mean that the adversary first specifies the argument x that is to be evaluated on a function f determined later, oppositely to the notions defined in [4], [3], [14], [15].

All games start with procedure INITIALIZE taking three integers m', n', p' as input from adversary, where m', n' and p' give upper limits for lengths of x, y and f. The procedure prepares the keys (g, e, d) based on (m', n', p') and chooses the challenge bit b uniformly at random. Procedure GARBLE_ARG prepares garbled argument X to the adversary based on the argument x chosen by the adversary. Procedure GARBLE_FUNC encrypts the argument f if the argument x is compatible for f. These two procedures are slightly different for the different security notions: GARBLE_ARG is the same for notions Mao and Obv, whereas GARBLE_FUNC is the same for Prv and Mao. FINALIZE procedure is again the same for all notions

- the procedure tells whether the adversary has predicted the challenge bit b correctly. Next, we briefly discuss the difference of procedures GARBLE_ARG and GARBLE_FUNC for different security notions and models.

GARBLE_ARG: : Consider first the adaptive simulation-based notions (see table III). In all three games, the choice of the challenge bit b determines whether the encryption of x is constructed using the actual garbling algorithm ($b = 1$) or devised by a simulator ($b = 0$). If the real garbling is used, KeyGen is first called to create the keys (g, e, d), after which x is encrypted with e to $X = \mathtt{En}(e, x)$. The other possibility is that a simulator $\mathcal{S}$, based on 1^k, m, n and p, creates X that resembles the original garbled argument. In game PrvSimRadap$_\ell$ the adversary gets garbled argument X and decryption key d as an output from procedure GARBLE_ARG, whereas in games MaoSimRadap$_\ell$ and ObvSimRadap$_\ell$ the adversary gets only X.

Then consider the adaptive indisinguishability-based model (see table IV). In all three games, procedure GARBLE_ARG takes two arguments, x_0 and x_1, as its inputs. The procedure first checks whether the lengths of x_0 and x_1 are the same. If they are not, the procedure is exited with value $\perp$ telling the adversary that the query has failed and no output was produced. Based on the challenge bit, the procedure encrypts either x_0 or x_1. In game PrvIndRadap$_\ell$, the adversary gets X and d, whereas in MaoIndRadap$_\ell$ and ObvIndRadap$_\ell$ games the adversary gets solely X.

GARBLE_FUNC: : Let us again first consider the simulation-based notions (see table III). The procedure prepares an encrypted function F based on the choice of the challenge bit b: the encryption is performed using the actual encryption algorithm ($b = 1$) or devised by a simulator ($b = 0$). In all three games, the procedure returns F.

Then consider the indisinguishability-based model (see table IV). In all three games, procedure GARBLE_FUNC takes two functions, f_0 and f_1, as its inputs. The procedure first checks that the arguments x_0, x_1 are of correct length n. Then the procedure checks whether $\Phi(f_0) = \Phi(f_1)$ and $\mathtt{ev}(f_0, x_0) = \mathtt{ev}(f_1, x_1)$. If some of these tests fail, the procedure is exited with return value $\perp$, telling the adversary that the query has failed and no output is provided. If the tests are passed, then the procedure returns $F = \mathtt{Ga}(g, f_b)$.

The games for static security notions are given in tables I and II. The static security games consist of INITIALIZE, GARBLE and FINALIZE. The procedure GARBLE works as if procedures GARBLE_ARG and GARBLE_FUNC were combined: the garbled argument X and function F are both prepared and returned to the adversary one go. The differences between the notions for static security are the same as for reverse-order adaptive security.

We are now ready to provide the first results related to the reverse-order adaptive security classes. We begin with results comparing the new $radap$ security classes to the corresponding $adap$ classes. The next theorem shows that static security(see [4] for the definition) is not affected by reversing the order of f and x.

Theorem 1: For any side-information function Φ and for any evaluation function ev, we have the equality $\mathtt{GS}(\text{xxx.yyy.stat}, \Phi) \bigcap \mathtt{GS}(\mathtt{ev}) = \mathtt{GS}(\text{xxx.yyy.rstat}, \Phi)) \bigcap \mathtt{GS}(\mathtt{ev})$ holds for xxx $\in$ {prv. mao, obv}, yyy $\in$ {sim, ind}.

Proof: The claim follows from the fact that the security games for classes $rstat$ and $stat$ are similar except of the way of generating the output $(F, X, d)/(F, X)$. However, algorithm KeyGen always obtains the values (m, n, p) directly from f or (f_0, f_1) just like Gb. Therefore, there is no difference whether the garblings were created by running solely Gb or consecutively running KeyGen and Ga. □

In the next theorem we show how to transform an adaptively secure garbling scheme into a reverse-order adaptively secure garbling scheme. For the proof, we define the *dual* $\widehat{\mathcal{G}}$ of the garbling scheme $\mathcal{G}$ as follows. Let $\mathcal{G} = (\mathtt{KeyGen}, \mathtt{Ga}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev})$ be a garbling scheme. The dual of $\mathcal{G}$ is the garbling scheme $\widehat{\mathcal{G}} = (\widehat{\mathtt{KeyGen}}, \widehat{\mathtt{Ga}}, \widehat{\mathtt{En}}, \mathtt{De}, \widehat{\mathtt{Ev}}, \widehat{\mathtt{ev}})$, where $\widehat{\mathtt{KeyGen}}$, $\widehat{\mathtt{Ga}}$, $\widehat{\mathtt{En}}$, $\widehat{\mathtt{Ev}}$ and $\widehat{\mathtt{ev}}$ are defined as follows. If KeyGen returns (g, e, d) on input $(1^k, m, n, p)$, then $\widehat{\mathtt{KeyGen}}$ returns (e, g, d) on input $(1^k, p, n, m)$. We also define $\widehat{\mathtt{Ga}} = \mathtt{En}$ and $\widehat{\mathtt{En}} = \mathtt{Ga}$. The dual of Ev is the garbled evaluation function $\widehat{\mathtt{Ev}}(X, F) = Y = \mathtt{Ev}(F, X)$ and the dual of ev is the function $\widehat{\mathtt{ev}}(x, f) = y = \mathtt{ev}(f, x)$.

Theorem 2: Let xxx $\in$ {prv, mao, obv}, yyy $\in$ {sim, ind} and $\Phi(f) = \Phi_{min}(f) = (m, n, p)$ for any function f. Then, for any $\ell \in \mathbb{N}$ the following equivalence holds: $\mathcal{G} \in \mathtt{GS}(\text{xxx.yyy.adap}_\ell, \Phi) \bigcap \mathtt{GS}(\mathtt{ev})$ if and only if $\widehat{\mathcal{G}} \in \mathtt{GS}(\text{xxx.yyy.radap}_\ell, \widehat{\Phi}_{min}) \bigcap \mathtt{GS}(\widehat{\mathtt{ev}})$ where $\widehat{\mathcal{G}}$ and $\widehat{\mathtt{ev}}$ are duals of $\mathcal{G}$ and ev as explained above and side-information function $\widehat{\Phi}_{min}$ maps x to (p, n, m).

Proof: We prove the claim for simulation-based privacy, i.e for xxx=prv and yyy=sim. The other cases can be proven in similar manner. Let $\mathcal{G} = (\mathtt{KeyGen}, \mathtt{Ga}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev})$ be a garbling scheme and $\widehat{\mathcal{G}} = (\widehat{\mathtt{KeyGen}}, \widehat{\mathtt{Ga}}, \widehat{\mathtt{En}}, \mathtt{De}, \widehat{\mathtt{Ev}}, \widehat{\mathtt{ev}})$ its dual. Assume that $\widehat{\mathcal{G}}$ is prv.sim.radap$_\ell$ secure. Our aim is to prove that then $\mathcal{G}$ is prv.sim.adap$_\ell$ secure. To do that, let $\mathcal{A}$ be an arbitrary adversary playing the security game PrvSimAdap$_\ell$ related to the garbling scheme $\mathcal{G}$ and side-information function Φ_{min}. Side-information function Φ_{min} maps f to (m, n, p) where m is the length of x, n the length of $y = \mathtt{ev}(f, x)$ and p the length of f. We construct another adversary $\mathcal{B}$ playing the PrvSimRadap$_\ell$ game related to garbling scheme $\widehat{\mathcal{G}}$ and side-information function $\widehat{\Phi}_{min}(x) = (p, n, m)$. Adversary $\mathcal{B}$ uses $\mathcal{A}$ as its subroutine as explained next. Figure 2 illustrates the situation.

When the PrvSimRadap$_\ell$ game starts, adversary $\mathcal{B}$ is expected to provide parameters m, n and p to INITIALIZE procedure. Instead of sending m, n and p to its INITIALIZE, adversary $\mathcal{B}$ challenges adversary $\mathcal{A}$ to play game PrvSimAdap$_\ell$ game. Adversary $\mathcal{A}$ presumes to play the original, not emulated, PrvSimAdap$_\ell$ game and makes its query to GARBLE by sending function f to $\mathcal{B}$. Adversary $\mathcal{B}$ emulates GARBLE procedure, denoted by EMUL_GARBLE in fig. 2, as follows. $\mathcal{B}$ first computes (m', n', p') by using the side-information function Φ_{min} and sends (p', n', m') to INITIALIZE. In

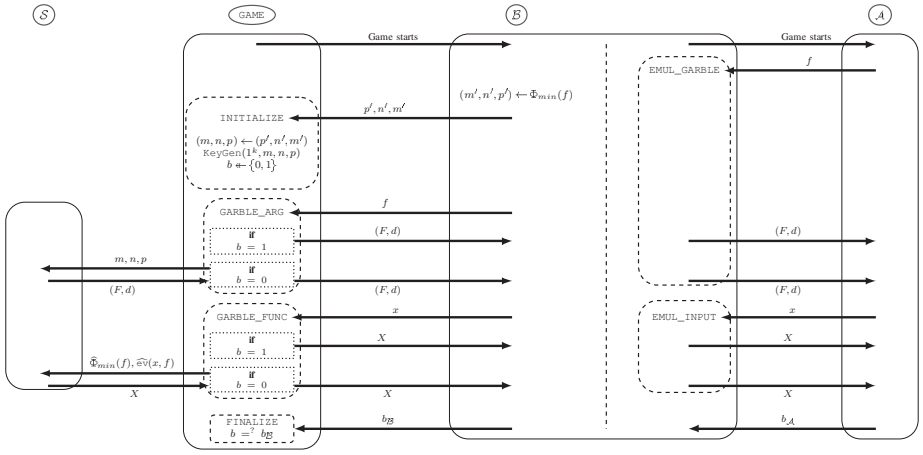


Figure 2. In the above figure, $\mathcal{B}$ is an adversary playing the game PrvSimRadap$_\ell$ denoted in the diagram by GAME. $\mathcal{S}$ is the simulator in this game. $\mathcal{A}$ is an adversary presuming to play game PrvSimAdap$_\ell$, but actually adversary $\mathcal{B}$ uses $\mathcal{A}$ as a subroutine by trying to emulate the actual game PrvSimAdap$_\ell$.

INITIALIZE procedure the key generator creates keys based on $m = p'$, $n = n'$ and $p = m'$, i.e. $\widehat{\mathtt{KeyGen}}(1^k, p, n, m) = (e, g, d)$. Also the challenge bit for game PrvSimRadap$_\ell$ is chosen. After this is done, $\mathcal{B}$ queries GARBLE_ARG with input f. Now, based on the challenge bit b, the procedure GARBLE_ARG creates (F, d) either with real algorithms or with simulator $\mathcal{S}$. If $b = 1$, then procedure GARBLE_ARG uses $\widehat{\mathtt{En}} = \mathtt{Ga}$ to create $\widehat{X} = F = \mathtt{Ga}(g, f)$ and constructs return value $(\widehat{X}, d) = (F, d)$. This means that adversary $\mathcal{B}$ is able to emulate correctly procedure GARBLE in PrvSimAdap$_\ell$ game, because in this case the algorithm Ga is used to create (F, d). If $b = 0$, then simulator $\mathcal{S}$ creates (F, d) based on $(m, n, p) = (p', n', m')$. Also in this case $\mathcal{B}$ correctly emulates GARBLE for $\mathcal{A}$. In PrvSimAdap$_\ell$ game, the simulator gets input $(1^k, \Phi(f))$ and in this proof we use $\Phi(f) = \Phi_{min}(f) = (m, n, p)$. In game PrvSimRadap$_\ell$ the simulator has input $(1^k, p', n', m')$, i.e. exactly the same information as in PrvSimAdap$_\ell$ game. Regardless of the choice of the challenge bit, procedure GARBLE_ARG returns (F, d) to adversary $\mathcal{B}$. Adversary $\mathcal{B}$ forwards (F, d) to $\mathcal{A}$, who is now expected to make a query to INPUT.

Adversary $\mathcal{A}$ queries INPUT with argument x by sending it to $\mathcal{B}$. Then adversary $\mathcal{B}$ makes its GARBLE_FUNC query with the same input x. Again, based on challenge bit b either algorithms of the garbling scheme $\widehat{\mathcal{G}}$ are used or the simulator $\mathcal{S}$ is called to construct garbled function $\widehat{F}$. If $b = 1$ then the algorithm $\widehat{\mathtt{Ga}} = \mathtt{En}$ is used to create $\widehat{F} = X = \mathtt{En}(e, x)$. This is the correct behavior for emulated INPUT procedure, because in the case $b = 1$ the algorithm En is used to encrypt x to garbled argument X. In the case $b = 0$, $\widehat{F} = X$ is constructed by the simulator based on input $\widehat{\Phi}_{min}$ and $\widehat{\mathtt{ev}}(x, f)$. Also in this case $\mathcal{B}$ emulates INPUT correctly. Simulator's input $(y, \Phi_{min}(f))$ in PrvSimAdap$_\ell$ game is exactly the same as in PrvSimRadap$_\ell$ game: $\mathtt{ev}(f, x) = y = \widehat{\mathtt{ev}}(x, f)$ and $\Phi_{min}(f) = (m, n, p)$ contains the same information as $\widehat{\Phi}_{min}(x)$. Regardless of the challenge bit, adversary $\mathcal{B}$ gets X as return value from GARBLE_FUNC. $\mathcal{B}$ forwards this to $\mathcal{A}$ who now can make a new INPUT query or proceed to FINALIZE.

After these steps the adversary $\mathcal{A}$ has collected at most ℓ garbled inputs X for the garbled function F in both games. Based on this information, the adversary answers $b_{\mathcal{A}}$ and sends this to $\mathcal{B}$. Adversary $\mathcal{B}$ answers similarly, $b_{\mathcal{B}} = b_{\mathcal{A}}$. Procedure FINALIZE now tells both adversaries, whether their answer was correct.

Let us now analyze the win probability of both adversaries in the two games. The win probability of $\mathcal{A}$ in game PrvSimAdap$_\ell$ is the same as the win probability of $\mathcal{B}$ in game PrvSimRadap$_\ell$ because the answers of $\mathcal{A}$ and $\mathcal{B}$ are the same. This means that if the advantage of $\mathcal{A}$ in game PrvSimAdap$_\ell$ is non-negligible then adversary $\mathcal{B}$ has the same non-negligible advantage in game PrvSimRadap$_\ell$. But this contradicts the fact that $\widehat{\mathcal{G}}$ is prv.sim.radap$_\ell$ secure. It follows that the garbling scheme $\mathcal{G}$ must be prv.sim.adap$_\ell$ secure. Using similar deduction we also get the converse claim: if $\widehat{\mathcal{G}} \in \mathtt{GS}(\text{prv.sim.radap}_\ell)$ then $\mathcal{G} \in \mathtt{GS}(\text{prv.sim.adap}_\ell)$. □

IV. Building up the hierarchy for the new security classes

In this section we prove relations regarding the classes of reverse-order adaptive garbling schemes defined in the previous section. First we study how the reusability parameter ℓ affects to the reverse-order adaptive security. Then we show that adaptive security does not imply reverse-order adaptive security and vice versa for $\ell \geq 2$.

Theorem 3: Let $\mathcal{K}_\ell$ be a class of reverse-order adaptively secure garbling scheme, i.e. $\mathcal{R}_\ell = \mathtt{GS}(\text{xxx.yyy.adap}_\ell, \Phi)$ for xxx $\in$ {prv, mao, obv} and yyy $\in$ {ind, sim} and $\ell \in \mathbb{N}$. Then $\mathcal{K}_{\ell+1} \subsetneq \mathcal{K}_\ell$ or $\mathcal{K}_{\ell+1} = \mathcal{K}_\ell = \emptyset$.

Proof: The claim $\mathcal{R}_{\ell+1} \subseteq \mathcal{R}_\ell$ is fairly obvious, because the adversary playing the game related to class $\mathcal{R}_{\ell+1}$ can always use only ℓ GARBLE_FUNC calls instead of the maximum allowed number of calls.

Next we need to prove that in the case $\mathcal{K}_\ell \neq \emptyset$ there exist a garbling scheme that belongs to class $\mathcal{K}_\ell$ but does not belong to class $\mathcal{K}_{\ell+1}$. Let $\mathcal{G} = (\mathtt{KeyGen}, \mathtt{Ga}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev})$ be a garbling scheme in $\mathcal{K}_\ell$. We construct another garbling scheme $\mathcal{G}' = (\mathtt{KeyGen}', \mathtt{Ga}', \mathtt{En}', \mathtt{De}, \mathtt{Ev}', \mathtt{ev})$ by modifying four components of $\mathcal{G}$, namely KeyGen, Ga, En and Ev.

Let us start with the modified KeyGen algorithm. The algorithm creates keys $((g, g_2, t \text{ shares of key } g_2), e, d)$. The first component $(g, g_2, t$ shares) consists of the following components. The first key g is used by the garbling algorithm to construct F. The second component is a key g_2 that belongs to a completely independent symmetric, non-deterministic, secure encryption scheme $\mathtt{En}_2$. This key g_2 is used to encrypt f. Finally, the third component is a secret sharing scheme that creates t shares of key g_2 in such a way that $\ell + 1$ shares are needed to reconstruct the key g_2.

After generating the keys with KeyGen, the argument is encrypted with the modified encryption algorithm $\mathtt{En}'$. The algorithm takes $((e, g_2), x)$ as its inputs and returns (X, X_2) where $X = \mathtt{En}(e, x)$ and $X_2 = \mathtt{En}_2(g_2, x)$.

The third modification is targeted to garbling algorithm Ga. On input $((g, g_2, t \text{ shares}), f)$ the algorithm outputs (F, F_2, s) where $F = \mathtt{Ga}(g, f)$, $F_2 = \mathtt{En}_2(g_2, f)$ and s is a random share of key g_2.

The modified garbled evaluation algorithm $\mathtt{Ev}'$ takes $((F, F_2, s), (X, X_2))$ as its inputs. The algorithm omits the new parts F_2, X_2, s and computes $Y = \mathtt{Ev}'((F, F_2, s), (X, X_2)) = \mathtt{Ev}(F, X)$.

Our aim is now to prove that the constructed garbling scheme $\mathcal{G}'$ belongs to class $\mathcal{K}_\ell$ but not to class $\mathcal{K}_{\ell+1}$. The first claim follows from the fact that any adversary playing a game with at most ℓ GARBLE_FUNC calls obtains only ℓ random shares of key g_2 which are not useful in reconstructing the key and hence in discovering f and x. For the second part of the claim, let us consider an arbitrary adversary who is allowed to make $\ell + 1$ GARBLE_FUNC queries in its game. This adversary gets $\ell + 1$ shares of g_2 and there is a certain positive probability not depending on k that these shares can be used to reconstruct g_2. As a consequence, the adversary has a chance to discover f and x. Using that information the adversary would be able to find the challenge bit b and win the game. This leads to non-negligible advantage of the adversary. This shows that $\mathcal{G}'$ cannot belong to security class $\mathcal{K}_{\ell+1}$. □

As a corollary we get that $\mathcal{R}_\ell \subsetneq \mathcal{R}_1$ for $\ell \geq 2$. Next we remove the constraint that the GARBLE_FUNC procedure can be called at most ℓ times, allowing the adversary playing the security game to call GARBLE_FUNC arbitrarily many times.

Definition: Let xxx $\in$ {prv, mao, obv} and yyy $\in$ {ind, sim}. We define a new class $\mathcal{C}_* = \mathtt{GS}(xxx.yyy.radap_*, \Phi)$ which is the class of secure garbling schemes when the adversary is allowed to call the GARBLE_FUNC procedure arbitrarily many times in the XxxYyyRadap game.

Theorem 4: Let us again denote $\mathcal{K}_\ell = \mathtt{GS}(\text{xxx.yyy.radap}_\ell, \Phi)$ for xxx $\in$ {prv, mao, obv} and yyy $\in$ {ind, sim}. Then we have the following inclusion:

$$\mathcal{R}_* \subsetneq \bigcap_{\ell=1}^{\infty} \mathcal{R}_\ell \qquad \text{or} \qquad \mathcal{R}_* = \bigcap_{\ell=1}^{\infty} \mathcal{R}_\ell = \emptyset$$

Proof: The proof follows the ideas used in the proof of Theorem 2 in [14]. □

Theorem 5: Let xxx $\in$ {prv, mao, obv} and yyy $\in$ {sim, ind}. For any reusability parameters $\ell > 1$ and $\ell' > 0$ the following non-inclusions hold:

$$\mathtt{GS}(\text{xxx.yyy.adap}_{\ell'}) \not\subseteq \mathtt{GS}(\text{xxx.yyy.radap}_\ell)$$
$$\mathtt{GS}(\text{xxx.yyy.radap}_{\ell'}) \not\subseteq \mathtt{GS}(\text{xxx.yyy.adap}_\ell),$$

given that classes $\mathtt{GS}(\text{xxx.yyy.adap}_\ell)$ and $\mathtt{GS}(\text{xxx.yyy.radap}_\ell)$ are non-empty.

Proof: Let us assume that $\mathtt{GS}(\text{xxx.yyy.adap}_{\ell'})$ and $\mathtt{GS}(\text{xxx.yyy.radap}_{\ell'})$ are non-empty. Consider first the non-inclusion $\mathtt{GS}(\text{xxx.yyy.adap}_{\ell'}) \not\subseteq \mathtt{GS}(\text{xxx.yyy.radap}_\ell)$. If $\mathtt{GS}(\text{xxx.yyy.adap}_{\ell'}) \bigcap \mathtt{GS}(\text{xxx.yyy.radap}_\ell) = \emptyset$ then the claim is obvious. Otherwise, let $\mathcal{G} = (\mathtt{KeyGen}, \mathtt{Ga}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev}) \in \mathtt{GS}(\text{xxx.yyy.adap}_{\ell'}) \bigcap \mathtt{GS}(\text{xxx.yyy.radap}_\ell)$, where both algorithms Ga and En thus must be non-deterministic. Using $\mathcal{G}$ we construct a garbling scheme $\mathcal{G}' = (\mathtt{KeyGen}, \mathtt{Ga}', \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev})$ which is xxx.yyy.adap$_{\ell'}$ secure but not xxx.yyy.radap$_\ell$ secure. The algorithms in $\mathcal{G}'$ are just like in $\mathcal{G}$ except of the garbling algorithm $\mathtt{Ga}'$, which is simply the deterministic version of Ga. Now, $\mathcal{G}'$ still is xxx.yyy.adap$_{\ell'}$ secure, because F is created only once with $\mathtt{Ga}'$ in the security game and hence the determinism of $\mathtt{Ga}'$ does not affect the game. This is not the case in game XxxYyyRadap$_\ell$, where the algorithm $\mathtt{Ga}'$ may be used up to $\ell > 1$ times. The deterministic $\mathtt{Ga}'$ always returns the same F for the same f. An adversary gains advantage of this in both indistinguishability- and simulation-based games. In indistinguishability-based games, the adversary calls GARBLE_FUNC with inputs (f, f) and (f, f') with $f \neq f'$. If GARBLE_FUNC returns the same F both times, then the adversary knows that this is the garbling of f, meaning that the challenge bit was $b = 0$. If the garblings are different, then the adversary finds out that $b = 1$. In the simulation-based games, the adversary queries GARBLE_FUNC with the same function f. If the garblings returned by the procedure are different, then the adversary knows for sure that $b = 0$. In the other case, adversary answers $b = 1$ even though there is a small possibility that this is not the correct answer (the simulator may have produced the same F).

Consider then the second non-inclusion $\mathtt{GS}(\text{xxx.yyy.radap}_{\ell'}) \not\subseteq \mathtt{GS}(\text{xxx.yyy.adap}_\ell)$. In this case let $\mathcal{G} = (\mathtt{KeyGen}, \mathtt{Ga}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev}) \in$

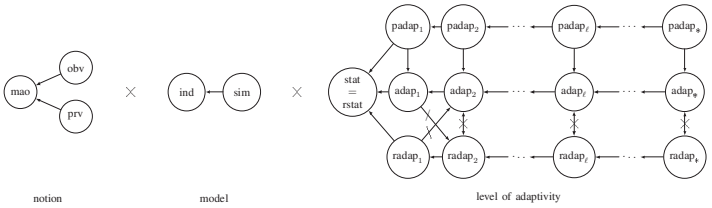


Figure 3. Hierarchy of garbling scheme classes represented as a Cartesian product of graphs. The lines $\nrightarrow$ show that there is no inclusion.

$\mathtt{GS}(\text{xxx.yyy.radap}_{\ell'}) \bigcap \mathtt{GS}(\text{xxx.yyy.adap}_{\ell})$ be a garbling scheme with non-deterministic algorithms $\mathtt{Ga}$ and $\mathtt{En}$. We construct a garbling scheme $\mathcal{G}' = (\mathtt{KeyGen}, \mathtt{Ga}, \mathtt{En}', \mathtt{De}, \mathtt{Ev}, \mathtt{ev})$ which is exactly as $\mathcal{G}$ except for the encryption algorithm $\mathtt{En}'$. Algorithm $\mathtt{En}'$ is the deterministic version of algorithm $\mathtt{En}$. It can be shown similarly as above that $\mathcal{G}'$ belongs to $\mathtt{GS}(\text{xxx.yyy.radap}_{\ell'})$ but not to $\mathtt{GS}(\text{xxx.yyy.radap}_{\ell})$. □

The next three theorems tell that the relation between the different security notions and models remain the same for reverse-order adaptive security as for static and adaptive security. These results tell, that simulation-based security always implies indistinguishability-based security. Moreover, privacy and obliviousness both imply matchability-only in both simulation- and indistinguishability-based models. Each theorem can be proved following the ideas used in proofs of corresponding theorems 4, 5 and 6 in [14].

Theorem 6: Let $\ell \in \mathbb{N}$ and xxx $\in \{\text{prv, mao, sim}\}$. Then $\mathtt{GS}(\text{xxx.sim.radap}_{\ell}, \Phi) \subseteq \mathtt{GS}(\text{xxx.ind.radap}_{\ell}, \Phi)$.

Theorem 7: The following inclusion holds for any $\ell \in \mathbb{N}$: $\mathtt{GS}(\text{prv.ind.radap}_{\ell}, \Phi) \bigcup \mathtt{GS}(\text{obv.ind.radap}_{\ell}, \Phi) \subseteq \mathtt{GS}(\text{mao.ind.radap}_{\ell}, \Phi)$.

Theorem 8: The following inclusion holds for any $\ell \in \mathbb{N}$: $\mathtt{GS}(\text{prv.sim.radap}_{\ell}, \Phi) \bigcup \mathtt{GS}(\text{obv.sim.radap}_{\ell}, \Phi) \subseteq \mathtt{GS}(\text{mao.sim.radap}_{\ell}, \Phi)$.

V. Conclusion

We have now completed describing the relations of classes of garbling schemes achieving reverse-order adaptive security. The central results of this paper are threefold. Firstly, we show that adaptivity and reverse-order adaptivity are quite different except of the static security: adaptive security does not imply reverse-order adaptivity and vice versa for $\ell \geq 2$. Secondly, we show that, similarly to adaptive security classes, the hierarchy of reverse-order adaptive security classes is is either infinite or collapses to the empty set.. Thirdly, for a fixed reusability parameter ℓ, the relations between the models and notions are exactly the same for adaptive and reverse-order adaptive classes (see fig. 3 in [14]).

To conclude, these three results together allow us to extend the hierarchy of [15]. We use the idea of *graph Cartesian product* for directed graphs in similar manner as in earlier papers [14], [15]: Using the graph Cartesian product, we present the hierarchy of security classes of garbling schemes as an product of security notion {prv, obv, mao}, security model {ind, sim} and the level of adaptivity {stat, adap, radap} with the subscript ℓ telling the level of reusability. The complete hierarchy is illustrated in fig. 3.

References

[1] Panda Security 2014, "Community - free access - cloud antivirus," http://www.cloudantivirus.com/en/forHome/, accessed April 29, 2014.

[2] D. Beaver, S. Micali, and P. Rogaway, "The round complexity of secure protocols," *Proc. of the* 22^{nd} *STOC*, pp. 503–513, 1990.

[3] M. Bellare, V. T. Hoang, and P. Rogaway, "Adaptively secure garbling scheme with applications to one-time programs and secure outsourcing," *Asiacrypt 2012*, vol. 7685 of LNCS, pp. 134–153, 2012.

[4] ——, "Foundations of garbled circuits," in *Proc. of ACM Computer and Communications Security (CCS'12)*. ACM, 2012, pp. 784–796.

[5] M. Bellare and P. Rogaway, "Code-based game-playing proofs and the security of triple encryption," *Advances in Cryptology, Proc. of Eurocrypt 2006*, vol. 4004 of LNCS, pp. 409–426, 2006.

[6] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," *Proc. of TCC 2011*, vol. 6597 of LNCS, pp. 253–273, 2011.

[7] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verfiable computing: Outsourcing computation to untrusted workers." *CRYPTO 2010*, vol. 6223 of LNCS, pp. 465–482, 2010.

[8] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, 2009, crypto.stanford.edu/craig.

[9] S. Goldwasser, Y. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich, "Reusable garbled circuits and succinct functional encryption," *Proc. of the* 45^{th} *STOC*, pp. 555–564, 2013.

[10] S. Goldwasser, Y. Kalai, and G. Rothblum, "One-time programs," *CRYPTO2008*, vol. 5157 of LNCS, pp. 39–56, 2008.

[11] S. Gorbunov, V. Vaikuntanathan, and H. Wee, "Attribute-based encryption for circuits," *Proc. of the* 45^{th} *STOC*, pp. 545–554, 2013.

[12] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. of ACM Computer and Communications Security (CCS'06)*. ACM, 2006, pp. 89–98.

[13] Y. Lindell and B. Pinkas, "A proof of security of Yao's protocol for secure two-party computation," *Journal of Cryptology*, vol. 22(2), pp. 161–188, 2009.

[14] T. Meskanen, V. Niemi, and N. Nieminen, "Hierarchy for classes of garbling schemes," *Proc. of Central European Conference on Cryptology (CECC'14)*, 2014.

[15] ——, "Hierarchy for classes of projective garbling schemes," *Proc. of International Conference on Information and Communications Technologies (ICT 2014)*, 2014.

[16] A. Yao, "Protocols for secure computations," *Proc. of* 23^{rd} *SFCS, 1982*, pp. 160–164, 1982.

[17] ——, "How to generate and exchange secrets," *Proc. of* 27^{th} *FOCS, 1986.*, pp. 162–167, 1986.

Paper VI

VI

Extended Model of Side-Information in Garbling

T. Meskanen and V. Niemi and N. Nieminen (2015). In *The 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-15)*, pages 950-957. IEEE

2015 IEEE Trustcom/BigDataSE/ISPA

Extended Model of Side-Information in Garbling

Tommi Meskanen
University of Turku,
Turku, FINLAND
tommes[at]utu.fi

Valtteri Niemi
University of Helsinki,
Helsinki, FINLAND
valtteri.niemi[at]helsinki.fi

University of Turku,
Turku, FINLAND

Noora Nieminen
University of Turku,
Turku, FINLAND
nmniem[at]utu.fi

Turku Centre for Computer Science (TUCS),
Turku, Finland

***Abstract*—Increasingly many applications utilize network-based solutions these days, such as cloud computing or Internet of Things technologies. Processing private data in various applications over the Internet raises concerns about the user privacy. These concerns may be solved by using novel cryptographic methods, of which garbling schemes is one. Side-information is a key concept for defining the security of garbling schemes since it tells what is allowed to be leaked about the garbled evaluation. Current definitions have a full support to logic circuits while the concept of a garbling scheme should encompass all garbling techniques independent of the model of computation. In this paper, we improve the definition of side-information to fit any computation model, especially Turing machines. Moreover, we show that our definition of side-information also describes better the various threats against the security of garbling schemes, including possible side-channel attacks. We also demonstrate that the new definition has also the following advantages compared to the existing definitions. Our model of side-information supports a wider set of applications, including partial garbling schemes. Our model simplifies the security definitions of garbling schemes without compromising the existing results about the security relations of garbling schemes.**

***Index Terms*—secure multiparty computation, garbling schemes, side-information function**

I. Introduction

Privacy is one of the greatest challenges for various network based applications. In many applications, private data is used for different computation purposes. In some of the cases, the client would not like the evaluator to learn this private data. During past decades, various cryptographic methods to protect the private data have been proposed. These methods include protocols for *secure multiparty computation*. One important aim of secure multiparty computation is to evaluate a function on an input without revealing privacy-sensitive information about the function or the input. One way to achieve this is to use *garbling techniques*. In the following section, we briefly introduce relevant literature.

A. Related research

The first garbling technique originates from Yao, who proposed a technique called *garbled circuits* [17]. Recently, Yao's technique has been formalized to transform the garbling technique from a cryptographic tool into a cryptographic primitive. To achieve this, Bellare et al. proposed a new concept of *a garbling scheme* in [3]. According to [3], the concept of a garbling scheme would encompass all garbling techniques independent of the way of representing the function f. In other words, the same framework would fit not only logic circuits but also Turing machines (TM), random-access machines (RAM), deterministic finite automatons (DFA) and so on. A generic definition of garbling scheme has been advantageous. In past few years, methods to garble Turing machines(garbled TM [7] and RAM's [10], [6]) have been introduced.

In brief, a garbling scheme works as follows (a more detailed description of garbling can be found in Section II). A party called *client* wants to evaluate a function f on an argument x but has decided to outsource the computation to another party called *evaluator*. The client garbles f and x and sends them to the evaluator. Then the evaluator computes the garbled function on the garbled argument without getting to know the original argument or the original function. The garbled result can be ungarbled by either party. The ungarbled result now represents the result of the original evaluation.

Due to the argument and function hiding property, garbling schemes have several possible applications, e.g. cloud computing among many others. A client, for example a health care provider, has privacy sensitive data which should be processed in a cloud. If the health care provider and the cloud agree to use a garbling scheme, then the privacy sensitive data, the processing algorithm and the final computation result can all be hidden from the third party cloud during the data processing.

Different applications set different requirements for the security of garbling schemes. The security of garbling schemes can be classified according to three properties. The first property is *security notion*, including *privacy*, *obliviousness* and *authenticity*. They also consider *the level of adaptivity* of the garbling scheme by defining three levels of adaptivity *static, coarse-grained* and *fine-grained* security [3], [4]. In addition, the security can be either *simulation-* or *indistinguishability-based*. All security concepts are combinations of these three properties.

978-1-4673-7952-6/15 $31.00 © 2015 IEEE
DOI 10.1109/Trustcom-BigDataSe-ISPA.2015.469

The security concepts have recently been generalized by Meskanen et al. in [11], [12], [13], [14], [15]. They also build a hierarchy for classes of garbling schemes showing the relations between the different security concepts. They have also introduced a new parameter, *reusability parameter*. This new parameter is inspired by the work of Goldwasser et al. [8] in which the first construction of *a reusable garbled circuit* was introduced. The reusability parameter models how many times it is safe to use the same garbled function for different garbled arguments. Not only garbled circuits enjoy the reusability property but also reusable garbling schemes for Turing machines have been constructed [7].

As a summary, garbling seems to have obtained its place as a cryptographic primitive - the notions for garbling schemes encompass different computation models as well as different security concepts. However, the definitions seem to be incomplete. While the definition of a garbling scheme is independent of the computation method, the concept defining the security of a garbling scheme seems to support some models of computation better than the others. This concept is called *side-information*.

Side-information function is a central tool in measuring the security of a garbling scheme. Side-information function models the information that is allowed to be leaked about f and x during the garbled evaluation. In other words, an adversary knowing the garbled function F and the garbled argument X is allowed to find out nothing beyond the information included in the side-information.

In the definition used by Bellare et al., the length of the argument, the length of the final value as well as the length of string presentation of the circuit are assumed to be included in the side-information. The garbled counterparts of the function f, the argument x and the value y reveal information about the lengths of f, x and y and therefore it is natural that $|f|$, $|x|$ and $|y|$ are easily found from the side-information. Bellare et al. also define that the side-information depends only on the function f. In the case of circuits, $|f|$, $|x|$ and $|y|$ are fixed when the circuit is fixed. However, determining $|f|$, $|x|$ and $|y|$ based on the description of the Turing machine is not anymore a natural requirement. For example, the length of the input and the length of the output cannot generally be determined by the Turing machine only. This suggests that the model of side-information does not support Turing machines like it supports circuits.

The concept of a side-information is not only important from the theoretical but also from the practical point-of-view. The garbling schemes `Garble1` and `Garble2` proposed and implemented by Bellare et al. (for the implementation, see [1]) achieve certain level of security which is parameterized with certain side-information function. For example, `Garble1` achieves indistinguishability-based privacy, if the topology of the circuit is allowed to be leaked. [2]

Another application in which the leaked information is central is a recently introduced concept of a concept of *partial garbling scheme* [9]. A partial garbling scheme is designed for a situation in which the argument x contains partly public and partly private information. According to [9], *public* means information that is allowed to be leaked about x during the garbled evaluation. It is hard to imagine that this feature would be reached using Bellare et al.'s model of side-information function that is based on information deducible from f only.

Our contributions

In this paper we extend the model of side-information function. The first extension is that the side-information depends on both f and x. The second extension is that our model of side-information depends also on the encryption key e and the decryption key d.

By doing this, in addition to logical circuits, also other computation models are supported in the security definitions of a garbling scheme. Moreover, our concept of side-information models better the overall security of garbling schemes by taking different types of attacks into account. Various attacks can reveal information about f, x and y. Following only the execution of encryption algorithm and capturing the garbled argument $X = \mathtt{En}(e, X)$ may reveal something about the argument x as well as possibly the encryption key e. If the adversary can follow the execution of garbling algorithm and capture the garbled function F then the adversary finds something about function f only. Thirdly, if the adversary possesses F and X and is able to follow the execution of decryption algorithm, then the adversary finds out something about the evaluation the final value $y = \mathtt{De}(d, \mathtt{Ev}(F, X))$ and possibly the decryption key d. Figure 1 illustrates the possible attacks of an adversary during the garbled evaluation process.

The new model of side-information also lets us simplify the different security concepts of garbling schemes. We provide the definitions of these new classes. First, we prove that our new model of security is compatible with the old model of security. We also show that the hierarchy of security classes will remain the same as in [13], [14].

This paper is organized as follows. In Section II we provide necessary definitions. Section III contains description of the new model of side-information and a more detailed discussion how the new model has been constructed. In Section IV we present the security classes using our new model of side-information. We also show in this section that our new model is compatible with the old model as well as that a similar hierarchy will be obtained in our new model as well.

II. Definitions

This section contains the basic definitions and notations used throughout this paper. A *string* is a infinite sequence of bits.The empty string is denoted by ε. Notation $y \leftarrow S$ means that an element is selected uniformly at random from the set S, and this element is assigned as a value to the variable y. An algorithm is denoted by A and notation

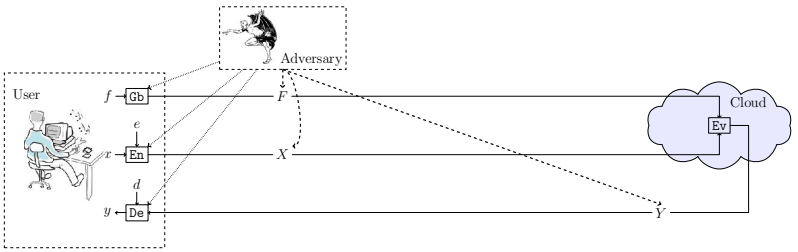


Figure 1. Diagram illustrating the possible attacks for an adversary. The dotted lines show possible targets for side-channel attacks. The dashed lines illustrate the attacks to directly retrieve information.

$A(x_1, \ldots, x_n)$ refers to the output of the algorithm A on inputs $x_1, \ldots, x_n$. We say that a function $f : \mathbb{N} \to \mathbb{R}$ is negligible if for every $c > 0$ there is an integer N_c such that $|f(n)| < n^{-c}$ for all $n > N_c$.

A. About computation models

In this paper, we focus on the fundamental differences in logic circuits and Turing machines that gives the motivation to improve the definition of side-information in garbling. Here, we omit the formal definitions of a logic circuit as well as a Turing machine and refer to literature for formal definition (e.g. [16]).

The definitions of a circuit and a Turing machine have a fundamental difference. If a function f is interpreted as a circuit, then the length of the argument x must be equal to the number of input wires. Moreover, the circuit fixes the length of the final value, since the length equals the number of output wires. In the description of a Turing machine, there are no corresponding restrictions to the lengths of the argument x or the lengths of final values y. However, for both circuits and Turing machines, the argument must be such that computing $\mathtt{ev}(f, x)$ is possible.

The second difference between circuits and Turing machines is related to the running time. Circuits can be evaluated in constant time for any input which is not the case for Turing machines. This observation about the running time of $\mathtt{ev}(f, x)$ now indicates, that the side-information function cannot only be dependent on the function f, it should also depend on the argument x.

B. Garbling scheme

The idea of a garbling scheme is to allow a function f to be evaluated privately on an argument x. Here, privately means that neither f nor x should be revealed to the evaluator. Sometimes it is even desired that the result of the evaluation $y = f(x)$ is kept secret. To achieve this both the function f and its argument x are garbled. The evaluator performs garbled evaluation using the garbled function F and the garbled argument X and gets a garbled value Y. This garbled value is then ungarbled either by the evaluator or the user. The garbled evaluation must yield the same final value as the original evaluation, meaning

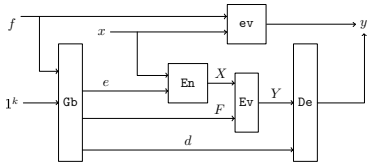


Figure 2. A garbling scheme. The diagram shows that the final value obtained via garbling must coincide the final value obtained by direct evaluation, i.e. $\mathtt{ev}(f, x) = y = \mathtt{De}(d, Y)$ where F is the garbled function, $Y = \mathtt{Ev}(F, X)$ the garbled value and $X = \mathtt{En}(e, x)$ the garbled argument.

that the result of the evaluation is independent of the method used for evaluation.

More formally, a garbling scheme is a 5-tuple of algorithms $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev})$. The last component of this tuple is the evaluation algorithm that computes the value of function f on argument x, i.e. $y = f(x)$. The first component of $\mathcal{G}$ is the garbling algorithm $\mathtt{Gb}$ which on input $(1^k, f)$ generates encryption key e, decryption key d and the garbled function F. Next, the garbled argument X is computed using algorithm $\mathtt{En}$ with input (e, x). Then, the garbled evaluation algorithm $\mathtt{Ev}$ is used for computing the garbled value $Y = \mathtt{Ev}(F, X)$. Finally, the decryption key d is used for ungarbling the garbled value Y, i.e. $y = \mathtt{De}(d, Y)$. Now, we must have that $\mathtt{ev}(f, x) = y = \mathtt{De}(d, Y) = \mathtt{De}(d, \mathtt{Ev}(F, X))$, meaning that the evaluation must yield the same result independent from the actual method of evaluation. Figure 2 illustrates how a garbling scheme works.

C. Tools to measure the security of a garbling scheme

The proofs in this paper are based on *code-based games* [5]. A game is a collection of three procedures: `INITIALIZE`, `FINALIZE` and other named procedures. We assume throughout this paper that procedures `INITIALIZE` and `FINALIZE` are compulsory in a game, whereas other procedures are optional.

The entity playing a game is called *adversary*. Every game starts with `INITIALIZE` procedure. Then the adversary may invoke other procedures before feeding its output to the `FINALIZE` procedure. Based on the input from the adversary `FINALIZE` creates a string that tells

the outcome from the game typically consisting of one bit of information: whether the adversary has won or not.

III. Generalizing the side-information function

Let us first briefly summarize the general properties of the side-information function in the model Bellare et al. use in [4]. The side-information $\Phi(f)$ is a measure of security of a garbling scheme since the side-information $\Phi(f)$ models the information leaked about the function f during the garbled evaluation. For example, $\Phi(f)$ may leak the topology of circuit f or even the entire circuit f. According to [4], the minimal requirement for the side-information $\Phi(f)$ is that the length of the argument $|x|$, the length of the final value $|y|$ and the length of the function $|f|$ can be efficiently computed from $\Phi(f)$.

For circuits, the length of the argument x and the length of the final value y are dependent on the circuit. However, the same does not hold for Turing machines. Therefore, the side-information cannot depend only on the function f, it should also depend on x.

A straight-forward way to include x to the side-information is the following. We define the side-information as a function $\Psi(f, x) = (\Phi(f), |x|, |y|, |f|)$, where $\Phi(f)$ encompasses the side-information leaked about f. This definition explicitly fulfills the requirement that $|x|$, $|y|$ and $|f|$ must be efficiently computable from the side-information. Moreover, the length of the argument x does not need to be deducible from f anymore. Using this model, varying the first component $\Phi(f)$ we can achieve exactly the same side-information models as presented in [4]: in the case Φ_{topo} we define, that the component $\Phi(f)$ in Ψ_{topo} leaks the topology of f (if f is modeled as a circuit) and in the case Φ_{circ} we define $\Phi(f) = f$.

Unfortunately, there are some major flaws in this model. The straight-forward model does not fully take into account the interplay of both f and x. At least information about y, the time and space complexity of computing $\mathtt{ev}(f, x)$ depend on both f and x, not from f only. This lets us conclude that the straight-forward model is too simplistic.

To summarize, the information leaked during the garbled evaluation can be threefold. The information may be related only to the argument x. This information contains at least the length of x. Secondly, the information may contain information related only to the function, which is at least the length of f but may for example contain information about the structure of the function (e.g. topology if f is a circuit or the number of transition functions if f is a Turing machine). Thirdly, there is information that depends on both f and x, for example information about the final value $y = f(x)$ or the running time. To model this, the side-information function $\Psi(f, x)$ should be a composition of three side-information functions, first of which is dependent on x, second of which is dependent of f and third of which is dependent on both f and x.

The componentwise side-information function also fits better the threat model presented in fig. 1. However, the side-information function that depends on (f, x) is not sufficient. Further analysis of the threat model indicates that an adversary may also target attacks towards the actual garbling processes, not only against the garbled evaluation. More specifically, the adversary may attack the garbling algorithm `Gb`, encryption algorithm `En` and decryption algorithm `De`. These three algorithms depend not only from (f, x) but also the keys (e, d).

Next, we give the formal definition of the side-information function $\Psi_{\text{tot}}(e, d, f, x)$ described above.

Definition 1: Side-information function Ψ_{tot} is a function which depends on four arguments: the encryption key e, the decryption key d, the function f and the argument x. The function Ψ_{tot} returns a string. The length of the argument x, the length of the final value y and the length of the function f must be efficiently computable from the string $\Psi_{\text{tot}}(e, d, f, x)$.

The total side-information Ψ_{tot} may consist of several smaller blocks of side-information. For example, the side-information $\Psi_{\text{tot}}(e, d, f, x)$ always includes a part that is dependent on the function f and the argument x, denoted by $\Psi_{\text{ev}}(f, x)$. Another examples of partial side-information functions are Ψ_{func}, $\Psi_{\mathtt{Gb}}$ and Ψ_{rest}. The side-information function Ψ_{func} depends only on the function f, and at most the length of f must be efficiently computable from it. The side-information function $\Psi_{\mathtt{Gb}}$ depends on (e, d, f) and $\Psi_{\text{rest}}(e, d, f, x)$ encompasses all the other information that is not included to any other partial side-information. The partial side-information function $\Psi_{\text{ev}}(f, x)$ and other partial side-information functions can be efficiently computed from $\Psi_{\text{tot}}(e, d, f, x)$.

IV. Security classes of garbling schemes

In this section, we define the security concepts for garbling schemes. Our definitions are fundamentally different compared with the definitions in [3], [4], [11], [12], [13], [14], [15]. Security notions notions *privacy*, *obliviousness* and *matchability-only* were having their own security games that differed only a little. Now, we can achieve the same notions through varying the side-information function which simplifies the definitions. Next, we explain how this can be realized.

A. Security definitions of garbling schemes

We start by providing the descriptions of the new security games using the new model of side-information introduced in the previous Section III. Then we explain how these games are used for measuring the security of a garbling scheme.

All security games SIM.STAT, IND.STAT, SIM.ADAP$_\ell$ and IND.ADAP$_\ell$ start with procedure `INITIALIZE` in which the challenge bit b is chosen uniformly at random. Then, games SIM.STAT and IND.STAT continue with a

proc INITIALIZE()
$b \leftarrow \{0,1\}$

proc GARBLE(f, x)
if $x \notin D_f$
  then return $\perp$
if $b = 1$ **then**
  $(F, e, d) \leftarrow \mathtt{Gb}(1^k, f)$
  $X \leftarrow \mathtt{En}(e, x)$
  $z \leftarrow \Psi_{\text{tot}}(e, d, f, x)$
else
  $(F, X, z) \leftarrow \mathcal{S}(1^k, \Psi_{\text{ev}}(f, x))$
return (F, X, z)

proc FINALIZE(b')
return $b = b'$

Table I
THE GAMES DEFINING THE SIMULATION-BASED STATIC SECURITY SIM.STAT. THE GAME DEPENDS ON THE CHOICE OF $\mathcal{G}, \Phi$ AND $\mathcal{S}$.

proc INITIALIZE()
$b \leftarrow \{0,1\}$

proc GARBLE(f_0, f_1, x_0, x_1)
if $x_0 \notin D_{f_0}$ **or** $x_1 \notin D_{f_1}$
  then return $\perp$
if $\Psi_{\text{ev}}(f_0, x_0) \neq \Psi_{\text{ev}}(f_1, x_1)$
  then return $\perp$
$(F, e, d) \leftarrow \mathtt{Gb}(1^k, f_b)$
$X \leftarrow \mathtt{En}(e, x_b)$
$z \leftarrow \Psi_{\text{tot}}(e, d, f_b, x_b)$
return (F, X, z)

proc FINALIZE(b')
return $b = b'$

Table II
THE GAMES DEFINING THE INDISTINGUISHABILITY-BASED STATIC SECURITY IND.STAT. THE GAME DEPENDS ON THE CHOICE OF $\mathcal{G}, \Phi$ AND $\mathcal{S}$.

proc INITIALIZE()
$b \leftarrow \{0,1\}$

proc GARBLE_FUNC(f)
if $b = 1$ **then**
  $(F, e, d) \leftarrow \mathtt{Gb}(1^k, f)$
  $w \leftarrow \Psi_{\mathtt{Gb}}(e, d, f)$
else
  $(F, w) \leftarrow \mathcal{S}(1^k, \Psi_{\text{func}}(f))$
return (F, w)

proc GARBLE_ARG(x)
if $x \notin D_f$
  then return $\perp$
if $b = 1$ **then**
  $X \leftarrow \mathtt{En}(e, x)$
  $z \leftarrow \Psi_{\text{tot}}(e, d, f, x)$
else
  $(X, z) \leftarrow \mathcal{S}(\Psi_{\text{ev}}(f, x))$
return (X, z)

proc FINALIZE(b')
return $b = b'$

Table III
THE GAMES DEFINING THE SIMULATION-BASED ADAPTIVE SECURITY SIM.ADAP$_\ell$. THE GAME DEPENDS ON THE CHOICE OF $\mathcal{G}, \Phi$ AND $\mathcal{S}$.

proc INITIALIZE()
$b \leftarrow \{0,1\}$

proc GARBLE_FUNC(f_0, f_1)
if $\Psi_{\text{func}}(f_0) \neq \Psi_{\text{func}}(f_1)$
  then return $\perp$
$(F, e, d) \leftarrow \mathtt{Gb}(1^k, f_b)$
$w \leftarrow \Psi_{\mathtt{Gb}}(e, d, f_b)$
return (F, w)

proc GARBLE_ARG(x_0, x_1)
if $x_0 \notin D_{f_0}$ **or** $x_1 \notin D_{f_1}$
  then return $\perp$
if $\Psi_{\text{ev}}(f_0, x_0) \neq \Psi_{\text{ev}}(f_1, x_1)$
  then return $\perp$
$X \leftarrow \mathtt{En}(e, x_b)$
$z \leftarrow \Psi_{\text{tot}}(e, d, f_b, x_b)$
return (X, z)

proc FINALIZE(b')
return $b = b'$

Table IV
THE GAMES DEFINING THE INDISTINGUISHABILITY-BASED SECURITY IND.ADAP$_\ell$. THE GAME DEPENDS ON THE CHOICE OF $\mathcal{G}, \Phi$ AND $\mathcal{S}$.

query to procedure GARBLE. In games SIM.ADAP$_\ell$ and IND.ADAP$_\ell$, the game continues first with a query to procedure GARBLE_FUNC, then with possibly multiple queries to procedure GARBLE_ARG. All four games end with a query to procedure FINALIZE.

Next we describe the garbling procedures in games SIM.STAT, IND.STAT, SIM.ADAP$_\ell$ and IND.ADAP$_\ell$ in more details. Let us start with the static security games.

In simulation-based security game (see table I), GARBLE takes a function f and an argument x as input. First, the procedure checks whether the argument x can be evaluated with function f by testing whether x belong to the domain D_f of f. If $x \notin D_f$ then the procedure returns $\perp$ and the procedure is ended with no information about the garbled function F or garbled argument X. If $x \in D_f$ then the procedure generates the garbled function F, the garbled argument X and the side-information z based on the challenge bit, randomly chosen by the game. If $b = 1$ is 1, then (F, X, z) is generated by the original garbling algorithms Gb, En and side-information function Ψ_{tot}. Otherwise, a simulator $\mathcal{S}$ generates (F, X, z) based on the security parameter 1^k and the side-information $\Psi_{\text{ev}}(f, x)$.

In the indistinguishability-based security game (see table II), the procedure GARBLE takes two functions f_0, f_1 and two arguments x_0, x_1 as input. First, the procedure checks whether x_0 belongs to the domain of f_0 and x_1 belongs to the domain of f_1. If $x_0 \notin D_{f_0}$ or $x_1 \notin D_{f_1}$ then the procedure is ended with return value $\perp$. Otherwise, the procedure generates (F_b, X_b, z) based on the choice of the challenge bit. (F_b, X_b) is generated with garbling algorithms Gb and En from (f_b, x_b) where $b \in \{0,1\}$. The value z represents the side-information Ψ_{tot} related to (f_b, x_b, e, d).

Consider next the adaptive security games SIM.ADAP$_\ell$ and IND.ADAP$_\ell$. The parameter ℓ tells the level of reusability: the adversary $\mathcal{A}$ is allowed to make at most ℓ queries to procedure GARBLE_ARG. Note that ℓ can also be infinite.

The GARBLE_FUNC procedure takes f as an input in simulation-based security model the procedure (see table III), Then, depending on the choice of the challenge bit, GARBLE_FUNC prepares F that is either generated by the real garbling algorithm Gb (in the case $b = 1$) or F is generated by a simulator $\mathcal{S}$ (case $b = 0$). The simulator generates F based on the security parameter 1^k and the side-information that is leaked about the function f, i.e. $\Psi_{\text{func}}(f)$. The procedure GARBLE_FUNC returns F to the adversary. The procedure GARBLE_ARG behaves in similar manner: if $b = 1$ then the procedure GARBLE_ARG uses the real encryption algorithm En to generate X. If $b = 0$ then the simulator $\mathcal{S}$ generates X based on side-information $\Psi_{\text{ev}}(f, x)$.

In indistinguishability-based model (see table IV), procedure GARBLE_FUNC takes two functions f_0 and f_1 as an input. Depending on the challenge bit, the procedure garbles either f_0 (in the case $b = 0$) or f_1 ($b = 1$). The procedure GARBLE_FUNC returns F to the adversary. Procedure GARBLE_ARG behaves in similar manner: if the challenge bit has value $b = 1$ then the argument x_1 is encrypted, whereas in the case $b = 0$ the argument x_0 is encrypted.

In all four games (SIM.STAT, IND.STAT, SIM.ADAP$_\ell$, IND.ADAP$_\ell$), the task of the adversary $\mathcal{A}$ is to find out the correct value of the challenge bit b. The probability of answering the correct value is denoted by $\Pr[\mathcal{A} \text{ wins}]$. The advantage of an adversary is defined as $\mathbf{Adv}_{\mathcal{A}} = 2 \cdot \Pr[\mathcal{A} \text{ wins}] - 1$.

A garbling scheme $\mathcal{G}$ is SIM.STAT secure over Ψ_{tot}, if for all polynomial-time adversaries $\mathcal{A}$ there exist a polynomial-time simulator $\mathcal{S}$ such that the advantage $\mathbf{Adv}_{\mathcal{A}}$ is negligible in SIM.STAT game. Garbling scheme $\mathcal{G}$ is SIM.ADAP$_\ell$ secure over Ψ_{tot} if for all polynomial-time adversaries $\mathcal{A}$ there exist a polynomial-time simulator $\mathcal{S}$ such that the advantage $\mathbf{Adv}_{\mathcal{A}}$ is negligible in SIM.ADAP$_\ell$ game.

A garbling scheme $\mathcal{G}$ is IND.STAT secure over Ψ_{tot}, if for all polynomial-time adversaries $\mathcal{A}$ the advantage $\mathbf{Adv}_{\mathcal{A}}$ is negligible in IND.STAT game. Garbling scheme $\mathcal{G}$ is IND.ADAP$_\ell$ secure over Ψ_{tot} if for all polynomial-time adversaries $\mathcal{A}$ the advantage $\mathbf{Adv}_{\mathcal{A}}$ is negligible in SIM.ADAP$_\ell$ game.

Our security definitions consist of two components: the security model (simulation- or indistinguishability-based) and the type of adaptivity (static, adaptive). Garbling schemes achieving adaptive security can also be reusable. The level of reusability is measured with reusability parameter ℓ. Reusability parameter ℓ tells how many times the same garbled function can be securely used for evaluating different arguments. In the security game this means that the procedure `GARBLE` is queried at most ℓ times

Notations. We use notation XXX.YYY where $XXX \in$ {SIM, IND}, $YYY \in$ {STAT,ADAP$_\ell$} for a XXX.YYY secure garbling scheme. We use notation $\texttt{GS}(XXX.YYY, \Psi_{\text{tot}})$ to denote the corresponding security class that is a set of all garbling schemes achieving XXX.YYY security over Ψ_{tot}.

B. Compatibility of the new model to the established model of side-information function

In this section we show that the security definitions presented in the previous section are compatible with the definitions of [3], [4], [11], [13] when the function f is represented as a circuit and x is a bit string. In other words, we consider the specific circuit evaluation algorithm $\texttt{ev}_{circ}$. The security classes defined in [3], [4], [11], [13] are denoted by $\texttt{GS}(\text{xxx.yyy.zzz}, \Phi)$ where the first component represents the security notion xxx $\in$ {prv, obv, mao}, the second component represents the security model yyy $\in$ {sim, ind}, the third component represents the level of adaptivity zzz $\in$ {stat,adap$_\ell$} and Φ represents the side-information function depending only on function f. The security games defining security classes $\texttt{GS}(\text{xxx.yyy.zzz}, \Phi)$ can be found in [3], [11]. The class of garbling schemes using the circuit evaluation algorithm $\texttt{ev}_{circ}$ is denoted by $\texttt{GS}(\texttt{ev}_{circ})$.

In the three following theorems we assume that the side-information $\Psi_{\text{tot}}(e, d, f, x)$ consists of the following partial side-information components: $\Psi_{\texttt{ev}}(f, x)$, $\Psi_{\texttt{Gb}}(e, d, f)$, $\Psi_{\text{func}}(f)$ and $\Psi_{\text{rest}}(e, d, f, x)$.

Theorem 1: Let $xxx \in$ {sim, ind} and $XXX \in$ {SIM, IND} correspondingly. Let $yyy \in$ {stat, adap$_\ell$} and $YYY \in$ {STAT, ADAP$_\ell$} correspondingly. If $\Psi_{\texttt{ev}}(f, x) = (\texttt{ev}_{circ}(f, x), \Phi(f))$, $\Psi_{\texttt{Gb}}(e, d, f) = \Psi_{\text{func}}(f) = \Phi(f)$ and $\Psi_{\text{rest}}(e, d, f, x) = d$ then the following equality holds.

$$\texttt{GS}(prv.xxx.yyy, \Phi) \bigcap \texttt{GS}(\texttt{ev}_{circ}) = \texttt{GS}(\text{XXX.YYY}, \Psi_{\text{tot}}) \bigcap \texttt{GS}(\texttt{ev}_{circ}).$$

Proof: Omitted due to lack of space. □

Theorem 2: Let $xxx \in$ {sim, ind} and $XXX \in$ {SIM, IND} correspondingly. Let $yyy \in$ {stat, adap$_\ell$} and $YYY \in$ {STAT, ADAP$_\ell$} correspondingly. If $\Psi_{\text{rest}}(e, d, f, x) = \varepsilon$ and $\Psi_{\texttt{ev}}(f, x) = \Psi_{\text{func}}(f) = \Psi_{\texttt{Gb}}(e, d, f) = \Phi(f)$ then the following equality holds.

$$\texttt{GS}(obv.xxx.yyy, \Phi) \bigcap \texttt{GS}(\texttt{ev}_{circ}) = \texttt{GS}(\text{XXX.YYY}, \Psi_{\text{tot}}) \bigcap \texttt{GS}(\texttt{ev}_{circ})$$

Proof: Omitted due to lack of space. □

Theorem 3: Let $xxx \in$ {sim, ind} and $XXX \in$ {SIM, IND} correspondingly. Let $yyy \in$ {stat, adap$_\ell$} and $YYY \in$ {STAT, ADAP$_\ell$} correspondingly. If $\Psi_{\texttt{ev}}(f, x) = (\texttt{ev}_{circ}(f, x), \Phi(f))$, $\Psi_{\text{rest}}(e, d, f, x) = \varepsilon$ and $\Psi_{\text{func}}(f) = \Psi_{\texttt{Gb}}(e, d, f) = \Phi(f)$ then the following equality holds.

$$\texttt{GS}(mao.xxx.yyy, \Phi) \bigcap \texttt{GS}(\texttt{ev}_{circ}) = \texttt{GS}(\text{XXX.YYY}, \Psi_{\text{tot}}) \bigcap \texttt{GS}(\texttt{ev}_{circ})$$

Proof: Omitted due to lack of space. □

C. Relations between the security classes under the new model

Bellare et al. as well as Meskanen et al. have proven several relations between the security classes of garbling schemes in [3], [4], [11], [12], [13], [14], [15]. In this section we prove that similar relations hold also in our new model.

Theorem 4: We have the following inclusions

(a) $\texttt{GS}(\text{SIM.ADAP}_1, \Psi_{\text{tot}}) \subseteq \texttt{GS}(\text{SIM.STAT}, \Psi_{\text{tot}})$
(b) $\texttt{GS}(\text{IND.ADAP}_1, \Psi_{\text{tot}}) \subseteq \texttt{GS}(\text{IND.STAT}, \Psi_{\text{tot}})$

Proof: We prove the claim for case (a). The case (b) can be proven using similar ideas. Thus, let a garbling scheme $\mathcal{G}$ be SIM.ADAP$_1$ secure over side-information function Ψ_{tot}. Our aim is to prove that $\mathcal{G}$ achieves also simulation-based static security over Ψ_{tot}. Let adversary $\mathcal{A}$ be an arbitrary adversary playing the SIM.STAT game against $\mathcal{G}$. We construct an adversary $\mathcal{B}$ playing the SIM.ADAP$_1$ game that emulates game SIM.STAT for adversary $\mathcal{A}$. In other words, adversary $\mathcal{B}$ uses $\mathcal{A}$ as a subroutine as follows. The game SIM.ADAP$_1$ starts with the choice of the challenge bit b. After that, adversary $\mathcal{B}$ is asked to make its query to procedure `GARBLE_FUNC`. Instead of choosing the function f itself, adversary $\mathcal{B}$ asks adversary $\mathcal{A}$ to provide input for procedure `GARBLE` in

SIM.STAT game. Adversary $\mathcal{A}$ presumes to play an actual SIM.STAT game, so $\mathcal{A}$ sends (f, x) to $\mathcal{B}$ who emulates the GARBLE procedure. Adversary $\mathcal{B}$ checks first, whether argument x belongs to the domain of function f. If $x \notin D_f$, then $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$ and procedure GARBLE is ended with no output value for $\mathcal{A}$. Otherwise adversary $\mathcal{B}$ sends f to GARBLE_FUNC procedure which returns (F, w) to $\mathcal{B}$. Then adversary $\mathcal{B}$ calls GARBLE_ARG with input x. Now, if $x \notin D_f$ then procedure GARBLE_ARG is ended with no output to $\mathcal{B}$. Otherwise, GARBLE_ARG returns (X, z) to $\mathcal{B}$. $\mathcal{B}$ combines the outputs of GARBLE_FUNC and GARBLE_ARG into (F, X, z) and sends this to $\mathcal{A}$. Adversary $\mathcal{A}$ sends $b_{\mathcal{A}}$ to $\mathcal{B}$ who in turn uses $\mathcal{A}$'s answer as its own answer $b_{\mathcal{B}}$.

In game SIM.ADAP$_1$ adversary $\mathcal{B}$ learns (F, x, z) where $z = \Psi_{\text{tot}}(e, d, f, x)$ if $b = 1$ and if $b = 0$ then z is created by simulator $\mathcal{S}$ based on $\Psi(f, x)$. In game SIM.STAT adversary $\mathcal{A}$ is allowed to find out (F, X, z) where $z = \Psi_{\text{tot}}(e, d, f_b, x_b)$. This means that the information that adversary $\mathcal{B}$ gets in game SIM.ADAP$_1$ is exactly what is needed in SIM.STAT game. □

Theorem 5: Let $\mathcal{C}_\ell$ be a class of adaptively secure garbling schemes, i.e. $\mathcal{C}_\ell = \mathtt{GS}(\text{SIM.ADAP}_\ell, \Psi_{\text{tot}})$ or $\mathcal{C}_\ell = \mathtt{GS}(\text{IND.ADAP}_\ell, \Psi_{\text{tot}})$. Then $\mathcal{C}_{\ell+1} \subsetneq \mathcal{C}_\ell$ or $\mathcal{C}_{\ell+1} = \mathcal{C}_\ell = \emptyset$.

Proof: The inclusion $\mathcal{C}_{\ell+1} \subseteq \mathcal{C}_\ell$ is fairly obvious, because the adversary playing the game related to class $\mathcal{C}_{\ell+1}$ can always use only ℓ INPUT calls instead of the maximum allowed number of calls.

Then we need to prove that $\mathcal{C}_{\ell+1} \subsetneq \mathcal{C}_\ell$ when $\mathcal{C}_\ell \neq \emptyset$. Let $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev})$ be a garbling scheme in $\mathcal{C}_\ell$. We construct another garbling scheme $\mathcal{G}' = (\mathtt{Gb}', \mathtt{En}', \mathtt{De}, \mathtt{Ev}', \mathtt{ev})$ by modifying three components of $\mathcal{G}$, namely $\mathtt{Gb}, \mathtt{En}$ and $\mathtt{Ev}$, and show that $\mathcal{G} \notin \mathcal{C}_{\ell+1}$ while $G \in \mathcal{C}_\ell$.

First we describe how the encrypting algorithm $\mathtt{En}$ is modified. The new encryption algorithm $\mathtt{En}'$ consists of three parts. The first part is the original encrypting algorithm $\mathtt{En}$. The second part is a symmetric nondeterministic secure encryption algorithm $\overline{\mathtt{En}}$ that is completely independent from $\mathtt{En}$. Algorithm $\overline{\mathtt{En}}$ is used for encrypting x with an independent key $\overline{e}$. Finally, the third part represents a secret sharing scheme that creates t shares of key $\overline{e}$ in such a way that $\ell+1$ shares are needed to reconstruct the key $\overline{e}$. Now, garbling argument x with algorithm $\mathtt{En}'$ is performed as follows: $X' = \mathtt{En}'(e', x) = (X, \overline{X}, s)$ where $e' = (e, \overline{e}, t \text{ shares of key } \overline{e}), X = \mathtt{En}(e, x)$, $\overline{X} = \overline{\mathtt{En}}(\overline{e}, x)$ and s is a random share of $\overline{e}$.

The modified garbling algorithm $\mathtt{Gb}'$ creates (F, e, d) in the similar manner as $\mathtt{Gb}$ would if scheme $\mathcal{G}$ had been used. In addition, algorithm $\mathtt{Gb}$ creates $\overline{e}$ based on the security parameter 1^k. Algorithm $\mathtt{Gb}'$ also creates t shares of $\overline{e}$. The output of algorithm $\mathtt{Gb}'$ is $(F', e', d) = ((F, \overline{F}), (e, \overline{e}, \text{t shares of key } \overline{e}), d)$ where F is the garbled function and $\overline{F} = \overline{\mathtt{En}}(\overline{e}, f)$.

The garbled evaluation algorithm $\mathtt{Ev}'$ takes $(F', X') = ((F, \overline{F}), (X, \overline{X}, s_i))$ as its inputs. The algorithm omits the new parts $\overline{F}, \overline{X}, s_i$ and computes $Y = \mathtt{Ev}'((F, \overline{F}), (X, \overline{X}, s)) = \mathtt{Ev}(F, X)$.

Our aim is now to prove that the constructed garbling scheme $\mathcal{G}'$ belongs to class $\mathcal{C}_\ell$ but not to class $\mathcal{C}_{\ell+1}$. The first claim follows from the fact that any adversary playing a game with at most ℓ INPUT calls obtains only ℓ random shares of key $\overline{e}$ which are not useful in reconstructing the key and hence in discovering x and f. For the second part of the claim, let us consider an arbitrary adversary who is allowed to make $\ell + 1$ INPUT queries in its game. This adversary gets $\ell + 1$ shares of $\overline{e}$ and there is a certain positive probability not depending on k that these shares can be used to reconstruct $\overline{e}$. As a consequence, the adversary has a chance to discover f and x. Using that information the adversary would be able to find the challenge bit b and win the game. This leads to non-negligible advantage of the adversary. This shows that $\mathcal{G}'$ cannot belong to security class $\mathcal{C}_{\ell+1}$. □

Theorem 6: Let $\ell \in \mathbb{N}$. Then the following inclusions hold

(a) $\mathtt{GS}(\text{SIM.STAT}, \Psi_{\text{tot}}) \subseteq \mathtt{GS}(\text{IND.STAT}, \Psi_{\text{tot}})$

(b) $\mathtt{GS}(\text{SIM.ADAP}_\ell, \Psi_{\text{tot}}) \subseteq \mathtt{GS}(\text{IND.ADAP}_\ell, \Psi_{\text{tot}})$

Proof: Consider first the case (a). Let us assume that $\mathcal{G}$ is SIM.STAT secure over side-information function Ψ_{tot}. Let $\mathcal{A}$ be an arbitrary adversary playing the IND.STAT security game. We construct an adversary $\mathcal{B}$ for game SIM.STAT as follows. Adversary $\mathcal{B}$ uses $\mathcal{A}$ as a subroutine in the following way.

First, the challenge bit b in the game SIM.STAT is chosen uniformly at random. Next, adversary $\mathcal{B}$ is asked to give input for procedure GARBLE. Instead of choosing (f, x) itself, $\mathcal{B}$ tells $\mathcal{A}$ that the game IND.STAT has started and $\mathcal{A}$ should provide its input (f_0, f_1, x_0, x_1) to GARBLE procedure in game IND.STAT. Adversary $\mathcal{A}$ presumes to play an actual IND.STAT game, so $\mathcal{A}$ sends (f_0, f_1, x_0, x_1) to adversary $\mathcal{B}$. Now, $\mathcal{B}$ chooses the $c \leftarrow \{0, 1\}$ representing the challenge bit in the game IND.STAT. Adversary $\mathcal{B}$ first checks that the arguments x_0, x_1 are suitable for functions f_0 and f_1. If x_0 does not belong to the domain of function f_0 or x_1 does not belong to the domain of function f_1 then $\mathcal{B}$ returns $\perp$ to adversary $\mathcal{A}$ and the GARBLE procedure in game IND.STAT is finished. Otherwise, adversary $\mathcal{B}$ checks whether the side-information $\Psi_{\text{ev}}(f_0, x_0) = \Psi_{\text{ev}}(f_1, x_1)$. If $\Psi_{\text{ev}}(f_0, x_0) \neq \Psi_{\text{ev}}(f_1, x_1)$ then $\mathcal{B}$ returns $\perp$ to adversary $\mathcal{A}$ and again GARBLE in IND.STAT game is ended with no output to $\mathcal{A}$.

In spite of the test results related to the input (f_0, f_1, x_0, x_1), adversary $\mathcal{B}$ sends (f_c, x_c) to GARBLE procedure (in game SIM.STAT). If x_c does not belong to the domain of function f_c then adversary $\mathcal{B}$ receives $\perp$. Otherwise, adversary $\mathcal{B}$ gets (F, X, z) from GARBLE procedure. If $x_0 \in D_{f_0}$, $x_1 \in D_{f_1}$ and $\Psi_{\text{ev}}(f_0, x_0) = \Psi_{\text{ev}}(f_1, x_1)$ then $\mathcal{B}$ sends (F, X, z) to adversary $\mathcal{A}$. Then, $\mathcal{A}$ returns $b_{\mathcal{A}}$, the answer to the challenge, to adversary $\mathcal{B}$. Adversary $\mathcal{A}$ wins the game if $b_{\mathcal{A}} = c$. Adversary $\mathcal{B}$ answers $b_{\mathcal{B}} = 1$ in its game if and only if $x_{\in} D_{f_0}$, $x_1 \in D_{f_1}$, $\Psi_{\text{ev}}(f_0, x_0) = \Psi_{\text{ev}}(f_1, x_1)$ and $b_{\mathcal{A}} = c$. Otherwise $\mathcal{B}$ answers $b_{\mathcal{B}} = 0$.

Let us now analyse the possible outcomes of the games. If $x_0 \notin D_{f_0}$ or $x_1 \notin D_{f_1}$ or $\Psi_{\text{ev}}(f_0, x_0) \neq \Psi_{\text{ev}}(f_1, x_1)$ then adversary $\mathcal{B}$ answers 0 regardless of $\mathcal{A}$'s answer. The answer $b_{\mathcal{B}} = 0$ is correct with probability $\frac{1}{2}$ in this case. If the above tests for (f_0, f_1, x_0, x_1) are passed then there are two different outcomes. First, if $b = 0$ then (F, X, z) is generated by simulator $\mathcal{S}$. In this case, $\mathcal{A}$'s answer is not better than a guess. $\mathcal{A}$ guesses correctly with probability $\frac{1}{2}$. Moreover, adversary $\mathcal{A}$ wins the game if and only if adversary $\mathcal{B}$ loses its game. Namely, if $b_{\mathcal{A}} = c$ then $\mathcal{A}$ wins the game, but $\mathcal{B}$ loses since $b_{\mathcal{B}} = 1 \neq 0 = b$. On the other hand, if $\mathcal{A}$ loses then $b_{\mathcal{A}} \neq c$. Then $\mathcal{B}$ answers 0 which is the correct answer in game SIM.STAT. Next, consider the case $b = 1$. Then (F, X, z) is created by the actual garbling algorithms based on either (f_0, x_0) or (f_1, x_1). Now, since the input (f_0, f_1, x_0, x_1) passes the tests, the win probability of adversary $\mathcal{A}$ in IND.STAT game is $\frac{1}{2} + \frac{1}{2}\mathbf{Adv}_{\mathcal{A}}$. This is also the win probability of adversary $\mathcal{B}$, because $\mathcal{B}$ wins the game whenever $b_{\mathcal{A}} = c$.

The above analysis lets us now determine the overall win probability of adversary $\mathcal{B}$:

$$\begin{aligned}\Pr[\mathcal{B} \text{ wins }] &= \frac{1}{2}\Pr[\mathcal{B} \text{ wins} | b = 0] + \frac{1}{2}\Pr[\mathcal{B} \text{ wins} | b = 1] \\ &= \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \left(\frac{1}{2} + \frac{1}{2}\mathbf{Adv}_{\mathcal{A}}\right) \\ &= \frac{1}{2} + \frac{1}{4}\mathbf{Adv}_{\mathcal{A}}\end{aligned}$$

On the other hand, we have that the win probability of adversary $\mathcal{B}$ satisfies equation $\Pr[\mathcal{B} \text{ wins }] = \frac{1}{2} + \frac{1}{2}\mathbf{Adv}_{\mathcal{B}}$. This yields that $\mathbf{Adv}_{\mathcal{A}} = 2 \cdot \mathbf{Adv}_{\mathcal{B}}$. We assumed that the garbling scheme $\mathcal{G}$ is SIM.STAT secure over Ψ_{ev}. Thus there is a simulator $\mathcal{S}$ such that the advantage of $\mathcal{B}$ is negligible. But then also the advantage of adversary $\mathcal{A}$ is negligible, which proves the claim in case (a).

The case (b) is proven in similar manner as above. □

V. Conclusion

The concept of side-information is central for garbling schemes since it parameterizes the security notations of garbling schemes. The current model of side-information is defined in such way that it only depends on the function f that needs to be garbled. For functions represented as logical circuits, this is not a restriction. However, for Turing machines this causes an issue. Since the goal is a general concept of garbling as a cryptographic primitive, the computation model should not restrict the applicability of garbling schemes.

To overcome the issue mentioned above, we have extended the model of side-information by letting it to depend on f, x, encryption key e and decryption key d instead of only function f. In this way, we are also able to cover a wider variety of attacks against a garbling scheme (cf. fig. 1).

Using the extended model of side-information, we have obtained the following results. We have simplified the definitions of different security notions by choosing different side-information function for each notion. Our new security definitions are compatible with the old definitions. Moreover, we have proved that the previously known relations between the security classes hold also in the new model.

Acknowledgments

This work was supported by the Finnish Academy project "Cloud Security Services" which is greatly appreciated.

References

[1] JustGarble. http://cseweb.ucsd.edu/groups/justgarble/. Accessed: 2014-10-13.

[2] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway. Efficient garbling from a fixed-key blockcipher. In *Proc. of Symposium on Security and Privacy 2013*, pages 478–492. IEEE, 2013.

[3] M. Bellare, V. T. Hoang, and P. Rogaway. Adaptively secure garbling scheme with applications to one-time programs and secure outsourcing. In *Proc. of Asiacrypt 2012*, volume 7685 of LNCS, pages 134–153. Springer, 2012.

[4] M. Bellare, V. T. Hoang, and P. Rogaway. Foundations of Garbled Circuits. In *Proc. of ACM Computer and Communications Security (CCS'12)*, pages 784–796. ACM, 2012.

[5] M. Bellare and P. Rogaway. Code-Based Game-Playing Proofs and the Security of Triple Encryption. *Advances in Cryptology, Proc. of Eurocrypt 2006*, 4004 of LNCS:409–426, 2006.

[6] C. Gentry, S. Halevi, S. Lu, R. Ostrovsky, M. Raykova, and D. Wichs. Garbled RAM Revisited. In *Proc. of* 33^{rd} *Eurocrypt*, volume 8441 of LNCS, pages 405–422, 2014.

[7] S. Goldwasser, Y. Kalai, R. Popa, V. Vaikuntanathan, and N. Zeldovich. How to Run Turing Machines on Encrypted Data. In *Proc. of* 33^{rd} *CRYPTO*, volume 8043 of LNCS, pages 536–553, 2013.

[8] S. Goldwasser, Y. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable Garbled Circuits and Succinct Functional Encryption. In *Proc. of the* 45^{th} *STOC*, pages 555–564. ACM, 2013.

[9] Y. Ishai and H. Wee. Partial Garbling Schemes and Their Applications. In J. Esparza, P. Fraigniaud, T. Husfeldt, and E. Koutsoupias, editors, *Automata, Languages, and Programming*, volume 8572 of *Lecture Notes in Computer Science*, pages 650–662. Springer Berlin Heidelberg, 2014.

[10] S. Lu and R. Ostrovsky. How to Garble RAM Programs. In *Proc. of* 32^{nd} *Eurocrypt*, volume 7881 of LNCS, pages 719–734, 2013.

[11] T. Meskanen, V. Niemi, and N. Nieminen. Classes of Garbled Schemes. *Infocommunications Journal*, V(3):8–16, 2013.

[12] T. Meskanen, V. Niemi, and N. Nieminen. Garbling in Reverse Order. In *The 13th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-14)*, 2014.

[13] T. Meskanen, V. Niemi, and N. Nieminen. Hierarchy for Classes of Garbling Schemes. In *Proc. of Central European Conference on Cryptology (CECC'14)*, 2014.

[14] T. Meskanen, V. Niemi, and N. Nieminen. Hierarchy for Classes of Projective Garbling Schemes. In *Proc. of International Conference on Information and Communications Technologies (ICT 2014)*, 2014.

[15] T. Meskanen, V. Niemi, and N. Nieminen. On Reusable Projective Garbling Schemes. In *2014 IEEE International Conference on Computer and Information Technology (CIT 2014)*, 2014.

[16] J. E. Savage. Models of Computation: Exploring the Power of Computing, 1997.

[17] A. Yao. How to generate and exchange secrets. In *Proc. of* 27^{th} *FOCS, 1986.*, pages 162–167. IEEE, 1986.

Paper VII

How to Use Garbling for Privacy Preserving Electronic Surveillance Services

VII

T. Meskanen and V. Niemi and N. Nieminen (2015). *Cyber Security and Mobility*, 4(1):41-64

How to Use Garbling for Privacy Preserving Electronic Surveillance Services

Tommi Meskanen[1], Valtteri Niemi[1] and Noora Nieminen[1,2]

[1]*Department of Mathematics and Statistics, University of Turku, 20014 Turun yliopisto, FINLAND*
[2]*Turku Centre for Computer Science (TUCS), FINLAND*
Corresponding Authors: {tommes; pevani; nmniem}@utu.fi

Received 15 September 2014; Accepted 17 April 2015;
Publication 22 May 2015

Abstract

Various applications following the Internet of Things (IoT) paradigm have become a part of our everyday lives. Therefore, designing mechanisms for security, trust and privacy for this context is important. As one example, applications related to electronic surveillance and monitoring have serious issues related to privacy. Research is needed on how to design privacy preserving surveillance system consisting of networked devices. One way to implement privacy preserving electronic surveillance is to use tools for multiparty computations. In this paper, we present an innovative way of using garbling, a powerful cryptographic primitive for secure multiparty computation, to achieve privacy preserving electronic surveillance. We illustrate the power of garbling in a context of a typical surveillance scenario. We discuss the different security measures related to garbling as well as efficiency of garbling schemes. Furthermore, we suggest further scenarios in which garbling can be used to achieve privacy preservation.

Keywords: Internet of Things, privacy, electronic surveillance, garbling schemes.

Journal of Cyber Security, Vol. 4, 41–64.
doi: 10.13052/jcsm2245-1439.413

1 Introduction

Nowadays, we are surrounded by an increasing variety of *things* or *objects* that are connected with each other and accessible through the Internet. This trend is a consequence of a novel paradigm, Internet of Things (IoT), in which the devices form a network configured to reach goals common to all devices. The paradigm itself has gained increasing interest after the introduction of technologies that enable computing-like devices to share their states through the common network. These technologies include *Radio-frequency identification tags* (RFID) [10], *Near-field communication* (NFC) techniques and *Wireless sensor and actuator network* (WSAN) [27]. As an example of a network of devices trying to reach a common goal, consider an anti-theft system with motion detecting sensors. The sensors located differently interact with each other in order to detect unauthorized motion and prevent intruders. Many other applications of IoT can be found in [3, 27]. Some of the applications mentioned in [3] have been collected into Figure 1.

1.1 Related Research on Security

The technological advances alone are not sufficient to guarantee success for IoT-based solutions – the security of the technology is an important aspect as well. There are a variety of security threats related to IoT, as Roman et al. show in [26]: The threats are targeted at infrastructure, protocol and network security, data and privacy, identity management, trust and governance as well as at fault tolerance. For example, current Internet protocols may not meet the

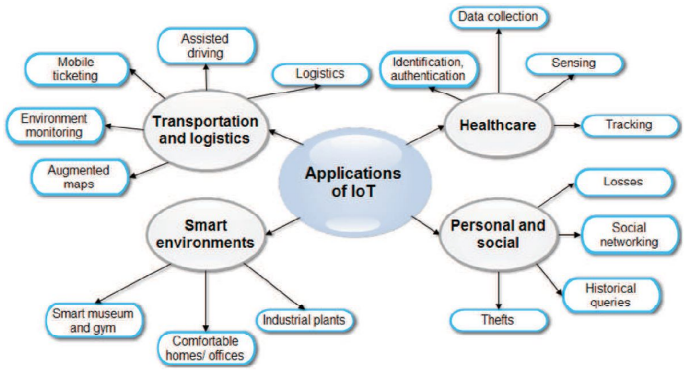


Figure 1 Applications of the Internet of Things (adapted from [3]).

security requirements of IoT, especially in the IP-based IoT as Heer et al. show in [18]. The physical security of IoT devices, e.g. tamper-resistance, is also an important aspect. An overview of different threats and possible solutions can be found in [27], whereas a more detailed threat analysis of RFID can be found in [10] and analyses of NFC from [17].

Several security threats are also identified by Kozlov et al. in [20]: there are numerous scenarios which endanger the security, trust or privacy of the IoT and these issues must be taken into account when considering legislation related to the Internet of Things. According to Weber [28], the IoT technology used by private enterprises must have resilience to attacks, authenticate the retrieved address and object information, have an access control and ensure client privacy. The privacy concerns are notable in situations in which the actions of individuals are monitored in a privacy-sensitive context. For example, a failed implementation of IoT related technology in a supermarket may violate the client privacy by enabling "the mining of medical data, invasive targeted advertising, and loss of autonomy through marketing profiles or personal affect monitoring" [29]. However, innovative ways of deploying privacy preserving IoT in privacy sensitive environments successfully are also possible: Abie et al. consider risk-based adaptive security framework for IoT in eHealth in [2]. More generally, techniques to achieve privacy preserving IoT applications have been considered widely. For example, privacy preserving electronic surveillance [11, 24] and even privacy preserving data mining [8] are possible by using a set of powerful cryptographic methods, called *secure multiparty computation* (SMPC).

1.2 Our Contributions

The solutions to achieve SMPC include a variety of protocols, e.g. *oblivious transfer* [25], *secure sum protocols* [8] and *garbled circuits* [30]. In this paper, we consider a way of achieving privacy preserving IoT applications by applying SMPC protocols. More specifically, we introduce a new way to realize privacy preserving electronic surveillance.We present a new tool in this context, *garbling*, which enables private computation on encrypted data.

The paper is organized as follows. We demonstrate the power of garbling in an example scenario presented in Section 2. In Section 3, we describe the realization of the privacy preserving electronic surveillance. In Section 4, we analyze the novel application of garbling in more details by considering its efficiency and what kinds of security goals are achieved by this technique. Section 5 concludes the paper and proposes directions for future research related to privacy preserving electronic surveillance.

2 Problem Setting: Privacy Preserving Electronic Surveillance System

Electronic surveillance is an application where privacy is a central concern. Many cryptographic tools have been proposed to ensure at least some level of privacy. In this paper, we present an innovative way of achieving privacy by using garbling in the context of electronic surveillance. Let us consider the following scenario as an example.

The client in this scenario is an elderly person living alone who wants to use the security service provided by a security company. The security company bases its service on an electronic surveillance system consisting of *Closed-Circuit Televisions* (CCTV) and various sensors (for example, motion detectors and/or sensors measuring the activity of the client). The security company collects data obtained by the system for further analysis. The analysis process contains tools for *data mining*, *pattern recognition* and *machine learning* – the intelligent surveillance system is supervised to react correctly on different situations. In certain situations (for instance, when the ongoing event seems to differ significantly from the usual course of events), the system evokes an alarm. The alarm together with an assessment of the situation enables the security company to react appropriately to the situation (e.g. call police/ambulance, send a guard from the company or just notify the client). The security company has outsourced its data center services into *a cloud* managed by a third-party company. The data from the surveillance system is stored and analyzed entirely in the cloud environment.

The main concern in this scenario is how the privacy for the client is managed. First obvious requirement is that anyone beyond the client, the security company and the cloud should not learn the contents of the data collected by the surveillance system. This requirement can be reached by simply encrypting the data on the client side and decrypting the data on the security company side. As a consequence, the security company and the cloud provider can follow everything that is going on at the client's home. A serious concern is that the third-party company managing the cloud can learn something about the client that could be used for unwanted or even malicious purposes. Thus it is highly justified to hide the raw data also from the cloud whereas the security company needs the raw data to be able to react correctly in the alarming situations. A solution to this is to use two-party computation between the security company and the cloud. This would allow the cloud to analyze the surveillance data without allowing the cloud to learn the raw data or the analytics tools.

However, this solution is still problematic. The security company should monitor the surveillance data of numerous customers in real time while the analysis on cloud is ongoing. This is not desirable because of several reasons. From the company's perspective, real-time monitoring is inefficient – several employees are tied to follow the monitors and are demanded to be in alert readiness all the time, even though nothing alarming is happening. From the client's perspective, the all-time surveillance is distracting and feels privacy violating – the security company should be able to study the raw data only in alarming situations and not otherwise.

To summarize the above analysis of the scenario, the implementation of the privacy preserving electronic surveillance system should have the following properties.

Confidentiality: All the information related to the electronic surveillance is kept secret from parties excluding the client, the security company and the third-party cloud. The third-party cloud performs the analysis on encrypted data. The cloud retrieves the encrypted surveillance data from the client and the encrypted surveillance data from the security company. The cloud is not allowed to find out the unencrypted surveillance data (the data is privacy-sensitive) or the analytics tool (the tool may be intellectual property of the security company). Depending on the contract between the security company, the client and the cloud, the final analysis result can either be concealed from the cloud or can be revealed to the cloud. These alternatives are discussed in more detail in Section 4. The security company is not allowed to retrieve the unencrypted surveillance data unless the analysis result yields an alarm. The client is not allowed to learn the implementation details of the analytics tool (the tool may be intellectual property of the security company).

Integrity: We may assume that the client is honest and therefore the surveillance data is authentic. Cloud can be honest, semi-honest or even malicious – a garbling scheme achieving certain level of security (explained in Section 4) guarantees that the analysis result is also authentic. Additionally, integrity of data in transit is protected, e.g. by using message authentication codes.

Entity authentication: The cloud does not need to authenticate itself, since all the data it processes is encrypted (in the case the cloud is not allowed to find out the analysis result). The security company and the client authenticate themselves when the system is first configured. After the authentication, we

assume that the channel between the client and the security company is confidential and authentic.

Access control: The security company is able to retrieve the unencrypted surveillance data only in the case in which the final analysis result yields an alarm. This requires that the analytics tools must not reveal the surveillance data. To ascertain this, the client and the security company use a trusted auditor that verifies appropriateness of the analytics tool (the analytics tool does not leak surveillance data in the final analysis report). We have described the different solutions for accessing the unencrypted surveillance data in Section 3.1.

Authorization: Access control, entity authentication and other security measures naturally require that access to various resources is properly authorized. For example, the client has to authorize the security company to have access to raw data in case of alarm and the security company has to provide authorization for the cloud provider in order to receive garbled data from the client.

Non-repudiation: We assume that the garbling protocol will achieve authenticity. This guarantees that the cloud cannot forge the garbled evaluation, and that the encrypted analysis result is authentic. We assume that the surveillance data is authentic. We also assume that the channel between the client and the security company is confidential and authentic. We also assume that the cloud service provider and the security company are not in the conspiracy against the client. Then log data collected by all parties can be used for non-repudiation purposes, see also discussion about logs in Section 3.2.

Availability: The system is naturally based on the assumption that raw data will be available for the security company in alarming situations. Related to this, there are threats purely concerning implementation. For example, burglars may cut off sending of data to the cloud. Also, the surveillance data stream may be interrupted on client side. As an example, robbers may break the CCTV equipment and sensors or the client may throw a towel on top of the surveillance camera etc.

To achieve these properties, we need an additional tool that enables the cloud to evaluate the analytics algorithms on the surveillance data without learning anything about the algorithms or data. A tool that fulfills this requirement is garbling. The formal definition of garbling and different security aspects related to garbling can be found in Section 4. In the following section, we concentrate on how the surveillance system using garbling should be implemented.

3 Operating Model: How to Build Privacy Preservation in the Surveillance System

In this section, we describe the operating model that aims at a solution for the problem presented in the previous section: How can the security company provide privacy preserving electronic surveillance to an elderly person even when all the data services of the company have been outsourced to a third-party cloud provider.

Our solution is based on a cryptographic tool for secure multiparty computation, *garbling*. Garbling enables *secure and private function evaluation*. A user who does not have enough computing resources utilizes a possibly untrustworthy evaluator, such as cloud, to accomplish the evaluation of some function f on argument x. However, the user wants to keep both the function f and the argument x secret from the evaluator. The user and the cloud agree on using a garbling scheme that works as follows. First, the user garbles function f and its argument x and obtains garbled function F and garbled argument X. The user gives F and X to the evaluator who runs the garbled evaluation to get the garbled value $Y = \mathrm{Ev}(F, X)$. Now, either the evaluator or the user ungarbles Y to get the final value y, which is equal to the result of the original evaluation $y = \mathrm{ev}(f, x)$.

In the scenario presented in the previous section, the electronic surveillance system is designed under the IoT paradigm, making the computational resources of the system limited. This means that the surveillance data analysis must take place outside the surveillance system, for example at the data center of the security company. Since the data center services are outsourced, the analysis takes place in the cloud managed by a third-party company. The surveillance data from the client's home is privacy sensitive as are the analytics tools of the security company, so the three parties agree on using a garbling scheme. Figure 2 illustrates the scenario showing also how the garbling scheme is used by the different parties.

3.1 Responsibilities of the Different Parties

The surveillance data is garbled on the client side. In this way, neither the cloud nor the security company is able to access the raw data directly. The garbled surveillance data is sent to the cloud for analysis. The security company has garbled its analytics tools that act as the function to be evaluated on cloud. After receiving both the garbled data and the garbled analytics tools, the cloud runs the garbled evaluation getting the garbled final value. If the cloud is allowed to decrypt the garbled analysis result, then it decrypts the

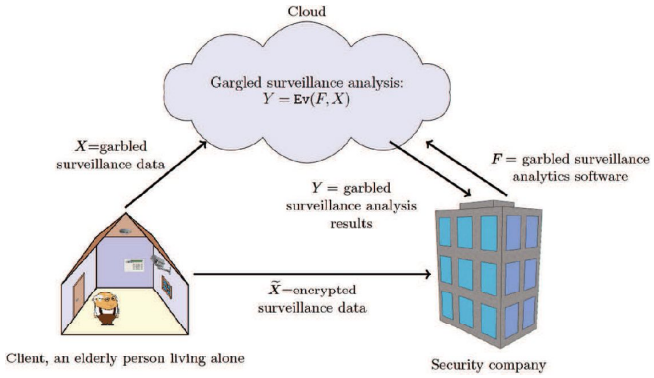


Figure 2 Scenario about electronic surveillance.

garbled value getting the final outcome of the analysis. This final outcome is sent to the security company for further investigation. However, letting the cloud learn the analysis results might not be convenient – it can violate the privacy in similar manner as the actual surveillance data. Thus, a more convenient way of implementation is that the cloud sends the garbled analysis outcome to the security company for further investigation. Now, the security company ungarbles the data received from the cloud. Based on the analysis outcome, the security company takes corresponding actions (e.g. by visiting the home or calling the police, an ambulance, a social worker, the person's relatives etc.).

A straight-forward way of reacting to the alarm situation for the security company is that a guard from the company visits the client for further inspection in spite of what has caused the alarm. Then, the security company does not have or even need an access to the raw data (in Figure 2 this means, that no encrypted surveillance data $\widetilde{X}$ is provided to the security company). However, this is not a practical approach. The company should adaptively react to different alarms – for example a robbery should cause different reactions than the client staying suspiciously long in the shower.

To adaptively react to the various situations, the security company needs an access to the raw data. Granting the security company access to the raw data with no restrictions is not a satisfactory solution since it would violate the requirements set to the system: the raw data should be accessible for the security company only in alarming situations (see Confidentiality requirement in Section 2). One possible way of realizing the access control

would be to encrypt the raw data twice, independently for the cloud and for the security company. The raw data is protected against the cloud by garbling the argument x. Garbling x is modeled by encrypting x using the encryption algorithm En together with encryption key *e*. Respectively, the algorithm De with the decryption key d is used to ungarble *Y* to final value y. The encryption against the security company utilizes an independent encryption algorithm $\widetilde{\mathrm{En}}$ with key $\widetilde{e}$. The decryption algorithm $\widetilde{\mathrm{De}}$ with the decryption key $\widetilde{d}$ is used to recover the raw data from the encrypted data $\widetilde{X}$ – this key is called *recovery key* to avoid confusion between the two keys d (which is needed for ungarbling) and $\widetilde{d}$ (which is needed for recovering x from $\widetilde{X}$).

The surveillance data is collected in pieces and these pieces are then encrypted and sent to the cloud (X) and to the security company ($\widetilde{X}$) by the client. Data pieces may contain overlaps so that successful reconstruction of the course of events without gaps is possible. Since the cloud does not learn the keys (e, d), and hence learns nothing privacy – violating about the surveillance data x or the analysis result y, the same keys (e, d) may be used for many evaluations by the analytics tool.

The same does not hold for the keys ($\widetilde{e}$, $\widetilde{d}$) related to the encryption of surveillance data against the security company. If the same keys ($\widetilde{e}$, $\widetilde{d}$) were used, then the security company would-be able to follow all the surveillance data after accessing the keys ($\widetilde{e}$, $\widetilde{d}$) for the first time – even in the non – alarming situations. This clearly violates the privacy policy we have set to the system. Thus, a more sophisticated access control method is needed. We have identified the following two approaches to implement access to the recovery key $\widetilde{d}$.

3.2 The First Approach

In this approach, the recovery key $\widetilde{d}$ for recovering the encrypted surveillance data $\widetilde{X}$ is possessed by the security company. However, the key $\widetilde{d}$ must be protected by an electronic seal because otherwise the company could decrypt all the surveillance data and not only the data related to alarms. The company is allowed to break the seal whenever the analysis yields an alarm. After breaking the seal, the company uses the recovery key to obtain the actual surveillance data consisting of the moments some time before and after the alarm.

The above approach requires a countermeasure to detect unauthorized access to the surveillance data. One possible way is to utilize event logging. Each of the three parties related to the surveillance are maintaining their

own independent event logs. The independent logs contain information that can be derived from the activities of the different parties (for example, the company logs access to the raw data together with a synopsis of the analysis results). These three independent logs can in principle be compared to detect unauthorized or illegitimate access to the backup data. Of course, the different logs can be forged and the comparison does not work in the desired way in case there are conspiracies between the parties but solving conspiracy issues is not in the scope of this paper.

In this approach, the efficiency of the implementation depends on the efficiency of the used garbling scheme as well as the efficiency of the used independent encryption scheme $\mathcal{E} = (\widetilde{\mathrm{KeyGen}}, \widetilde{\mathrm{En}}, \widetilde{\mathrm{De}})$. The efficiency of garbling schemes is discussed in more detail in Section 4.3. The efficiency of the encryption scheme $\mathcal{E}$ is due to the choice of the security company. For example, $\mathcal{E}$ may be AES-128.

3.3 The Second Approach

In this approach, the recovery key $\widetilde{d}$ is in client's possession. Since the security company does not possess the decryption key $\widetilde{d}$ of $\widetilde{X}$, the company cannot monitor the data unless it is handed the decryption key. The company should be able to get the decryption key only in alarming situations. A straight-forward way of implementing the access control into the recovery key $\widetilde{d}$ is to use timestamped key management (see [19] for further information). The security company can access the keys $\widetilde{d}$ related to the raw data having certain timestamps that correspond to the time of the alarm detection as well as the data from some moments before and after the alarm detection. The client can later check which keys have been sent to the security company and, if needed, check the corresponding raw data.

There is also a more innovative way of implementing the access control into the recovery key $\widetilde{d}$. Informally, our idea is to send the recovery key $\widetilde{d}$ to the security company via the cloud in such a way that the cloud does not learn the recovery key. Moreover, the security company will receive the key only in the case that the final analysis results yield an alarm. Next, we explain in more details, how this functionality can be implemented.

For simplicity, let us assume that the final surveillance analysis result is either alarm or no alarm, i.e. $y \in \{\text{alarm, no alarm}\}$. Now, we want that the security company gets $\widetilde{d}$ whenever y = alarm. This can be reached by attaching first the recovery key $\widetilde{d}$ to the surveillance data, i.e. $x_m = (x, \widetilde{d})$. This argument is then garbled and sent to the cloud, thus the cloud is not able

to learn $\widetilde{d}$. The function f needs to be modified in order to be able to handle the new argument type. We define the modified function as follows

$$f_m(x_m) = \begin{cases} (y, \widetilde{d}) & \text{if } y = \text{alarm} \\ (y, \varepsilon) & \text{otherwise (where } \varepsilon \text{ is the empty string)} \end{cases}$$

The garbling scheme works in a similar manner as before. The cloud gets the garbled argument X_m from the client and the garbled function F_m from the security company. The cloud computes the garbled value Y_m and sends it to the security company. The security company ungarbles Y_m and gets $y_m = (y, \beta)$. Here, $\beta \in \left\{\widetilde{d},\ \varepsilon\right\}$ depends on whether y = alarm or not.

The modifications in x and f now give the required functionalities. First of all, the security company gets the recovery key $\widetilde{d}$ only in the case y yields an alarm. Secondly, sending the key via the cloud is not insecure – the key remains garbled during the whole garbled evaluation in similar manner as the argument and the function.

The efficiency of implementation using this approach depends on the efficiency of the used garbling scheme and the efficiency of the used encryption scheme $\mathcal{E} = \left(\widetilde{\text{KeyGen}}, \widetilde{\text{En}}, \widetilde{\text{De}}\right)$. However, the function f_m and the argument x_m are more complex than in the first approach, since the argument x_m contains the recovery key $\widetilde{d}$ and the function f_m needs to process $\widetilde{d}$ somehow. This means, that the second approach is not as efficient as the first approach. On the other hand, the second approach provides better control over the use of the recovery key $\widetilde{d}$.

4 Implementation of the Surveillance Service

In this section, we describe garbling schemes in more details. We start by defining the concept after which we discuss the different security measures for garbling schemes. We also discuss which of the security concepts are ideal for the use in the context of privacy preserving electronic surveillance.

4.1 Building Blocks

As mentioned earlier, our main building block to construct a privacy preserving and cloud-assisted surveillance system is garbling. Garbling enables surveillance data to be analyzed on cloud environment without compromising the privacy of the client or revealing business secrets in the form of the analytics tool.

The surveillance analytics tool may contain algorithms e.g. for anomaly detection [7, 9] (to detect the abnormal situations among the normal situations), and for machine learning. Recently, a method for running machine learning algorithms on encrypted data has been proposed [16].

One possible way of teaching the analytics tool is the following. Before the surveillance starts, the company and the client may have collected data from normal situations. These labeled situations together with the data from the surveillance system act as the training data for the semi-supervised learning (see [31] for more information) algorithm that now helps in doing the final analysis together with the other algorithms. We do not concentrate on the exact implementation of the analytics tool as our focus is on the tools enabling the privacy preserving surveillance.

4.2 Formal Definitions for Garbling

Formally, a garbling scheme is a 6-tuple of algorithms, (KeyGen, Ga, En, De, Ev, ev). The last component of the tuple is the evaluation algorithm ev: an algorithm that computes the value of function f on argument, i.e. $y = f(x)$. In our scenario, the function f is the surveillance analytics tool and the argument x is the surveillance data. To hide the analytics tool and the surveillance data, both f and x are garbled. To do this, first key generation algorithm KeyGen is called to generate three keys $(g,\ e,\ d)$. The garbling algorithm Ga computes the garbled function F = Ga(g, f) based on the function f and garbling key g. The encryption algorithm En computes the garbling X = En(e, x) based on argument x and encryption key e. The garbled evaluation function (the garbled analytics tool) computes the garbled value Y = Ev(F, X) (garbled analysis). Finally, the decryption algorithm De ungarbles Y and returns the final analysis result y = De(d, Y) by using the decryption key d issued by the KeyGen algorithm. Note that the final analysis result must be the same despite of the method used for evaluation: the garbled evaluation must yield the same final analysis result as the actual evaluation, i.e. ev$(f, x) = y$ = De(d, Y) = De(d, Ev(F, X)). The garbled evaluation process is illustrated in Figure 3. For further details, consult e.g. [22].

In the example scenario presented in this paper, a garbling scheme G = (KeyGen, Ga, En, De, Ev, ev) is used as follows. The client in the scenario uses the algorithms KeyGen and En. The security company uses algorithm Ga. The cloud uses algorithm Ev. Depending on the case, either the cloud or the security company uses algorithm De. Figure 4 illustrates how the different algorithms are run by different parties in the example scenario.

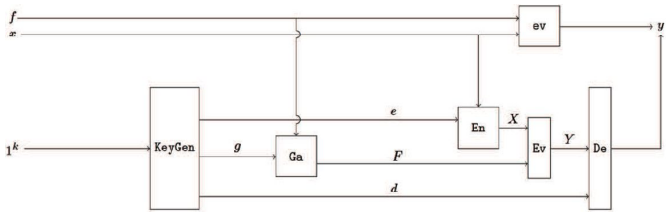


Figure 3 The components and the workings of a garbling scheme. The diagram is adapted from [22].

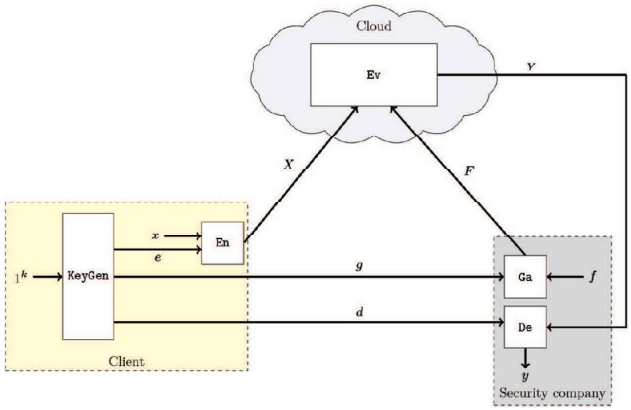


Figure 4 Garbling scheme in the surveillance scenario.

In the illustration, we assume that the channel between the client and the security company assures data integrity, authenticity and confidentiality. This is not assumed for the channel between the client/the security company and the cloud. We also present the situation in which the security company is the party ungarbling Y.

4.3 Security Considerations

In this section, we first introduce different security concepts for garbling schemes. Exact definitions for each concept can be found in literature [6, 5, 21–23]. Then, we analyze which of the security concepts meet the requirements for the privacy preserving electronic surveillance system proposed in the previous section.

Every security concept can be characterized by *security notion* and *level of reusability*. The security notion tells what kind of information about the function f and the argument x is allowed to be leaked. The notion *function and argument hiding* means that the garbling scheme is allowed to leak $f(x)$, but neither f nor x. The notion *function, argument and final value hiding* does not allow the garbling scheme to leak any of $f,\ x$ or $f(x)$. The notion *matchability-only* does not allow the garbling scheme to leak f nor x, but when evaluating f on two different arguments x_1 and x_2, the garbling scheme is allowed to leak whether $f(x_1){=}\ y_1 = y_2 = f(x_2)$. Note that the names of the notions differ from the ones used in literature. We use non-standard names to distinguish what we mean by privacy in the example scenario and privacy related to garbling schemes. The notion *function and argument hiding* corresponds the notion *privacy* in [5, 23]. The notion *function, argument and final value hiding* corresponds the notion *obliviousness* in [5, 23].

The security notions described above deal with secrecy. The *authenticity* property can also be formalized for garbling schemes. Authenticity guarantees that an adversary is unable to create a garbled value Y from a garbled function F and its garbled argument X such that $Y \neq F(X)$ but which will be considered authentic. For a formal definition of authenticity for garbling schemes, consult [6]. The authenticity of garbling schemes is needed to fulfill requirement 2, demanding that the final analysis result must be authentic. Thus, the garbling scheme used in the example scenario must achieve authenticity in the sense explained in [6].

Another characteristic of a garbling scheme is *the level of reusability*. The level of reusability tells how many times the same garbled function can be securely used for different arguments. The first reusability level enables only one-time use of the same garbled function [5, 6, 21] whereas higher levels of reusability enable several or even arbitrary reuse of the garbled function [15, 23].

Let us first recall Confidentiality, Integrity, Entity authentication, Access control, Authorization, Non-repudiation and Availability requirements presented in Section 2. The Confidentiality requirement says that the unencrypted surveillance data must be kept secret from third parties, including the cloud. Moreover, the analytics tool must be hidden from third-party cloud. From the garbling point-of-view, this means that the garbling scheme should be at least function and argument hiding. The Confidentiality requirement also tells that hiding the final analysis result depends on the contract between the client, security company and the cloud.

We have identified three possible configurations of the surveillance system all of which set different security requirements to the garbling scheme in use. In all three cases, the surveillance data as well as the surveillance data analytics tools are kept secret from the cloud. The differences in the configurations are related to the final analysis results: are the analysis results kept totally, partially or not at all secret from the cloud. Let us next provide more details of these three different configurations.

Case 1: The cloud is allowed to learn nothing about the resulting analysis. This means that the surveillance data, the analytics tool and the final analysis result are all hidden from the cloud. From the garbling point-of-view, this is the same as hiding the function, the argument and the final value. This is desirable, because the third-party company may use the information about the analysis for its own purposes that might be unwanted by the client. Thus, the garbling scheme must leak none of f (the analytics tool), x (the surveillance data) or $y = \mathsf{ev}(f, x)$ (the analysis result) to the cloud. A garbling scheme that is function, argument and final value hiding meets these requirements.

Case 2: The cloud is allowed to learn indirect information about the analysis result but possibly not the actual content of the final analysis. From the garbling point-of-view, this is the same as hiding the function, the argument and the final value but leaking some information about the final value. One justification for this weaker privacy requirement is the following: the cloud service provider may anyway be able to find out the actions of the security company related to certain garbled analysis results. For example, the cloud has found that garbled analysis result Y yields a call to the police. When the same garbled analysis result Y is found later again on the cloud, the cloud service provider is able to predict the reaction of the security company – the security company will probably call to the police.

Thus, we could require that the cloud service provider cannot find out f, x or y but it is able to find out whether the certain Y yields similar actions as before. For the garbling scheme this means that the scheme should not leak $f,\ x$ or y but it may leak whether $f(x_1) = y_1 = y_2 = f(x_2)$ when computing $\mathsf{ev}(f_1, x_1)$ and $\mathsf{ev}(f_2, x_2)$. This is exactly what a garbling scheme achieving matchability-only security provides.

Case 3: The cloud service provider is allowed to learn the final analysis result. From the garbling point-of-view, this means that the garbling scheme is allowed to leak the final value y whereas it must hide the function and the argument. However, the Confidentiality requirement tells that the cloud is allowed to learn nothing about the ungarbled surveillance data, meaning

that the final analysis cannot contain parts of surveillance data. To assure this, the final value could be something else than a review of the surveillance data. It may also be *the type of alarm*, like no alarm, low urgency, medium urgency, high urgency etc. or simply alarm/no alarm. Now it may under some circumstances be acceptable to let the cloud provider to know the type of the alarm. This means the garbling scheme should hide f and x but it may leak y. This is exactly what a function and argument hiding garbling scheme provides. However, it is questionable whether the cloud should generally learn that there is an alarming situation at the client. This violates the requirement that the third – party cloud should learn nothing about the surveillance, not even the fact that the security company is being alarmed.

The above reasoning suggests that the garbling scheme should either be function, argument and final value hiding or achieve matchability-only. From the practical point of view, matchability-only is preferable as it has been shown in [21–23] that it is at least as easy to achieve matchability-only as to be function, argument and final value hiding. Moreover, for practical reasons one should be able to use the same garbled analytics tool for several garbled surveillance data entries. For the garbling scheme this converts to reusability. Thus we suggest that the applied garbling scheme should be reusable as well as achieve matchability-only and authenticity. This guarantees that the surveillance is privacy-preserving since the third parties do not learn the ungarbled surveillance data or the ungarbled analysis result.

4.4 Efficiency Considerations

The above concepts do not restrict the computation method for evaluating function f on argument x – the function f may represent models such as a circuit, a Turing machine or a random-access machine. Methods for garbling various computational models have been constructed. For example, there exist garbling schemes for circuits [30], Turing machines [14] and random-access machines [13].

The choice of the computation method affects the efficiency of the garbling scheme. Choosing circuits over Turing machines has at least two unfortunate consequences. The first consequence is related to the running time of circuits. The running time of a circuit is constant, implying that evaluating a circuit with any input takes the worst-case running time. This is not the case for Turing machines. Another unfortunate consequence is related to the size of the garbled function F. Turing machines outperform circuits also in this aspect: the size of garbled circuit is as large as the running time of the algorithm where

as the size of the garbled Turing machine depends only on the description of the algorithm and not on the input value x. [14]

On the other hand, using circuits as the computation method has benefits over Turing machines when considering the costs of constructing the garbling scheme. Garbled circuits are known to have efficient constructions [4] where as such are not known for garbled Turing machines. Garbled Turing machines typically use fully-homomorphic encryption [12] as a building block which causes inefficiency in the construction.

Next we provide some numbers on the efficiency of garbling. The values have been collected from [4], in which three different garbling schemes have been experimented by using JustGarble (the source code is open-source and is available in [1]) system on an x86-64 Intel Core i7-970 processor clocked at 3.201 GHz with a 12MB L3 cache. The three garbling schemes are based on a function and value hiding garbling scheme Garble1 presented in [6]. All three presented garbling schemes are based on *dual-key cipher*. The difference in the three schemes is the different optimization techniques used to reduce the evaluation time. For more details, consult [4].

On a circuit having 15.5 million gates, of which 9.11 million gates are XOR gates, the most efficient garbling scheme *GaX* uses approximately 0.49 seconds to garble the circuit and 0.23 seconds to evaluate the garbled circuit [4]. This shows that the time for evaluating and garbling using *GaX* is efficient even on quite large circuits, meaning that even complex algorithms represented as circuits can be efficiently garbled and evaluated with *GaX*.

Using this measurement data presented in [4], we can estimate the efficiency of our solution for the example scenario as follows. Let us assume that the garbling scheme *GaX* is used. Let us further assume that the analytics tool is presented as a circuit having approximately 15.5 million gates. If the client sends surveillance data at rate of 0.5 kilobits/second then an analysis result is received by the security company approximately once per second. Achieving the low sending rate of 0.5 kilobits per second requires some pre-processing of the surveillance data on client side, e.g concerning the captured video stream where raw data is accumulated in much higher data rate. As a summary, these figures seem to be acceptable from practical point-of-view.

There are some known issues related to the use of scheme *GaX*. The garbled argument *F* is constructed at the same time as the keys (e, d), meaning that one party needs to possess both the function and the argument. We can solve this problem in two ways. First one is to let a trusted authority to run the garbling algorithm Ga with the garbling key g obtained from the client and the function f obtained from the security company. Another solution would be

that the client and the security company use a secure multiparty computation protocol for computing $\mathrm{Ga}(g, f)$ together in such a way that the security company does not learn g and the client does not learn f. Unfortunately, both solutions add to the complexity of the system and increase the time to garble.

Another problem in using any of the garbling schemes from [4] is that these garbling schemes are not reusable. This means that every time a new surveillance data entry is ready to be processed, both the surveillance data entry and the analytics tool need to be garbled. This implies that big amount of the total computation happens in the client side, and therefore the benefit of using cloud is questionable. In garbling schemes supporting reusable garbled circuits, the analytics tool represented as a circuit is garbled only once which would increase the overall efficiency of the garbling scheme. In our scenario, this would correspond to a situation where most of the computation load can be moved to the cloud. Unfortunately, no efficient constructions for reusable garbled circuits are known.

5 Discussion

Applications following the Internet of Things paradigm have increased rapidly, even to the extent that the security, trust and privacy related to the applications have not been able to keep up with the progress. Especially, privacy preservation seems to be one of the hottest topics related to the Internet of Things. One application that is regarded as privacy violating is electronic surveillance, at least in the private premises such as homes. On the other hand, there is a need to monitor private homes in order to track emergencies and protect the customers from various threats. We are confronting the challenging task of creating a privacy preserving electronic surveillance system.

In this paper, we have presented a novel way of using garbling schemes to achieve privacy preservation in electronic surveillance. We illustrated the power of garbling with an example scenario. An elderly person living alone is subscribing to a security service that includes electronic surveillance. The surveillance data is analyzed by a security company that has outsourced its data services onto a third-party cloud. Garbling allows the private analysis of the surveillance data on cloud – the cloud learns neither the surveillance data nor the analytics tool.

The example scenario is not the only possible application for garbling. As another related example, a monitoring system can be installed in the homes of people using the services for *assisted living*. The party monitoring the

data should not learn the habits of the person using the system beyond the situations in which the person needs help. In this scenario, the security company may provide the monitoring services to the company providing the services for assisted living. This makes the privacy preservation even more complex task.

A variant of our example scenario presented in this paper is that the security company does not use third-party cloud services and instead does all the analysis itself. The operation of the parties present in this variant scenario resembles the operation of the same parties in the example scenario. However, there is one fundamental difference: the party possessing the analytics algorithms is the same as the party evaluating the analytics algorithms. This means that the security company first garbles the analytics algorithm as before, but then evaluates the garbled analytics algorithms on the garbled data received from the client.

In this case, the garbled evaluation might directly give the final analysis result y as hiding the result from the security company itself is useless. Moreover, the garbling of the analytics tool is not essential since the same party (the security company) both possesses the analytics tools and is responsible for the evaluation. Hence, the variant scenario is more efficient than the original scenario. However, moving the computation load from the cloud to the security company requires that the computational resources on the security company side should increase.

To conclude, we have found a novel solution to provide privacy preservation in an electronic surveillance system utilizing the Internet of Things paradigm. The biggest advantage in our solution is that garbling provides flexibility in the system. The surveillance analytics tool can be almost anything, from comparisons to complex machine learning algorithms. Moreover, the function f can be changed without need to reconfigure the whole system, easing the system maintenance.

The biggest obstacles for implementing the described system we have described is related to the implementation of efficient garbling schemes. There exist efficient garbling schemes (see Section 4.3) that support one-time use of the garbling scheme. But regarbling the analytics tool again for every surveillance data entry is not optimal from practical point-of-view. Reusable garbled circuits would solve this problem – however efficient garbling schemes supporting reusable garbled circuits are not known.

Future research may solve the problem of efficient reusable garbled circuits. Moreover, exploring further targets for innovative use of garbling in the context of IoT is important. An interesting target would be larger and more

complex systems having more parties, for example a scenario where a person uses services for assisted living. In this scenario, we would have four parties – the client, the assisted living service provider, the security service provider (providing the devices for monitoring the client) and the third-party cloud service provider.

Acknowledgments

Authors would like to thank anonymous referees for valuable suggestions that have improved the paper quality significantly. This work was supported by the Academy of Finland project "Cloud Security Services" which is greatly appreciated.

References

[1] JustGarble. http://cseweb.ucsd.edu/groups/justgarble/. Accessed: 2014-10-13.

[2] H. Abie and I. Balasingham. Risk-based Adaptive Security for Smart IoT in eHealth. In *Proceedings of the 7th International Conference on Body Area Networks,* BodyNets'12, pages 269–275, ICST, Brussels, Belgium, Belgium, 2012. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[3] L. Atzori, A. Iera, and G. Morabito. The Internet of Things: A Survey. *Computer Networks,* 54(15): 2787–2805, 2010.

[4] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway. Efficient garbling from a fixed-key blockcipher. In *Proc. of Symposium on Security and Privacy 2013,* pages 478–492. IEEE, 2013.

[5] M. Bellare, V. T. Hoang, and P. Rogaway. Adaptively secure garbling scheme with applications to one-time programs and secure outsourcing. In *Proc. of Asiacrypt 2012,* volume 7685 of LNCS, pages 134–153. Springer, 2012.

[6] M. Bellare, V. T. Hoang, and P. Rogaway. Foundations of Garbled Circuits. In *Proc. of ACM Computer and Communications Security (CCS'12)*, pages 784–796. ACM, 2012.

[7] V. Chandola, A. Banerjee, and V. Kumar. Anomaly Detection: A Survey. *ACM Comput. Surv.,* 41(3): 15: 1–15: 58, July 2009.

[8] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for Privacy Preserving Distributed Data Mining. *SIGKDD Explor. Newsl.,* 4(2): 28–34, Dec. 2002.

[9] T. Dunning and E. Friedman. *Practical Machine Learning: A New Look at Anomaly Detection.* O'Reilly Media, 2014.

[10] S. Evdokimov, B. Fabian, O. Günther, L. Ivantysynova, and H. Ziekow. RFID and the Internet of Things: Technology, Applications, and Security Challenges. *Foundations and Trends@in Technology, Information and Operations Management,* 4(2):105–185, 2011.

[11] K. B. Frikken and M. J. Atallah. Privacy Preserving Electronic Surveillance. In *Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society,* WPES '03, pages 45–52, New York, NY, USA, 2003. ACM.

[12] C. Gentry. *A Fully Homomorphic Encryption Scheme.* PhD thesis, Stanford University, 2009. crypto. stanford. edu/craig.

[13] C. Gentry, S. Halevi, S. Lu, R. Ostrovsky, M. Raykova, and D. Wichs. Garbled RAM Revisited. In *Proc. of* 33^{rd} *Eurocrypt,* volume 8441 of LNCS, pages 405–422, 2014.

[14] S. Goldwasser, Y. Kalai, R. Popa, V. Vaikuntanathan, and N. Zeldovich. How to Run Turing Machines on Encrypted Data. In *Proc. of* 33^{rd} *CRYPTO,* volume 8043 of LNCS, pages 536–553, 2013.

[15] S. Goldwasser, Y. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable Garbled Circuits and Succinct Functional Encryption. In *Proc. of the* 45^{th} *STOC,* pages 555–564. ACM, 2013.

[16] T. Graepel, K. Lauter, and M. Naehrig. ML Confidential: Machine Learning on Encrypted Data. In *International Conference on Information Security and Cryptology – ICISC 2012, Lecture Notes in Computer Science, to appear*. Springer Verlag, December 2012.

[17] E. Haselsteiner and K. Breitfuß. Security in near field communication (NFC). In *Workshop on RFID security*, pages 12–14, 2006.

[18] T. Heer, O. Garcia-Morchon, R. Hummen, S. L. Keoh, S. S. Kumar, and K. Wehrle. Security Challenges in the IP-based Internet of Things. *Wirel. Pers. Commun.*, 61(3): 527–542, 2011.

[19] A. V. D. M. Kayem, S. G. Akl, and P. Martin. Timestamped Key Management. In *Adaptive Cryptographic Access Control*, volume 48 of *Advances in Information Security*, pages 61–74. Springer US, 2010.

[20] D. Kozlov, J. Veijalainen, and Y. Ali. Security and Privacy Threats in IoT Architectures. In *Proceedings of the 7th International Conference on Body Area Networks*, BodyNets'12, pages 256–262, ICST, Brussels, Belgium, Belgium, 2012. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[21] T. Meskanen, V. Niemi, and N. Nieminen. Classes of Garbled Schemes. *Infocommunications Journal*, V(3): 8–16, 2013.

[22] T. Meskanen, V. Niemi, and N. Nieminen. Garbling in Reverse Order. In *The 13th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-14)*, 2014.

[23] T. Meskanen, V. Niemi, and N. Nieminen. Hierarchy for Classes of Garbling Schemes. In *Proc. of Central European Conference on Cryptology (CECC'14)*, 2014.

[24] V. Oleshchuk. Internet of things and privacy preserving technologies. In *1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology, 2009. Wireless VITAE 2009.*, pages 336–340, 2009.

[25] M. O. Rabin. How to Exchange Secrets with Oblivious Transfer. Technical report tr-81, Aiken Computation Lab, Harvard University, 1981.

[26] R. Roman, P. Najera, and J. Lopez. Securing the Internet of Things. *Computer*, 44(9): 51–58, Sept 2011.

[27] O. Vermesan, M. Harrison, H. Vogt, K. Kalaboukas, M. Tomasella, K. Wouters, S. Gusmeroli, and S. Haller. *Vision and Challenges for Realising the Internet of Things*. European Commission, Information Society and Media, 2010.

[28] R. H. Weber. Internet of Things – New security and privacy challenges. *Computer Law & Security Review*, 26(1): 23–30, 2010.

[29] J. S. Winter. Surveillance in Ubiquitous Network Societies: Normative Conflicts Related to the Consumer In-store Supermarket Experience in the Context of the Internet of Things. *Ethics and Inf. Technol.*, 161: 27–41, 2014.

[30] A. Yao. How to generate and exchange secrets. In *Proc. of* 27^{th} *FOCS, 1986.*, pages 162–167. IEEE, 1986.

[31] X. Zhu. Semi-Supervised Learning Literature Survey. http://pages.cs.wisc.edu/∼jerryzhu/pub/ssl_survey.pdf, July 2008.

Biographies

T. Meskanen had his PhD in 2005. Since then he has been working as a researcher and lecturer at University of Turku. His main research interests are cryptography and public choice theory. His email address is tommes@utu.fi.

V. Niemi is a Professor of Mathematics at the University of Turku, Finland. Between 1997 and 2012 he was with Nokia Research Center in various positions, based in Finland and Switzerland. Niemi was also the chairman of the security standardization group of 3GPP during 2003–2009. His research interests include cryptography and mobile security. Valtteri can be contacted at valtteri.niemi@utu.fi.

N. Nieminen is a doctoral student at Turku Centre for Computer Science, Department of Mathematics and Statistics at the University of Turku. Her research interests include cryptography and its applications. Contact her at nmniem@utu.fi.

Paper VIII

Privacy-Preserving Security Monitoring for Assisted Living Services

N. Nieminen and A. Lepistö (2015). In *Proceedings of the 17th International Symposium on Health Information Management Research (ISHIMR 2015)*, pages 189-199. York St. John Universty & University of Sheffield

VIII

Privacy-Preserving Security Monitoring for Assisted Living Services

Noora Nieminen[1,2] and Arto Lepistö[1]

[1]*Department of Mathematics and Statistics, University of Turku, Finland*

emails: {nmniem,alepisto}[at]utu.fi

[2]*Turku Centre for Computer Science (TUCS), Turku, Finland*

Novel techniques, such as cloud computing or the Internet of Things, are increasingly popular but they also have some real challenges. One of the main concerns is privacy and security, which are essential assets especially in health related services. Remote monitoring is a novel technique combining IoT and cloud techniques benefiting both the client and the assisted living service provider. In this paper, we propose a new way of achieving privacy-preserving remote health monitoring. We design a way of using garbling schemes for implementing privacy-preserving monitoring. We also provide an implementation for garbling the health assessment procedure.

Keywords

assisted living, garbling scheme, health monitoring, privacy

1. Introduction

In past years, emerging technologies have revolutionised health and security services: cloud computing and Internet of Things. Cloud computing has provided resources for storing and processing growing amount of data. Internet of Things (IoT) in turn allows telemonitoring. Personal health data collected by a network of different sensors and monitors can be sent to medical centres in real time over the Internet.

Even though both IoT and cloud computing provide flexibility and innovations the two techniques are not without problems. One of the most severe problems is related to security and trust of IoT and cloud-based solutions [1]. Some of the security issues are related to implementation, for example protocol and network security, physical security and tamper resistance [2,3]. There are also privacy and trust related threats related to the use of IoT and cloud-based applications [4].

For many network-based solutions, privacy is the main concern. The consequences of poorly designed and implemented IoT applications are severe. As an example, a badly designed and maintained IoT system in a supermarket may violate the client privacy. Bad design might, for example, enable *the mining of medical data, invasive targeted advertising, and loss of autonomy through marketing profiles or personal affect monitoring* [5, p.11].

Various privacy-preserving techniques have been introduced to solve privacy issues. There are several papers on privacy-preserving electronic surveillance [6, 7, 8], privacy-preserving data mining [9] and eHealth [10]. Most of the proposed privacy-preserving techniques in [6-10] use cryptographic techniques to achieve privacy preservation.

1.2 Contributions

In this paper, we propose a novel method for privacy-preserving holistic security monitoring in assisted living. By *holistic security monitoring*, we mean a security service in which different aspects of personal security, including health, physical integrity and protection of personal property, are simultaneously monitored and assessed. First, we introduce a context of use for the privacy-preserving security monitoring technique. Then we describe our implementation which is based on *garbling schemes*. We prove that garbling schemes fulfil the requirement for privacy preservation. We also demonstrate efficiency of garbling with a test case.

This paper is organised as follows. In section 2 we present related research, including network-based solutions in eHealth, assisted living and electronic surveillance. In Section 3 we present the scenario in which we will apply our privacy-preserving security monitoring technique. Moreover, in Section 3 we briefly explain how the assisted living situation surveillance can be implemented in privacy-preserving manner. In Section 4 we discuss the efficiency of our proposed remote security monitoring system. Section 5 concludes the paper.

2. Related work

In this section, we briefly present research related to network-based solutions in electronic surveillance, eHealth and assisted living. Then we discuss cryptographic solutions proposed for privacy-preserving services in network-based solutions including remote monitoring.

2.1 IoT and cloud based solutions in eHealth and assisted living

Medical records are considered highly privacy-sensitive. Therefore, eHealth applications utilising network technologies must be carefully designed and implemented. Appropriate implementation guarantees that patients' privacy might not be compromised as easily. There is a variety of issues related to cloud-based eHealth solutions as Löhr et al. show in [11, p.221]: "current e-health solutions and standards mainly focus on network security and access control policies, however, they do not address the client platform security appropriately."

Even though threats against eHealth and other network-based health solutions are serious, network-based solutions seem to give great advantages. As Trief et al. show in [12], the advantages of eHealth and telemedicine techniques are enormous not only for eHealth service providers but also for patients. This explains why eHealth and telemetry services are increasingly popular and why these kinds of services are being developed further.

One central research direction is to study how privacy preservation in network-based health services could be achieved in efficient and user-friendly manner. Various techniques to preserve privacy have already been introduced. For example, Layouni et al. propose in [13] a protocol that allows telemonitoring for eHealth but only at patients' approval. A method for remote ECG analysis has been proposed by Barni et al. in [14]. Even a method to automate emergency healthcare process by utilizing cloud services has recently been studied by Karthikeyan and Sukanesh in [15] as well as by Poulymenopoulou et al. in [16]. All these systems are designed for health institutions that collect vital data from patients. Systems for patients' personal use have also been proposed: mobile cloud-assisted eHealth services have been discussed in [17, 18, 19]. The main concern for personal use is privacy-preserving data storage and retrieval as well as auditability for misusing health data. A system achieving these three properties has recently been developed by Tong et al. in [20].

Using IoT, cloud computing and other information technologies is reaching wider and wider areas of applications. One of the most recent and emerging applications is smart environments and ambient assisted living (AAL). Smart environment is by definition *a small world where different kinds of smart device are continuously working to make inhabitants' lives more comfortable.* [21, p.3]. AAL systems are *developed for personalised, adaptive, and anticipatory requirements, necessitating high quality-of-*

service to achieve interoperability, usability, security, and accuracy. [22, p. 4312]. Various sensor network systems for assisted living have been proposed, e.g. AlarmNet [23] and iSenior [24].
In assisted living scenario, health data is important both for the patient and the health institution managing the assisted living service. First of all, telemonitoring has its challenges. In assisted living scenario, the patient wears body sensors that measure, for example, the heart rate and the blood pressure. These sensors are connected to Internet to send the data from the sensors to further analysis and storage. This kind of network is called body area network (BAN). BANs and their challenges are discussed in more details e.g. in [25]. Another important aspect is that the health data is securely evaluated. This requires a protocol that does not compromise the data integrity and privacy. Several protocols have been proposed to achieve this goal. For example, Brickell et al. [26] use branching programs as the model for diagnostics tool. Lin et al. have improved the protocol: some of the decryption related computation load is removed to the cloud [27].

2.2 Privacy preservation techniques

The data transfer channels are rarely inherently secure. These channels are often targets for passive and active attacks. The attacks aim at getting information which would not be available to the attacker through secure channels.

The cloud environment also has its security threats. Someone can attack the cloud and steal information from it. The cloud may also compromise the privacy, security or integrity of data purposely. To solve these issues, different cryptographic tools have been proposed.

One possible way to solve privacy issues is to utilize techniques for *secure multi-party computation*. Several multiparty protocols have been designed for privacy-preserving computation. For example, secure sum protocol has been applied in [8] to achieve a privacy-preserving electronic surveillance. In privacy-preserving health monitoring, garbled branching programs have been used [26, 27]. In this paper, we also use the garbling technique but instead of branching programs we use another computational model, logic circuits.

Garbled circuits are one of the first protocols for secure multiparty computation [28]. The technique of garbling has recently been formalised by Bellare et al. who proposed a concept of *a garbling scheme*. Along the definition, Bellare et al. proposed also several security concepts for garbling schemes [29,30]. Next, we provide the definition of a garbling scheme and its security. We follow the definitions of [31].

A garbling scheme consists of six algorithms $(KeyGen, Ga, En, Ev, De, ev)$. The algorithm ev computes the original function f on an argument x and returns the final value $y = f(x)$. We do not want to reveal f or x to the evaluator so the evaluator cannot use algorithm ev to directly compute $f(x)$. The evaluator performs *a garbled evaluation*. The protocol starts with key generation: the algorithm $KeyGen$ is used for generating three keys (g, e, d). The first key is the garbling key which is used for garbling the function f with the function garbling algorithm Ga. Algorithm Ga returns the *garbled function* F based on g *and* f. The encryption algorithm En is called with key e and argument x to generate the garbled argument X. Both garbled argument X and garbled function F are used for computing the garbled evaluation result $Y = Ev(F, X)$. Finally the garbled evaluation result Y is decrypted into the final function value $y = De(d, Y) = De(d, Ev(F, X))$. It is worth noting that the result of evaluation must be the same despite the way of evaluation – the garbled evaluation $y = De(d, Ev(F, X))$ must give the same result as the direct evaluation $y = ev(f, x)$. The working of a garbling scheme is also illustrated in Figure 1 below.

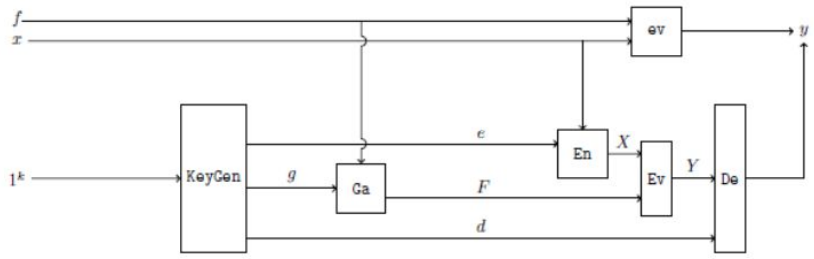


Figure 1 Garbling scheme and how its six algorithms work. The figure is adapted from [31].

According to Meskanen et al. [7], the security of garbling schemes can be characterised by two dimensions. The first dimension is the security notion which tells what information is hidden during the garbled evaluation. There are in total three different security notions: *function and argument hiding, function, argument and final value hiding* and *matchablility-only*. A garbling scheme which is *function and argument hiding* does not reveal the function or the argument to the evaluator, but the final value is allowed to be leaked. A garbling scheme which is *function, argument and value hiding* does not allow any of f, x, y to be leaked. The third notion describes a garbling scheme which hides f and x, but when evaluating $y_1 = De(d, Ev(F, X_1))$ and $y_2 = De(d, Ev(F, X_2))$, the equality of y_1 and y_2 can be leaked (but not the values y_1 or y_2).

The three security notions described above deal with secrecy. The fourth security notion for garbling schemes is *authenticity*. The authenticity of a garbling scheme is defined in the following way. If an adversary is unable to create a garbled value Y from a garbled function F and the garbled argument X such that $Y \neq Ev(F, X)$ but which would be considered as an authentic garbled value for some other F' and X', then the garbling scheme is said to achieve authenticity. For a formal definition of authenticity for garbling schemes, consult [29].

The second dimension for garbling schemes is *the level of reusability*: a parameter illustrating how many times the same garbled function can be securely used for different arguments. Some garbling schemes enable one-time use of the same garbled function. This corresponds to the first reusability level [29, 32]. Reusable garbling schemes allow the same garbled function to be used with several garbled arguments and garbling schemes allowing this have a higher reusability level [33, 34].

3. Scenario

This section is a description of the scenario to which we apply our privacy-preserving holistic security assessment procedure. The client in the scenario is an elderly person living alone. This elderly person needs occasional assistance in his daily routines. The routines may include going into shower or preparing meals. An assistant visits the elderly person at agreed points in time and records the health of the patient at each visit. When the assistant is not at the client, the health and the overall security is surveyed with different sensors and monitors. The health sensors may include appliances like a physical activity wristband, a heart rate monitor, an electronic sphygmomanometer and a pill count sensor in a pill dispenser (see e.g. [35,36]). Other sensors and monitors for assessing the overall security may include motion detectors, smoke detectors and closed circuit television cameras.

Another party in this scenario is the assisted living service provider who co-operates with a security company to guarantee the overall security of the client. The security company is responsible for the physical security of the client and his property, including protection against intruders, fire and theft. The security company is prepared for emergencies day and night. If the security analysis result yields

a less urgent alarm, the security company forwards the security analysis result to the assisted living service provider who is responsible for the daily care of the client.

The system at the client's home collects data for the analysis of both companies day and night. The monitor data is processed by an assessment algorithm that checks whether the vital information, such as heart rate and blood pressure, is within acceptable ranges and that no other security threats, e.g. a burglar or a fire, have been detected. The assessment algorithm consists of two parts. The first part is the health assessment algorithm f_1 which is owned by the assisted living service provider. The second part is the surveillance algorithm f_2 owned by the security company. The assisted living service provider authorizes the security company to garble the health assessment algorithm together with the security algorithm and send the garbled algorithms to the cloud.

We assume that the security company has outsourced its data services to a third party cloud. This includes storing and processing monitored data in the cloud. We do not assume the cloud to be specific to this kind of scenario. In other words, we do not assume that the cloud is secure or even trustworthy – our solution works with insecure and untrustworthy clouds as well.

If the monitored data looks abnormal with respect to earlier data or known recommendations, then the assessment algorithm triggers an alarm. The company to whom the alarm concerns is allowed to access the raw monitored data in a case of alarm has been the assessment result returned by the cloud. Otherwise, the raw data is kept secret from both companies. The reason for this arrangement is that in this way the client feels more confident about his privacy since only abnormal situations reveal information to outsiders.

3.1 Requirements for the implementation

There are several requirements for the security monitoring system which must be taken into account in the implementation. Below is a brief description of these requirements.

Integrity: The monitored surveillance data should be authentic for reliable security assessment. We may assume that the client is honest and therefore the security data is authentic. Cloud can be honest, semi-honest or even malicious – a garbling scheme achieving certain level of security (explained in Section 2.2) guarantees that the analysis result is also authentic. Additionally, integrity of data in transit is protected, e.g. by using message authentication codes.

Entity authentication: The cloud does not need to authenticate itself to the client or to the assisted living service provider, since all the data processed by the cloud is encrypted. The client and the security company need to authenticate themselves mutually. This is performed when the security monitoring system is first configured. We assume that after the mutual authentication, the channel between the client and the security company is confidential and authentic.

Access control: Any party outside the client, the security company and the assisted living service provider should never have access to the unencrypted security data, including the third party cloud. The security company should not be able to access the unencrypted security data unless the security assessment algorithm results in an emergent alarm. The assisted living company can access the raw data in the case of a non-emergent alarm. This requires that the security assessment algorithm must not reveal the processed security data. To ascertain this, a trusted auditor is used to verify appropriateness of the security assessment tool. Legal ways for the security company to access the raw data are discussed in more details in section 3.2.

Authorization: The client has to authorize the assisted living service provider to have access to raw data in case of alarm. The assisted living service provider authorizes the security company to garble

the health assessment algorithm. The security company provides authorization for the cloud provider in order to receive garbled data from the client.

Non-repudiation: Non-repudiation between the client, the security company and the assisted living service provider is guaranteed by the following facts. These three parties have authenticated themselves mutually at the configuration phase, after which the communication between the parties occurs through a confidential and authentic channel. We assume that the garbled data sent by the client is authentic and available. The implementation of the garbling scheme guarantees authenticity of the garbled evaluation. Moreover, log data collected by all parties can be used for non-repudiation purposes (the logs can be cross-compared with each other).

Availability: It is essential that the security data is available to the security company as well as the assisted living service provider in alarming situations. However, there are several implementation threats related to the availability. The monitors and sensors might not share any data (equipment is switched off by the client, battery is low etc.). The Internet connection between the cloud and the client may be interrupted for various reasons. The data from the monitors and sensors may also be compromised – improper measurements and purposefully distracting the measurements affect to the availability but also integrity of the security data. These availability issues may be solved by frequent checking and maintaining of the system.

3.2 Description of the protocol for security monitoring system

In this section, we describe the protocol which is used by the four parties, client, assisted living service provider, the security company and the cloud, to achieve privacy-preserving security monitoring. The central building block in our solution is a function, argument and value hiding garbling scheme that also achieves authenticity. The protocol is illustrated in Figure 2, whereas the use of a garbling scheme in the protocol is illustrated in Figure 3.

The protocol starts first at client side. The three keys related to the garbling are first generated. The client shares the function garbling key g and the ungarbling key d with the assisted living service provider and keeps the argument garbling key e. Then, the client garbles the security data x and sends the garbled security data $X = En(e,x)$ to the cloud. Meanwhile, the security company garbles the combined security assessment algorithm $f = (f_1, f_2)$ and sends the garbled function $F = Ga(g,f)$ to the cloud. Here, f_1 represents the health assessment algorithm obtained from the assisted living service provider and f_2 the surveillance analysis algorithm owned by the security company.

Once the cloud has received both the garbled health data and the garbled function, then the cloud performs the garbled evaluation. The cloud sends the result $Y = Ev(F,X)$ of the garbled evaluation to the security company who ungarbles Y. The final value y then reveals, whether there is anything alarming at client side or not.

Every time the client prepares garbled security data to the cloud, it also prepares encrypted security data X' which can be accessed only when the security assessment algorithm results in an alarm. To encrypt x to X' and to decrypt X' back to x, the client, the security company and the assisted living service provider agree to use an encryption scheme $\mathcal{E} = (Keygen', En', De')$ which is independent from the corresponding algorithms used in garbling.

To guarantee that the raw data is accessed only in alarming situations, access to the decryption key d' must be realised accordingly. This can be realised in different manners, as Meskanen et al. show in [7]. Here, we use a variant of the solution in which the decryption key d' is a part of the argument to the security assessment algorithm, and d' is revealed in the final result only when the assessment algorithm returns an alarm.

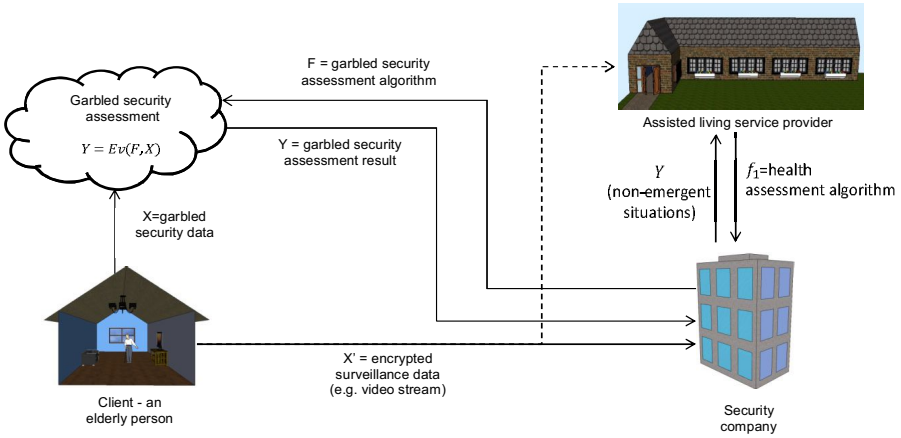


Figure 2 Security service scenario for an elderly person. In the case of an alarm, only the related department is authorised access to the raw security data.

In our variant, the decryption key d' is encrypted with a further independent encryption scheme $\widehat{\mathcal{E}} = (\widehat{Keygen}, \widehat{En}, \widehat{De})$ using two keys key_1 and key_2. The decryption key d' is encrypted with key_1in the case of an emergency. Only the security company owns the key_1, so in the case of an emergency only the security company is able to access the raw surveillance data. In the case of a non-emergent alarm, the decryption key d', key_2 is used for encrypting d'. The key key_2 is only owned by the assisted living service provider, which guarantees that only the assisted living service provider can access the raw surveillance data in a non-emergent alarm. If there is no alarm, neither company gets the decryption key d'.

More precisely, let us denote by f the actual security assessment algorithm and by x the health data. We define now

$$x = (x_{actual}, d') \tag{1}$$

$$f(x) = \begin{cases} (no\ alarm, \varepsilon) \\ (emergency, \widehat{En_{key1}}(d')) \\ (non-emergent\ alert, \widehat{En_{key2}}(d')) \end{cases} \tag{2}$$

In (1), the value x_{actual} represents the actual raw data collected by the surveillance system. In (2), the final result depends on the type of the alarm. If there is no alarm, neither the security company nor the assisted living service provider gets access to the raw surveillance data. If there is an emergency, only the security company gets access to the raw surveillance data. If there is a non-emergent alert, only the assisted living service provider gets access to the raw surveillance data.

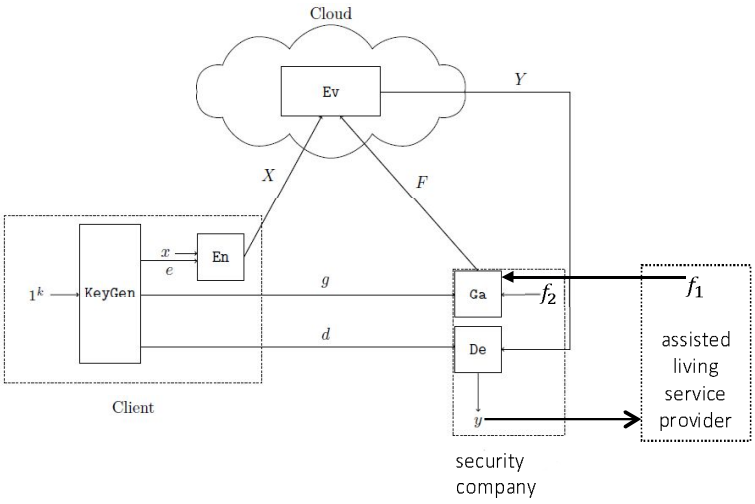


Figure 3 The use of a garbling scheme in security monitoring for assisted living

3.3 Security and privacy considerations

The integrity of the assessment result is guaranteed if the garbling scheme used in the protocol achieves authenticity. Moreover, other techniques like message authentication codes guarantee the data integrity in other channels as well.

Secondly, access to the health data is as desired. The choice of function f and the argument x as described in Section 3.1 guarantee that the assisted living service provider is allowed to access the raw health data only when the assessment result is an alarm. Moreover, garbling the health data prevents the cloud and other parties from accessing the raw health data, as desired. Furthermore, the assessment algorithm is kept private, because the garbling scheme is also function hiding.

Non-repudiation is partly guaranteed by the fact that we require from the garbling scheme that it achieves authenticity. This applies to the final assessment result.Event logs may be used as a countermeasure. All three parties maintain their independent logs which can be compared to detect unauthorised or illegitimate access to the backup data.

Entity authentication, availability and authorisation are not guaranteed by the use of garbling scheme. To achieve these security measures, different additional solutions are needed.

4. Implementation

We measure the efficiency of garbling by designing and implementing a simple but still practical health assessment algorithm. The system suggested in this paper is more than just a health assessment system. However, a simple but still practical test case provides valuable information about the running time of our garbling-based solution.

Our implementation uses logic circuits as the model of computation. The implementation of the garbling scheme fulfills function and argument hiding property. Our implementation optimizes JustGarble software package [37] for the needs of remote health assessment.

4.1 Health assessment algorithm

The system in our test case measures physical activity, heart rate, diastolic/systolic blood pressure and whether medicines have been taken. A wristband can be used for measuring physical activity and heart rate of the client. Blood pressure can be measured by an electronic sphygmomanometer. To determine whether the medicines have been taken a sensor is attached to the pill dispenser. All different devices are connected to the wireless network at client's home.

Our health assessment algorithm takes an argument consisting of 28 bits. Physical activity and medication taken are expressed using one bit of information (yes / no) whereas heart beat rate and the two blood pressure values are expressed using unsigned 8 bit representation. This means that the range for pulse, diastolic and systolic pressure is 0 - 255.

It is possible that some of the measurements cannot be updated for some reasons (the client has forgotten to do the measurements, the equipment is malfunctioning etc.). In this case, there are predefined thresholds for times that the system waits for input to the health assessment algorithm. If this threshold is crossed, the system changes the value to all zero value, which will trigger an alarm.

In normal functioning of the system, alarms are triggered in the following situations. The algorithm checks first whether the client has physical activity. If the client does not have physical activity, then the heart beat rate must be between 60 to 100 beats per minute. If the rate does not lie in this range, then an alarm is triggered. If the client has physical activity then the heart beat rate should be between 75 and 145. Otherwise, an alarm is triggered. The algorithm also checks blood pressure readings: the diastolic blood pressure should be between 60 and 90, the systolic blood pressure between 90 and 140. An alarm is triggered if either the diastolic or the systolic pressure is too low or too high. Finally, the algorithm checks whether the client has remembered to take the medicine. If all the values were at acceptable range, no alarm is triggered. The system stores the previous state and awaits the next argument to be processed. The time depends on how often the medical information is configured to be updated.

4.2 Performance evaluation

The function described above has been transformed into a logic circuit that consists of 545 gates including of 89 NOT gates, 302 XOR gates, 14 AND gates and 14 OR gates. We have first run the circuit on 10 random inputs to test the speed of the evaluation. We take the median value of those ten evaluations. The measurement has been performed on a laptop having Intel i5-4210U processor and Ubuntu 14.04 LTS as the operating system with Gnu gcc compiler 4.8.2. The clock rate of the laptop is 2.40GHz. The times are given in clock cycles which also have been converted to microseconds. The results can be found from table I. We have also tested the correctness of our solution by using realistic values for physical activity, heart beat rate, diastolic and systolic blood pressure as well as for medicines taken.

Table 1 Garbling and evaluation times for the health assessment algorithm

	Time to garble	Time to evaluate	Total
cycles	66509.8	32954.0	99463.8
microseconds	27.7	13.7	41.44

Table 1 shows that garbling and evaluating our health assessment algorithm is extremely fast, only 41.44 microseconds per one garbled evaluation round. One of the main reasons explaining the high speed is that the algorithm is quite simple even though it models a practical health assessment algorithm based on literature (see e.g. [27]). Moreover, our algorithm does not explicitly keep track of the previous inputs. The algorithm should keep track of e.g. the relation between physical activity and heart beat rate. One abnormal heart beat rate in physical activity may be temporal and acceptable. The alarm is triggered only if the same too high heart beat rate remains the same for e.g. 5 assessment cycles. This property would significantly reduce false positive alarms.

Table 2 Garbling and evaluation times of a random circuit

Number of gates (in thousands)	Time to garble (µs)	Time to evaluate (µs)	Total time (µs)
1	0.34	1.79	2.13
5	1.31	5.40	6.71
10	2.46	6.15	8.61
100	6.32	55.87	62.19
200	12.86	109.18	122.04
400	25.43	217.01	242.44

We also tested more complex circuits using our optimized version of JustGarble. In this test, the input consists of 1024 bits and the output of 1 bit. In table 2 we have collected the data by increasing the complexity of the circuit. To compare, a circuit consisting of 15 500 000 gates is garbled in 0.49 seconds and evaluated in 0.23 seconds, which makes one evaluation cycle to last 0.72 seconds [7]. From response time point-of-view, all these times are still reasonably fast.

There are some additional aspects that increase the running time above the times presented in tables 1 and 2. In our system, the recovery key d' is encrypted when it is sent to the cloud. The recovery key d' must also be decrypted at the company side, which slows down the response time. In addition, transferring the garbled data to the cloud takes some time, as well as transferring the garbled assessment results from the cloud to the assisted living service provider. Finding estimates for these times is not easy. However, the time used for the garbled evaluation of the security assessment algorithm is extremely fast, so our approach suits also situations where emergency response times need to be extremely short.

5. Conclusion

Designing and implementing privacy-preserving services is a real challenge. Especially services including surveillance or processing private data are examples where privacy-preserving property is needed but challenging to realise efficiently.

In this paper we propose a privacy-preserving security system including both health monitoring and other security services. Our solution is based on garbling schemes. The function and argument hiding property of garbling schemes guarantee privacy preservation in many contexts, e.g. in electronic surveillance or in health monitoring. We have also demonstrated the practicality of our solution. We have implemented a simplistic health assessment function. Moreover, we have tested the efficiency of garbling and evaluation of this function. The results show garbling and evaluation of even complex circuits is efficient.

Acknowledgments

This work was supported by the Finnish Academy project "Cloud Security Services" which is greatly appreciated.

References

[1] R. Roman, P. Najera, and J. Lopez. *Securing the Internet of Things.* Computer, 44(9):51–58, Sept 2011.
[2] T. Heer, O. Garcia-Morchon, R. Hummen, S. L. Keoh, S. S. Kumar, and K. Wehrle. *Security Challenges in the IP-based Internet of Things.* Wirel. Pers. Commun., 61(3):527–542, 2011.
[3] O. Vermesan, M. Harrison, H. Vogt, K. Kalaboukas, M. Tomasella, K. Wouters, S. Gusmeroli, and S. Haller. *Vision and Challenges for Realising the Internet of Things.* European Commission, Information Society and Media, 2010.
[4] D. Kozlov, J. Veijalainen, and Y. Ali. *Security and Privacy Threats in IoT Architectures.* In Proceedings of the 7th International Conference on Body Area Networks, BodyNets '12, pages 256–262, ICST, Brussels, Belgium, Belgium, 2012. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
[5] J. S. Winter. *Surveillance in Ubiquitous Network Societies: Normative Conflicts Related to the Consumer In-store Supermarket Experience in the Context of the Internet of Things.* Ethics and Inf. Technol., 16(1):27–41, 2014.
[6] K. B. Frikken and M. J. Atallah. *Privacy-preserving Electronic Surveillance.* In Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society, WPES '03, pages 45–52, New York, NY, USA, 2003. ACM.
[7] T. Meskanen, V. Niemi, and N. Nieminen. *How to use garbling for privacy-preserving electronic surveillance services.* Journal of Cyber Security and Mobility, 2015. To appear.
[8] V. Oleshchuk. *Internet of things and privacy-preserving technologies.* In 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology, 2009. Wireless VI-TAE 2009., pages 336–340, 2009.
[9] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. *Tools for Privacy-preserving Distributed Data Mining.* SIGKDD Explor. Newsl., 4(2):28–34, Dec. 2002.
[10] H. Abie and I. Balasingham. *Risk-based Adaptive Security for Smart IoT in eHealth.* In Proceedings of the 7th International Conference on Body Area Networks, BodyNets '12, pages 269–275, ICST, Brussels, Belgium, Belgium, 2012. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
[11] H. Löhr, A.-R. Sadeghi, and M. Winandy. *Securing the e-Health Cloud.* In Proceedings of the 1st ACM International Health Informatics Symposium, IHI '10, pages 220–229. ACM, 2010.
[12] P. M. Trief, J. Sandberg, R. Izquierdo, P. C. Morin, S. Shea, R. Brittain, E. B. Feldhousen, and R. S. Weinstock. *Diabetes Management Assisted by Telemedicine: Patient Perspectives.* Telemedicine and e-Health, 14(7):647–655, 2008.

[13] M. Layouni, K. Verslype, M. T. Sandıkkaya, B. De Decker, and H. Vangheluwe. *Privacy-Preserving Telemonitoring for eHealth.* In E. Gudes and J. Vaidya, editors, Data and Applications Security XXIII, volume 5645 of Lecture Notes in Computer Science, pages 95–110. Springer Berlin Heidelberg, 2009.
[14] M. Barni, J. Guajardo, and R. Lazzeretti. *Privacy-preserving evaluation of signal quality with application to ECG analysis.* In Information Forensics and Security (WIFS), 2010 IEEE International Workshop on, pages 1–6, Dec 2010.
[15] N. Karthikeyan and R. Sukanesh. *Cloud Based Emergency Health Care Information Service in India.* Journal of Medical Systems, 36(6):4031–4036, 2012.
[16] M. Poulymenopoulou, F. Malamateniou, and G. Vassilacopoulos. *Emergency Healthcare Process Automation Using Mobile Computing and Cloud Services.* J. Med. Syst., 36(5):3233–3241, Oct. 2012.
[17] M. Nkosi and F. Mekuria. *Cloud Computing for Enhanced Mobile Health Applications.* In Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, pages 629–633, Nov 2010.
[18] C. Doukas, T. Pliakas, and I. Maglogiannis. *Mobile health-care information management utilizing Cloud Computing and Android OS.* In Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE, pages 1037–1040, Aug 2010.
[19] D. Hoang and L. Chen. *Mobile Cloud for Assistive Healthcare (MoCAsH).* In Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific, pages 325–332, Dec 2010.
[20] Y. Tong, J. Sun, S. Chow, and P. Li. *Cloud-Assisted Mobile-Access of Health Data With Privacy and Auditability.* Biomedical and Health Informatics, IEEE Journal of, 18(2):419–429, March 2014.
[21] D. Cook and S. Das. *Smart Environments: Technology, Protocols and Applications* (Wiley Series on Parallel and Distributed Computing). Wiley-Interscience, 2004.
[22] M. Memon, S. R. W. C. F. Pedersen, F. H. A. Beevi, and F. O. Hansen. *Ambient Assisted Living Healthcare Frameworks, Platforms, Standards, and Quality Attributes.* Sensors, 14(3):4312– 4341, 2014.
[23] A. Wood, J. A. Stankovic, G. Virone, L. Selavo, Z. He, Q. Cao, T. Doan, Y. Wu, L. Fang, and R. Stoleru. *Context-aware wireless sensor networks for assisted living and residential monitoring.* Network, IEEE, 22(4):26–33, 2008.
[24] A. Rodrigues, J. S. Silva, and F. Boavida. *iSenior – A Support System for Elderly Citizens.* Emerging Topics in Computing, IEEE Transactions on, 1(2):207–217, 2013.
[25] G. Fortino, G. Di Fatta, M. Pathan, and A. Vasilakos. *Cloud-assisted body area networks: state-of-the-art and future challenges.* Wireless Networks, 20(7):1925–1938, 2014.
[26] J. Brickell, D. E. Porter, V. Shmatikov, and E. Witchel. *Privacy-preserving Remote Diagnostics.* In Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS '07, pages 498–507. ACM, 2007.
[27] H. Lin, J. Shao, C. Zhang, and Y. Fang. CAM: *Cloud-Assisted Privacy-preserving Mobile Health Monitoring.* Information Forensics and Security, IEEE Transactions on, 8(6):985–997, June 2013.
[28] A. Yao. How to generate and exchange secrets. In Proc. of 27th FOCS, 1986., pages 162–167. IEEE, 1986.
[29] M. Bellare, V. T. Hoang, and P. Rogaway. Foundations of Garbled Circuits. In Proc. of ACM Computer and Communications Security (CCS'12), pages 784–796. ACM, 2012.
[30] M. Bellare, V. T. Hoang, and P. Rogaway. *Adaptively secure garbling scheme with applications to one-time programs and secure outsourcing.* In Proc. of Asiacrypt 2012, volume 7685 of LNCS, pages 134–153. Springer, 2012.
[31] T. Meskanen, V. Niemi, and N. Nieminen. *Garbling in Reverse Order.* In The 13th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-14), 2014.
[32] T. Meskanen, V. Niemi, and N. Nieminen. *Classes of Garbled Schemes.* Infocommunications Journal, V(3):8–16, 2013.
[33] C. Gentry, S. Halevi, S. Lu, R. Ostrovsky, M. Raykova, and D. Wichs. Garbled RAM Revisited. In *Proc. of* 33*rd Eurocrypt*, volume 8441 of LNCS, pages 405–422, 2014.
[34] T. Meskanen, V. Niemi, and N. Nieminen. *Hierarchy for Classes of Garbling Schemes.* In Proc. of Central European Conference on Cryptology (CECC'14), 2014.
[35] S. Coughlin, M. Coughlin, S. Orr, and T. Surgeon. *Pill count sensor for automatic medicament dispensing machine,* July 15 2003. US Patent 6,592,005.
[36] N. Varvarelis and J. Samuelson. *Electronic pill dispenser,* Apr. 15 2008. US Patent 7,359,765.
[37] JustGarble. http://cseweb.ucsd.edu/groups/justgarble/. Accessed: 2014-10-13.

Part III

Omitted proofs in original publications

Chapter 6

Omitted proofs

In original publications, proofs of some theorems have been omitted. This chapter contains the omitted proofs of these theorems. These proofs have been left out from the papers due to strict page limits imposed by the publication venues. However, the proofs are included in this work for completeness. Sections in this chapter contain only the proofs, the theorems are not repeated since they can be found in the original publications.

6.1 Original publication III

Proof of Theorem 2: For the security notions of privacy and matchability-only, we prove the claim only under the following assumption $(*)$. For obliviousness, this additional assumption is not needed.

Assumption $(*)$ We assume that there exist two functions f_0, f_1 and two arguments $x_0 \neq x_1$ such that $\Phi(f_0) = \Phi(f_1)$ and $\mathtt{ev}(f_0, x_0) = \mathtt{ev}(f_1, x_1)$.

For example, if there exists a function f such that the evaluation of f, i.e. $\mathtt{ev}(f, \cdot)$, is not injective then the assumption $(*)$ holds.

There are pathological cases where the assumption $(*)$ does not hold. For instance, if both Φ and $\mathtt{ev}$ are identity functions then all garbling schemes are prv.yyy.padap$_\ell$ and mao.yyy.padap$_\ell$ secure. In this case, even a trivial garbling scheme that uses identity functions for garbling is secure (for indistinguishability).

The same assumption $(*)$ is needed when proving the following theorems: Theorem 1 and Theorem 2 in publication II, Theorem 1, Theorem 2 and Theorem 4 in publication III, Theorem 3 and Theorem 4 in publication V and Theorem 5 in publication VI. The assumption is needed for the privacy and matchability-only notions whereas for the obliviousness notion the results hold without this additional assumption. If we did not make the assumption $(*)$ for privacy and matchability-only security classes, then

in some situations all garbling schemes would be secure and the hierarchy would collapse. On the other hand, all inclusions are proper for garbling schemes achieving obliviousness, even in the case of pathological evaluation algorithms.

The inclusion $\mathcal{F}_* \subseteq \mathcal{F}_\ell$ clearly holds for any $\ell \in \mathbb{N}$, so we have the inclusion

$$\mathcal{F}_* \subseteq \bigcap_{\ell=1}^{\infty} \mathcal{F}_\ell.$$

If $\bigcap_{\ell=1}^{\infty} \mathcal{F}_\ell = \emptyset$ holds then we must have that $\mathcal{F}_* = \emptyset$ since $\mathcal{F}_*$ is a subset of $\bigcap_{\ell=1}^{\infty} \mathcal{F}_\ell = \emptyset$.

Let us assume that $\bigcap_{\ell=1}^{\infty} \mathcal{F}_\ell$ is non-empty. Our aim is to prove that there is a garbling scheme in $\bigcap_{\ell=1}^{\infty} \mathcal{F}_\ell$ which does not belong to $\mathcal{F}_*$.

Let $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{Ev}, \mathtt{De}, \mathtt{ev})$ be a garbling scheme in $\bigcap_{\ell=1}^{\infty} \mathcal{F}_\ell$. We construct a garbling scheme $\mathcal{G}' = (\mathtt{Gb}', \mathtt{En}', \mathtt{Ev}', \mathtt{De}, \mathtt{ev})$ by modifying the algorithms $\mathtt{Gb}$, $\mathtt{En}$ and $\mathtt{Ev}$ of garbling scheme $\mathcal{G}$ as follows.

The modified encrypting algorithm $\mathtt{En}'$ consists of three parts. The first part contains the original encrypting algorithm $\mathtt{En}$ from garbling scheme $\mathcal{G}$. The second part is a completely independent, symmetric, non-deterministic, secure encryption scheme $\widehat{\mathtt{En}}$ that is used to encrypt x_i with an independent key which will be denoted by $\widehat{e}$. This independent encryption key $\widehat{e}$ is generated by the garbling algorithm $\mathtt{Gb}'$. For the third part, we use a secret sharing scheme which is also constructed by the garbling algorithm $\mathtt{Gb}'$. The secret is the encryption key $\widehat{e}$ which is divided into t shares, where any subset of k shares of the secret is sufficient to reconstruct the key $\widehat{e}$. Here, k represents the security parameter in the garbling scheme $\mathcal{G}$.

The garbled argument bit X_i' is obtained with algorithm $\mathtt{En}'$ as follows: $\mathtt{En}'((e, \widehat{e}, t \text{ shares}), x_i) = (\mathtt{En}(e, x_i), \widehat{\mathtt{En}}(\widehat{e}, x_i), s) = (X_i, \widehat{X_i}, s)$ where s is a random share of $\widehat{e}$.

The garbling algorithm $\mathtt{Gb}'$ creates (F', e', d') by using algorithm $\mathtt{Gb}$ to generate (F, e, d). In addition, it has to create $\widehat{e}$ by taking the security parameter k into account (the encryption with $\widehat{e}$ cannot be breakable in polynomial time with respect to k). Moreover, $\mathtt{Gb}'$ creates t shares of $\widehat{e}$. The output of algorithm $\mathtt{Gb}'$ is (F', e', d') where $F' = ((F, \widehat{F}), e' = (e, \widehat{e}, \text{t shares of key } \widehat{e})$ and $d' = d$. The components of F' consist of the garbled function F (generated by $\mathtt{Gb}$) and encrypted function $\widehat{F} = \widehat{\mathtt{En}}(\widehat{e}, f)$.

The garbled evaluation algorithm $\mathtt{Ev}'$ takes $((F, \widehat{F}), (X, \widehat{X}, s))$ as its inputs. Here $X = X_1 \dots X_n$ is the fully specified garbled input which is obtained by using $\mathtt{En}$ of garbling scheme $\mathcal{G}$ whereas $\widehat{X} = \widehat{X_1} \dots \widehat{X_n}$ is the fully specified argument encrypted with the independent key $\widehat{e}$. Finally, $S = \{s_1, \dots s_n\}$ is the set of n random shares of secret $\widehat{e}$. The algorithm omits the new parts $\widehat{F}, \widehat{X}, s$ and computes $Y = \mathtt{Ev}'((F, \widehat{F}), (X, \widehat{X}, s)) = \mathtt{Ev}(F, X)$.

First we prove that $\mathcal{G}' \in \bigcap_{\ell=1}^{\infty} \mathcal{F}_\ell$. To do that, we first prove that $\mathcal{G}' \in \mathcal{F}_\ell$ for all $\ell \in \mathbb{N}$ from which the claim follows. Consider an arbitrary adversary playing the game related to class $\mathcal{F}_\ell$. The win probability of the adversary now depends on the parameters ℓ and k, because these two variables determine whether the adversary could have enough shares to reconstruct the second encryption key $\widehat{e}$. If $k > n \cdot \ell$ then the adversary does not have enough shares to reconstruct $\widehat{e}$, and the advantage in the game will be the same as the adversary would have with respect to garbling scheme $\mathcal{G}$, i.e. the advantage is negligible.

Finally, consider the claim $\mathcal{G}' \notin \mathcal{F}_*$. An adversary playing the game related to $\mathcal{F}_*$ is allowed to call `INPUT` procedure arbitrarily many times, especially more than k times. If the adversary calls `INPUT` procedure k^2 times then the well-known approximations related to birthday paradox imply that there is a non-zero probability, not depending on k, to get all the shares. Consequently, the adversary has a non-zero probability, not depending on k, to win the game. Let us now show why this is the case.

Here we need the assumption $(*)$ mentioned in the beginning. Shares of the encryption key $\widehat{e}$ do not help unless there are at least two different possible inputs. These two can then be told apart once all shares are available.

When the adversary plays the indistinguishability-based security game, the adversary chooses f_0, f_1, x_0 and x_1 such that the properties of assumption $(*)$ are satisfied. All the tests in the game are passed with these inputs and the adversary gets $(F, \widehat{F})$ and $(X, \widehat{X})$. Since there are enough shares to recover the encryption key $\widehat{e}$, the adversary can decrypt $\widehat{F}$ and $\widehat{X}$. The decryptions of $\widehat{F}$ and $\widehat{X}$ now reveal which of the functions and which of the corresponding arguments have been used in the garbling. In this case, the adversary is always able to win the game.

We have assumed that there are elements f_0, f_1, x_0 and x_1 which satisfy the properties of assumption $(*)$. When the adversary plays a simulation-based security game, he chooses randomly one of the functions f_0, f_1 and the corresponding argument, x_0 or x_1. It follows that the simulator in the game does not have a chance to distinguish which of (f_0, x_0) or (f_1, x_1) has been chosen by the adversary. The simulator outputs $(F, \widehat{F})$ and $(X, \widehat{X})$ in usual way. Since there are enough shares to recover the encryption key $\widehat{e}$, the adversary can decrypt $\widehat{F}$ and $\widehat{X}$. Now, there is at least 50% chance that the decryptions of $\widehat{F}$ and $\widehat{X}$ are not representing the function and the argument chosen by the adversary. In this case, the adversary is able to always win the game.

In both cases, being able to decrypt $\widehat{F}$ and $\widehat{X}$ leads to ability to win the game. This happens with a positive probability, not depending on the security parameter k. This leads to non-negligible advantage of the adversary,

implying that the garbling scheme $\mathcal{G}'$ does not belong to $\mathcal{F}_*$. This completes the proof. □

Proof of Theorem 3: Let us first consider the inclusion $\mathcal{F}_\ell \subseteq \mathcal{C}_\ell$. Let $\mathcal{G}$ be a garbling scheme in class $\mathcal{F}_\ell$. We prove that $\mathcal{G}$ belongs also to class $\mathcal{C}_\ell$.

Let $\mathcal{A}$ be an arbitrary adversary playing the security game against garbling scheme $\mathcal{G}$ in class $\mathcal{C}_\ell$. We construct another adversary $\mathcal{A}'$ playing the corresponding security game related to class $\mathcal{F}_\ell$. Adversary $\mathcal{A}'$ uses $\mathcal{A}$ as a subroutine as follows. The game related to class $\mathcal{F}_\ell$ starts with `INITIALIZE`, after which $\mathcal{A}'$ should give its input to `GARBLE` procedure. Instead, it tells adversary $\mathcal{A}$ to start a game related to class $\mathcal{C}_\ell$. Now, adversary $\mathcal{A}$ presumes to play the actual security game related to $\mathcal{C}_\ell$, but the game is actually emulated by $\mathcal{A}'$. Adversary $\mathcal{A}$ sends its input for `GARBLE` procedure (of $\mathcal{C}_\ell$ game) to $\mathcal{A}'$. $\mathcal{A}'$ sends the input from $\mathcal{A}$ to its `GARBLE` in $\mathcal{F}_\ell$ game and gets an output. $\mathcal{A}'$ sends this output to $\mathcal{A}$ who now proceeds to `INPUT` procedure. $\mathcal{A}$ starts sending its arguments, where all the m bits are fully specified, to $\mathcal{A}'$. Every time $\mathcal{A}'$ gets a fully specified garbled argument of length n, $\mathcal{A}'$ makes n queries to its `INPUT` by sending $\mathcal{A}'s$ argument bit by bit (the order in which the bits are sent does not matter). $\mathcal{A}'$ sends the fully specified garbled argument to $\mathcal{A}$ after getting all n garbled argument bits. Then $\mathcal{A}$ may make a new `INPUT` query or proceed to `FINALIZE`. $\mathcal{A}'$ does the same as $\mathcal{A}$ does. Because $\mathcal{A}$ is an adversary playing the game related to $\mathcal{C}_\ell$, $\mathcal{A}$ can make at most ℓ `INPUT` queries. Therefore, if $\mathcal{A}$ uses all ℓ allowed queries, $\mathcal{A}'$ must make $n \cdot \ell$ queries to `INPUT` in the game related to $\mathcal{F}_\ell$, and this is exactly the maximum number of allowed `INPUT` queries for adversary $\mathcal{A}'$ in $\mathcal{F}_\ell$ game.

Let us now consider the advantage of both adversaries. The answer of adversary $\mathcal{A}'$ to `FINALIZE` procedure in $\mathcal{F}_\ell$ game is exactly the same as the answer of adversary $\mathcal{A}$ in $\mathcal{C}_\ell$ game. Therefore, the probability that adversary $\mathcal{A}'$ wins its $\mathcal{C}_\ell$ game is the same as the probability that $\mathcal{A}$ wins its $\mathcal{F}_\ell$ game. This implies that the advantages of both adversaries are equal. According to our assumption, $\mathcal{G}$ belongs to security class $\mathcal{F}_\ell$ and because the advantage of any adversary playing the game related to class $\mathcal{F}_\ell$ is negligible, the advantage of $\mathcal{A}'$ is negligible. But $\mathcal{A}$ has the same advantage in his game which is now negligible. Moreover, we assumed $\mathcal{A}$ to be an arbitrary adversary which now implies that garbling scheme $\mathcal{G}$ belongs to security class $\mathcal{C}_\ell$ as well.

Next consider the claim $\mathcal{F}_\ell \neq \mathcal{C}_\ell$. We have to show that there is a garbling scheme in class $\mathcal{C}_\ell$ which does not belong to the class $\mathcal{F}_\ell$. The class $\mathcal{C}_\ell$ consists of all adaptively secure garbling schemes, including garbling schemes which do not have the projectivity property. On the other hand, all the garbling schemes in $\mathcal{F}_\ell$ are projective. Therefore, a non-projective, adaptively secure

garbling scheme belongs to $\mathcal{C}_\ell$ but not to $\mathcal{F}_\ell$, which proves the claim $\mathcal{F}_\ell \neq \mathcal{C}_\ell$. □

Proof of Theorem 4: Our aim is to prove that there exists a garbling scheme which belongs to class $\mathcal{F}_\ell$ but does not belong to class $\mathcal{C}_{\ell+1}$. We construct a garbling scheme $\mathcal{G}'$ starting from an arbitrary garbling scheme $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev}) \in \mathcal{F}_\ell$ by modifying algorithms `Gb`, `En` and `Ev`. The modifications to `Gb` and `Ev` are the same as the modifications made in proofs of Theorem 1 and Theorem 2 in this same publication III.

We modify the encryption algorithm `En` as follows. The modified algorithm $\mathtt{En}'$ takes $(e, \widehat{e}, x, \{s_1, s_2, \ldots, s_t\})$ as input and outputs $(X, \widehat{X}, s)$. Here, the key $\widehat{e}$ is an encryption key from an independent, secure, symmetric, nondeterministic encryption scheme $(\widehat{\mathtt{En}}, \widehat{\mathtt{De}})$. We encrypt the argument x with this independent key. The encrypted argument is denoted by $\widehat{X} = \widehat{\mathtt{En}}(\widehat{e}, x)$. The third component s is a randomly chosen share of the secret, which in this case is the encryption key $\widehat{e}$. The encryption keys e, $\widehat{e}$ and the shares of $\widehat{e}$ are generated by the garbling algorithm `Gb`.

The independent encryption scheme $(\widehat{\mathtt{En}}, \widehat{\mathtt{De}})$ is used to encrypt also the function f. The encryption of f with the independent key $\widehat{e}$ is denoted by $\widehat{\mathtt{En}}(\widehat{e}, f) = \widehat{F}$. The garbling algorithm $\mathtt{Gb}'$ first generates the keys e, $\widehat{e}$, d and the shares of $\widehat{e}$. Then it constructs the garbled function F and the encrypted function $\widehat{F}$. Depending on the security notion, the garbling algorithm $\mathtt{Gb}'$ returns $(F, \widehat{F}, d)$ (privacy) or $(F, \widehat{F})$ (matchability-only, obliviousness).

The encryption of the i^{th} input bit consists of the garbled argument bit X_i, the encrypted argument bit $\widehat{X_i} = \widehat{\mathtt{En}}(\widehat{e}, x_i||i||r)$ and a share chosen from the t shares of key $\widehat{e}$. We denote the garbled argument bit by $\overline{X_i} = (X_i, \widehat{X_i}, s)$. The choice of the share s is performed as follows. Let h be a hash function that takes two inputs, i and r, and returns an integer $j \in \{1, \ldots, t\}$. The t shares of $\widehat{e}$ are indexed from 1 to t, and the value of $h(i, r)$ tells which of the shares has been chosen. The number of shares needed to reconstruct $\widehat{e}$ is $n \cdot \ell + 1$.

Let us now consider the projectivity of the modified scheme $\mathcal{G}'$. Let $x = x_1 \ldots x_n$ and $x' = x'_1 \ldots x'_n$ be two bit strings of length n. Let $\overline{X}$ and $\overline{X'}$ be the garbled arguments for x and x' in scheme $\mathcal{G}'$. The arguments x and x' have been encrypted using the same randomness value r and the same encryption key e, as is assumed in the definition of projectivity. The garbled arguments $\overline{X}$ and $\overline{X'}$ can be presented in form $\overline{X} = \overline{X_1} \ldots \overline{X_n}$ and $\overline{X'} = \overline{X'_1} \ldots \overline{X'_n}$ where $\overline{X_i} = (X_i, \widehat{X_i}, s_{h(i,r)})$ and $\overline{X'_i} = (X'_i, \widehat{X'_i}, s_{h(i,r)})$ for all $i \in [1, n]$. If $x_i = x'_i$ then we have that $\overline{X_i} = \overline{X'_i}$. This follows from the following facts. First of all, we have that $X_i = X'_i$ because we assumed that the garbling scheme $\mathcal{G}$ is projective and these two garbled arguments were constructed using scheme $\mathcal{G}$. Secondly, $\widehat{X_i} = \widehat{X'_i}$: the decryptions of $\widehat{X_i}$ and $\widehat{X'_i}$ must be equal, since they were both encrypted with the same encryption

key $\widehat{e}$. Lastly, the shares of of key $\widehat{e}$ are also equal since the choice of the share depends only on the index i and the randomness value r which are the same for arguments x_i and x'_i. This now shows that our claim holds: if $x_i = x'_i$ then $X_i = X'_i$.

Conversely, if $\overline{X_i} = \overline{X'_i}$ then we have that $X_i = X'_i$, $\widehat{X_i} = \widehat{X'_i}$ and the secret shares are equal. Now we must have that $x_i = x'_i$. This follows from the projectivity of $\mathcal{G}$ and $X_i = X'_i$ (which were constructed with the scheme $\mathcal{G}$. This now shows that the projectivity property holds for the garbling scheme $\mathcal{G}'$.

Now we show that the garbling scheme $\mathcal{G}'$ belongs to class $\mathcal{F}_\ell$. First of all, the garbling scheme $\mathcal{G}'$ uses the garbling algorithm `Gb`, `En` and `Ev` from garbling scheme $\mathcal{G}$ and garbling scheme $\mathcal{G}$ is in $\mathcal{F}_\ell$. We show that the parts $\widehat{F}$, $\widehat{X}$ and the shares of $\widehat{e}$ are not useful in trying to get the garbling scheme $\mathcal{G}'$ out of the class $\mathcal{F}_\ell$. The number of shares received by the adversary along with every garbled argument bit is at most $n \cdot \ell < n \cdot \ell + 1$. These $n \cdot \ell$ shares are not useful in reconstructing $\widehat{e}$ and thus the adversary is not able to recover f or x_i's. To conclude, getting $n \cdot \ell$ shares of key $\widehat{e}$ does not increase the winning probability of the adversary, so the garbling scheme $\mathcal{G}'$ belongs to class $\mathcal{F}_\ell$.

Next we show that the constructed scheme $\mathcal{G}'$ does not belong to class $\mathcal{C}_{\ell+1}$. Every garbled argument bit carries a share of secret, and a fully specified argument thus reveals n shares of the secret. An adversary playing the game related to class $\mathcal{C}_{\ell+1}$ now may get as many as $n \cdot (\ell + 1) > n \cdot \ell + 1$ shares of the secret. Thus the adversary is allowed to get enough shares to reconstruct $\widehat{e}$ with certain positive probability that does not depend on the security parameter k. As a consequence, this adversary has a chance (that does not depend on k) of recovering f and x.

Now we show that, using f and x, the adversary has a non-negligible advantage in finding the correct challenge bit b and winning the game. We consider this claim in indistinguishability-based and simulation-based games separately. Here we make use of the assumption $(*)$ presented in the proof of Theorem 2 above.

If the adversary plays indistinguishability-based game then the adversary chooses two Â¨functions f_0, f_1 and two arguments x_0, x_1 satisfying the three properties of assumption $(*)$. Because the adversary now has a positive probability, not depending on the security parameter k, to reconstruct $\widehat{e}$ using his secret shares, he has a positive chance to decrypt $\widehat{F}$ and $\widehat{X}$. The decryption of $\widehat{F}$ reveals which of the two functions, f_0 or f_1, was garbled and the decryption of $\widehat{X}$ reveals which of the corresponding arguments was garbled. In either case the challenge bit b is revealed to the adversary.

In simulation-based game, the adversary chooses either (f_0, x_0) or (f_1, x_1) where f_0, f_1, x_0 and x_1 satisfy the three properties of assumption

$(*)$. Let us denote the choice of adversary by (f, x). We will now show that there is an adversary that has always a chance to win the security game if the adversary is able to decrypt $\widehat{F}$ or $\widehat{X}$. We consider the privacy, the obliviousness and the matchability-only notions separately.

In the privacy notion, the simulator gets $\Phi(f)$ as input when it generates $(F, \widehat{F}, d)$ and $(\Phi(f), y)$ when it generates $(X_i, \widehat{X_i}, s)$. Now, since the adversary has enough shares and is able to reconstruct $\widehat{e}$, the adversary has a chance to decrypt $\widehat{F}$ and $\widehat{X}$. If these were generated by the real garbling and encryption algorithms, then $\widehat{F}$ is decrypted to f and $\widehat{X}$ is decrypted to x. We assumed that there are inputs f_0, f_1, x_1 and x_2 which satisfy the properties of assumption $(*)$. The input (f, x) chosen by the adversary represents either (f_0, x_0) or (f_1, x_1). The simulator is not able to distinguish which of the two inputs was chosen by the adversary because the inputs to the simulator are the same in both cases. Therefore, if $\widehat{F}$ and $\widehat{X}$ were generated by the simulator, then there is a 50% chance that either $\widehat{F}$ is not decrypted to f or $\widehat{X}$ is not decrypted to x.

In matchability-only notion, the simulator gets the same input as in the privacy notion. Therefore, the same arguments can be used for showing that there is an adversary that has a chance to distinguish the simulator-generated $\widehat{F}$ and $\widehat{X}$ from those generated by the real garbling algorithms $\mathtt{Gb}'$ and $\mathtt{En}'$.

In the obliviousness notion, the simulator gets only $\Phi(f)$ as its input when it generates the garbled function $F' = (F, \widehat{F})$ or the garbled argument $X' = (X, \widehat{X}, s)$. In this case, the simulator does not get y as its input. The simulator's work of finding $\widehat{X}$ such that $\widehat{X}$ is decrypted to x cannot become easier when it is not given y as input. We can use the above arguments to show that there is at least 50% that the simulator generates F' and X' such that either the decryption of $\widehat{F}$ is different from f or the decryption of $\widehat{X}$ is different from x. In either case, the adversary has a chance to find out the value of the challenge bit.

In all above cases, the adversary gets enough shares of $\widehat{e}$, i.e. $n \cdot (\ell + 1)$, the adversary has a positive probability, not depending on k, to decrypt $\widehat{F}$ and $\widehat{X}$. This implies that the adversary has a chance to find out which function f and argument x represent the encrypted counterparts $\overline{F}$ and $\overline{X}$. This in turn reveals, with certain probability (that does not depend on k), the value of the challenge bit. In this case the adversary always wins the game. This implies that the adversary has a non-negligible advantage (not depending on k) in the game.

As a conclusion, learning f and x reveals the challenge bit b. Therefore, the adversary always has a non-negligible advantage (not depending on k) in the security game for class $\mathcal{C}_{\ell+1}$. This shows that $\mathcal{G}' \notin \mathcal{C}_{\ell+1}$. □

Proof of Theorem 6:

For the first part, let us assume that a garbling scheme $\mathcal{G}$ is prv.ind.padap$_\ell$ secure. Our aim is to prove that $\mathcal{G}$ is also mao.ind.padap$_\ell$ secure.

The only difference between PrvIndPadap$_\ell$ and MaoIndPadap$_\ell$ games is the `GARBLE` procedure. In PrvIndPadap$_\ell$ game the adversary gets (F, d) whereas in MaoIndPadap$_\ell$ game the adversary gets solely F. The `INPUT` procedures give the same amount of information for both adversaries, because the procedures are identical and called equally many times (from 0 to at most $n \cdot \ell$ times).

Therefore, having less information in MaoIndPadap$_\ell$ game cannot increase the adversary's winning probability compared to the win probability in PrvIndPadap$_\ell$ game. This implies that the adversary's advantage in game MaoIndPadap$_\ell$ is smaller than or equal to the advantage in game PrvIndPadap$_\ell$. Since we assumed that $\mathcal{G}$ is prv.ind.padap$_\ell$ secure, adversary's advantage in game PrvIndPadap$_\ell$ is negligible. This implies that adversary's advantage is negligible in game MaoIndPadap$_\ell$ as well, proving that $\texttt{GS}(\text{prv.ind.padap}_\ell, \Phi) \subseteq \texttt{GS}(\text{mao.ind.padap}_\ell, \Phi)$.

For the second part, let us assume that a garbling scheme $\mathcal{G}$ is obv.ind.padap$_\ell$ secure. Our aim is to prove that $\mathcal{G}$ is also mao.ind.padap$_\ell$ secure.

The `GARBLE` procedures for both games ObvIndPadap$_\ell$ and MaoIndPadap$_\ell$ are equal, whereas the `INPUT` procedures differ. If `INPUT` procedure is called less than n times on the same argument, then the adversary in ObvIndPadap$_\ell$ game has the same amount of information as the adversary in the MaoIndPadap$_\ell$ game. Every time the `INPUT` is called n times and one argument becomes fully specified, then the MaoIndPadap$_\ell$ adversary must pass the evaluation test $\texttt{ev}(f_0, x_0) = \texttt{ev}(f_1, x_1)$ which is not a part of ObvIndPadap$_\ell$ game.

Having to pass the test cannot be easier than not having the test at all. Therefore, the adversary's winning probability in game MaoIndPadap$_\ell$ cannot be greater than in game ObvIndPadap$_\ell$. This implies that the adversary's advantage in game MaoIndPadap$_\ell$ is smaller than or equal to the advantage in game ObvIndPadap$_\ell$. Since we assumed $\mathcal{G}$ to be obv.ind.padap$_\ell$ secure, the advantage in game ObvIndPadap$_\ell$ is negligible. This implies that the advantage in game MaoIndPadap$_\ell$ is also negligible, which now proves that $\mathcal{G}$ is mao.ind.padap$_\ell$ secure. □

Proof of Theorem 7: First consider the claim $\texttt{GS}(\text{prv.sim.padap}_\ell) \subseteq \texttt{GS}(\text{mao.sim.padap}_\ell)$. To prove the claim, let $\mathcal{G}$ be a prv.sim.padap$_\ell$ secure garbling scheme. Our aim is to prove that $\mathcal{G}$ is also a mao.sim.padap$_\ell$ secure garbling scheme.

Let $\mathcal{A}$ be an arbitrary adversary playing the game MaoSimPadap$_\ell$. We construct an adversary $\mathcal{B}$ for game PrvSimPadap$_\ell$. Adversary $\mathcal{B}$ uses $\mathcal{A}$ as

subroutine in the same manner as in the proof of Theorem 5 in this same publication III.

Let us consider the two games PrvSimPadap$_\ell$ and MaoSimPadap$_\ell$. The only difference in PrvSimPadap$_\ell$ and MaoSimPadap$_\ell$ games is the `GARBLE` procedure. In PrvSimPadap$_\ell$ game adversary $\mathcal{B}$ gets (F, d) whereas in MaoSimPadap$_\ell$ game adversary $\mathcal{A}$ gets F without d. Therefore, in the game PrvSimPadap$_\ell$ adversary $\mathcal{B}$ is allowed to compute y by using the decryption key d whereas in MaoSimPadap$_\ell$ adversary $\mathcal{A}$ has no d. The `INPUT` procedures give the same amount of information for both adversaries, because the procedures are identical and called equally many times (from 0 to at most ℓ times). Hence, adversary $\mathcal{B}$ is able to correctly emulate the game MaoSimPadap$_\ell$ to adversary $\mathcal{A}$, so adversary $\mathcal{A}$ may use other strategies than guessing when it gives its answer $b_\mathcal{A}$ to adversary $\mathcal{B}$. Since adversary $\mathcal{B}$ uses $\mathcal{A}$'s answer in his game and the correct answer b is the same in both games, both adversaries have the same winning probability in their games.

This implies that the advantage of adversary $\mathcal{A}$ in game MaoSimPadap$_\ell$ is the same as the advantage of adversary $\mathcal{B}$ in the PrvSimPadap$_\ell$ game. Since we assumed that $\mathcal{G}$ is a prv.sim.padap$_\ell$ secure garbling scheme, the advantage in PrvSimPadap$_\ell$ game is negligible for adversary $\mathcal{B}$. Thus, the advantage of an arbitrary adversary $\mathcal{A}$ in MaoSimPadap$_\ell$ game is negligible. This proves that `GS`(PrvSimPadap$_\ell$) $\subseteq$ `GS`(MaoSimPadap$_\ell$).

Then consider the claim `GS`(obv.sim.padap$_\ell$) $\subseteq$ `GS`(mao.sim.padap$_\ell$). To prove the claim, let $\mathcal{G}$ be a obv.sim.padap$_\ell$ secure garbling scheme. Our aim is to prove that $\mathcal{G}$ is also a mao.sim.padap$_\ell$ secure garbling scheme.

Let $\mathcal{A}$ be an arbitrary adversary playing the game MaoSimPadap$_\ell$. We construct an adversary $\mathcal{B}$ for game ObvSimPadap$_\ell$. Adversary $\mathcal{B}$ uses $\mathcal{A}$ as subroutine in the same manner as in the proof of Theorem 5 in this publication III.

Let $\mathcal{S}$ be the simulator for adversary $\mathcal{B}$ in game ObvSimPadap$_\ell$. The simulator $\mathcal{S}$ is takes $(1^k, \Phi(f))$ as input when it is called in `GARBLE` procedure. In `INPUT` procedure, the simulator $\mathcal{S}$ takes $(i, |Q|)$. Using simulator $\mathcal{S}$, we construct a simulator $\mathcal{S}'$ for adversary $\mathcal{A}$in game MaoSimPadap$_\ell$. When the simulator $\mathcal{S}'$ is called with input $(1^k, \Phi(f))$ in `GARBLE` procedure in game MaoSimPadap$_\ell$ then $\mathcal{S}'$ calls simulator $\mathcal{S}$ with the same input $(1^k, \Phi(f))$. When the simulator $\mathcal{S}'$ is called with input $(\tau, i, |Q|)$ in the `INPUT` procedure in game MaoSimPadap$_\ell$ then $\mathcal{S}'$ omits the first part τ in its input and calls $\mathcal{S}$ with input $(i, |Q|)$.

First, the actual game ObvSimPadap$_\ell$ begins with adversary $\mathcal{B}$. Adversary $\mathcal{B}$ is asked to provide input, a function f, to the `GARBLE` procedure. Instead of choosing the input himself, adversary $\mathcal{B}$ asks adversary $\mathcal{A}$ to start game MaoSimPadap$_\ell$. Adversary $\mathcal{A}$ presumes to play the actual MaoSimPadap$_\ell$, not an emulated game. Therefore, when $\mathcal{B}$ asks for

a function as input to GARBLE procedure in MaoSimPadap$_\ell$ game, adversary $\mathcal{A}$ sends a function f to $\mathcal{B}$. Adversary $\mathcal{B}$ then sends the function f to his GARBLE procedure in game ObvSimPadap$_\ell$. Based on the challenge bit, GARBLE procedure in the actual game either uses the garbling algorithm or the simulator $\mathcal{S}$ to generate the garbled function F as output. F is sent to adversary $\mathcal{B}$ who sends it to adversary $\mathcal{A}$.

Then adversary is asked to provide an argument bit as input to INPUT procedure. Again, $\mathcal{B}$ does not choose the argument bit himself; instead he asks adversary $\mathcal{A}$ to provide an argument bit as input to the INPUT procedure in the emulated game. Adversary $\mathcal{A}$ sends his input (i, c) to adversary $\mathcal{B}$ who sends it to his INPUT procedure. The INPUT procedure in $\mathcal{B}'s$ game now generates garbled argument F by using either the garbling algorithm En or the simulator $\mathcal{S}$. The INPUT procedure sends F to $\mathcal{B}$ who sends it to $\mathcal{A}$.

The above construction shows that adversary $\mathcal{B}$ is able to correctly emulate the MaoSimPadap$_\ell$ game with simulator $\mathcal{S}'$ for adversary $\mathcal{A}$. Therefore, adversary $\mathcal{A}$ cannot distinguish whether he plays an actual MaoSimPadap$_\ell$ game or an emulated MaoSimPadap$_\ell$ game. Adversary $\mathcal{A}$ is able to answer the challenge of game MaoSimPadap$_\ell$. Adversary $\mathcal{B}$ uses $\mathcal{A}$'s answer in his game. Furthermore, the answer b in both games is the same. Consequently, both adversaries have the same winning probability in their games and hence the advantage of both adversaries in their games are also equal.

According to our assumption, $\mathcal{G}$ is a ObvSimPadap$_\ell$ secure garbling scheme: There is a simulator such that the advantage of $\mathcal{B}$ is negligible. Let the simulator $\mathcal{S}$ mentioned above be such a simulator. Now, the simulator $\mathcal{S}'$ constructed above is such that the simulator $\mathcal{S}'$ makes the advantage of $\mathcal{A}$ the same as the advantage of adversary $\mathcal{B}$. In other words, we have found a simulator such that the advantage of adversary $\mathcal{A}$ is negligible. This now proves the claim GS(obv.sim.padap$_\ell$) $\subseteq$ GS(mao.sim.padap$_\ell$) which concludes the proof. □

6.2 Original publication V

Proof of Theorem 4:

It is fairly obvious that $\mathcal{R}_* \subseteq \mathcal{R}_\ell$ for any $\ell \in \mathbb{N}$ because the adversary playing the game related to $\mathcal{R}_*$ can always make arbitrarily many calls to GARBLE_ARG procedure, especially exactly ℓ calls. The claim $\mathcal{R}_* \subseteq \bigcap_{\ell=1}^{\infty} \mathcal{R}_\ell$ follows.

The second part of the theorem claims that there is a garbling scheme that belongs to the intersection $\bigcap_{\ell=1}^{\infty} \mathcal{R}_\ell$ but does not belong to the class $\mathcal{R}_*$, assuming that the intersection is nonempty. To prove this claim, we start with a garbling scheme $\mathcal{G} = (\mathtt{KeyGen}, \mathtt{Ga}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev}) \in \bigcap_{\ell=1}^{\infty} \mathcal{R}_\ell$ and

construct another garbling scheme $\mathcal{G}' = (\texttt{KeyGen}', \texttt{Ga}', \texttt{En}', \texttt{De}, \texttt{Ev}', \texttt{ev})$ as follows.

The modified key generation algorithm $\texttt{KeyGen}'$ takes $(1^k, m, n, p)$as input and outputs (g', e', d) where $g' = (g, g_2, t \text{ shares of key } g_2)$ and $e' = (e, g_2)$. Keys g, e and d are generated by the key generation algorithm $\texttt{KeyGen}$ of garbling scheme $\mathcal{G}$. The key g_2 in is an encryption key from a completely independent symmetric, non-deterministic encryption scheme. We use key g_2 to encrypt both the argument x and the function f with encryption algorithm $\texttt{En}_2$. For the third component of g', we construct a secret sharing scheme where the key g_2 is the secret divided into t shares. Exactly k shares are needed to reconstruct g_2, where k represents the security parameter.

The modified garbling algorithm $\texttt{Ga}'$ takes $g' = (g, g_2, t \text{ shares})$, f as input and computes (F, F_2, s) where $F = \texttt{Ga}(g, f)$, $F_2 = \texttt{En}_2(g_2, f)$ and s is a random share of key g_2. The modified encryption algorithm $\texttt{En}'$ takes $((e, g_2), x)$ as input and outputs (X, X_2) where $X = \texttt{En}(e, x)$ and $X_2 = \texttt{En}_2(g_2, x)$. The modified garbled evaluation function $\texttt{Ev}'$ takes $((F, F_2, s), (X, X_2))$ as input. The algorithm omits all new parts F_2, X_2, s and computes $Y = \texttt{Ev}(F, X)$.

First we prove that $\mathcal{G}' \in \bigcap_{\ell=1}^{\infty} \mathcal{R}_\ell$. To do that, we first prove that $\mathcal{G}' \in \mathcal{R}_\ell$ for all $\ell \in \mathbb{N}$ from which the claim follows. Consider an arbitrary adversary playing the game related to class $\mathcal{R}_\ell$. The win probability of the adversary now depends on the parameters ℓ and k, because these two variables determine whether the adversary could have enough shares to reconstruct the second encryption key e_2. If $k > \ell$ then the adversary does not have enough shares to reconstruct e_2. In this case, the advantage in the game will be the same as in the game related to the original garbling scheme $\mathcal{G}$, i.e. the advantage is negligible. Parameter k can always be chosen in the way that $k > \ell$ holds, which ascertains that $\mathcal{G}' \in \mathcal{R}_\ell$.

We still have to prove the claim $\mathcal{G}' \notin \mathcal{R}_*$. An adversary playing the game related to $\mathcal{R}_*$ is allowed to call `GARBLE_FUNC` procedure arbitrarily many times, especially more than k times. If the adversary calls `GARBLE_FUNC` procedure k^2 times then the well-known approximations related to birthday paradox imply that there is a non-zero probability, not depending from k, to find key e_2 and hence win the game by decrypting F_2. Next we show why this is the case.

The adversary can always choose the functions in the security game related to class $\mathcal{R}_*$. We consider next the indistinguishability-based and simulation-based games separately.

If the game is related to indistinguishability-based security notion then the adversary chooses two functions f_0, f_1 and two arguments x_0, x_1 which satisfy the three properties in assumption $(*)$. These inputs pass all the tests in the game and hence the adversary is able to get F_2 and X_2. In

addition, the adversary is able to recover the encryption key e_2 using the shares received together with the garbled functions. In this way, the adversary is able to find out which of the functions or which of the arguments was garbled because he has a chance to decrypt F_2 and X_2. In this way the adversary has a chance to find the value of the challenge bit.

If the game is related to simulation-based security notion then the adversary chooses either (f_0, x_0) or (x_1, f_1) where f_0, f_1, x_0, x_1 satisfy the three properties of assumption $(*)$. Let us denote the choice of the adversary by (f, x). The simulator is not able to distinguish which of (f_0, x_0) or (f_1, x_1) was chosen by the adversary based on its input since both choices have the same side-information and the result of evaluation is the same for both choices. Therefore, there is at least 50% chance that the simulator generates an encrypted function F_2 which is decrypted to a function $f' \neq f$ or the simulator generates an encrypted argument X_2 which is decrypted to an argument $x' \neq x$. The adversary has now a positive chance, not depending on k, to distinguish whether the garbled argument and the garbled function were constructed by using the actual garbling algorithms or by using the simulator, since the adversary has obtained enough shares of key e_2. This implies that the adversary has a non-negligible advantage in the simulation-based game.

In either case, the adversary has a non-negligible advantage in the game, which in turn yields that $\mathcal{G}'$ cannot belong to class $\mathcal{R}_*$. This concludes the proof. □

Proof of Theorem 6: Let us assume that $\mathcal{G} \in \texttt{GS}(\text{xxx.sim.radap}_\ell, \Phi)$. Our goal is to prove that $\mathcal{G} \in \texttt{GS}(\text{xxx.ind.radap}_\ell, \Phi)$. Let $\mathcal{A}$ be an adversary playing the XxxIndRadap_ℓ game and let us construct an adversary $\mathcal{B}$ for the XxxSimRadap_ℓ game. $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine as explained in the following.

The game XxxSimRadap_ℓ starts with `INITIALIZE` procedure in which the challenge bit is chosen uniformly at random. After that, the game tells the adversary $\mathcal{B}$ to start the game. Adversary $\mathcal{B}$ in turn challenges $\mathcal{A}$ to play a game XxxIndRadap_ℓ. Adversary $\mathcal{A}$ presumes that the challenge comes from a real XxxIndRadap_ℓ game, so $\mathcal{A}$ prepares its `GARBLE_ARG` input x_0, x_1 and sends them to $\mathcal{B}$. If x_0 or x_1 is not a bit string of length m, then $\mathcal{B}$ sends $\perp$ to $\mathcal{A}$. $\mathcal{B}$ lets $c \twoheadleftarrow \{0, 1\}$ and sends x_c to its `GARBLE_ARG` regardless of what was the result of the previous length test for the arguments x_0, x_1. Up to this point the games progress similarly in all three cases, but from now on the progress is slightly different. The progress of the game PrvSimRadap_ℓ is illustrated in fig. 6.1, the other cases follow the same idea.

In game PrvSimRadap_ℓ, $\mathcal{B}$ gets (X, d) as return. $\mathcal{B}$ sends (X, d) to $\mathcal{A}$, given that both arguments x_0 and x_1 are appropriate (both are bit strings of length m). Now $\mathcal{A}$ starts sending functions f_0, f_1 to $\mathcal{B}$. If $\Phi(f_0) \neq \Phi(f_1)$,

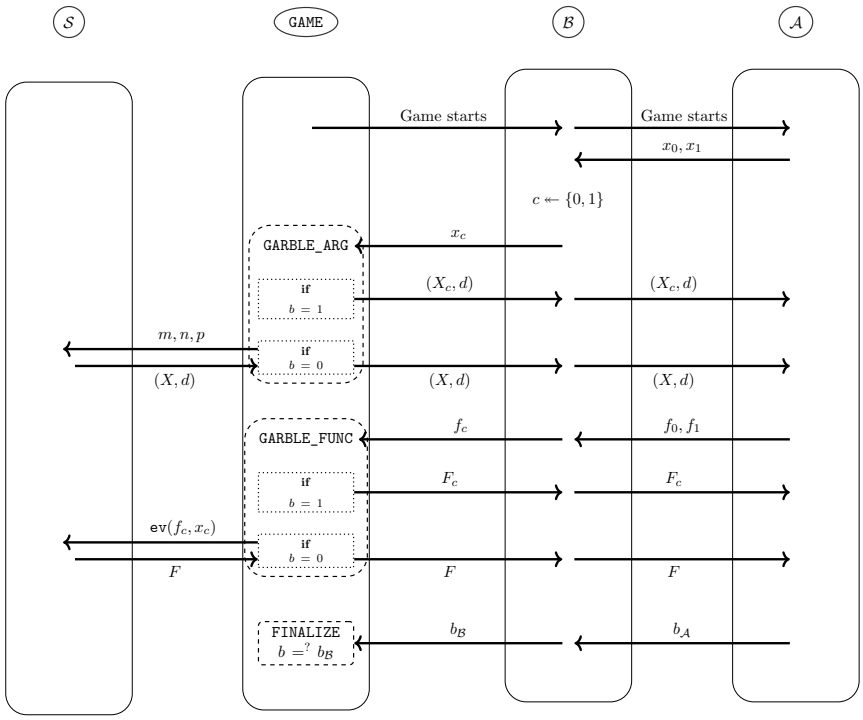


Figure 6.1: In the above figure, $\mathcal{B}$ is an adversary playing the game PrvSimRadap_ℓ denoted in the diagram by `GAME`. $\mathcal{S}$ is the simulator in this game. $\mathcal{A}$ is an adversary presuming to play game PrvIndRadap_ℓ, but actually adversary $\mathcal{B}$ uses $\mathcal{A}$ as a subroutine by trying to emulate the actual game PrvIndRadap_ℓ.

$\{x_0, x_1\} \not\subseteq \{0,1\}^m$ or $\mathsf{ev}(f_0, x_0) \neq \mathsf{ev}(f_1, x_1)$, $\mathcal{B}$ sends $\perp$ to $\mathcal{A}$. Regardless of possibly sending $\perp$ to $\mathcal{A}$, $\mathcal{B}$ sends f's to its own `GARBLE_FUNC`. Every time, $\mathcal{B}$ gets an F as a return. $\mathcal{B}$ forwards F to $\mathcal{A}$ if none of the previous tests failed. This may be repeated at most ℓ times.

The games MaoSimRadap_ℓ and MaoIndRadap_ℓ are similar to games PrvSimRadap_ℓ and PrvIndRadap_ℓ respectively. The only difference is that the adversary $\mathcal{B}$ does not get decryption key d from `GARBLE_ARG` procedure in MaoSimRadap_ℓ game or in MaoIndRadap_ℓ game.

The games ObvSimRadap_ℓ and ObvIndRadap_ℓ are similar to games MaoSimRadap_ℓ and MaoIndRadap_ℓ respectively. The only difference to MaoSimRadap_ℓ and MaoIndRadap_ℓ games is that the test $\mathsf{ev}(f_0, x_0) =^? \mathsf{ev}(f_1, x_1)$ is not performed in `GARBLE_FUNC` procedure.

In all three cases $\mathcal{A}$ returns its answer $b_{\mathcal{A}}$ to the challenge by sending $b_{\mathcal{A}}$ to adversary $\mathcal{B}$. Adversary $\mathcal{B}$ answers $b_{\mathcal{B}} = 1$ if and only if $b_{\mathcal{A}} = c$ and none of the tests in procedures `GARBLE_ARG` or `GARBLE_FUNC` ended in $\perp$. $\mathcal{B}$ answers 0 otherwise.

Let us now analyze the different outcomes of the game to find out the win probabilities of both adversaries.

If either argument, x_0 or x_1 is not a bit string of length m, then $\mathcal{B}$ answers 0 regardless of $\mathcal{A}$'s answer. This is the correct answer with probability $\frac{1}{2}$.

The next possibility is that both arguments, x_0 and x_1, are bit strings of length m but all inputs (x_0, x_1) given by adversary $\mathcal{A}$ yield $\perp$ in procedure `GARBLE_FUNC` in the it is playing, i.e. either PrvIndAdap_ℓ, MaoIndAdap_ℓ or ObvIndAdap_ℓ. Also in this case $\mathcal{B}$ answers 0 regardless of $\mathcal{A}$'s answer. The win probability of $\mathcal{B}$ is therefore $\frac{1}{2}$.

The third scenario is that both arguments, x_0 and x_1, are bit strings of length m and at least one call to `GARBLE_FUNC` did not end in $\perp$. There are two possibilities for b. First consider case $b = 0$ when X and F are created by the simulator $\mathcal{S}$. In this case, $\mathcal{A}$ may notice that his (emulated) game has deviated from the usual description by returning unexpected garbled values. In this case, adversary $\mathcal{A}$ may refuse to return any answer to adversary $\mathcal{B}$ (e.g. by returning $\perp$). Then, the adversary $\mathcal{B}$ answers 0 in his game which is the correct answer.

Another option is that $\mathcal{A}$ still continues his game despite of the fact that the game returns unexpected garblings. In this case, $\mathcal{A}$ does not receive any information about X or F, so its answer is no better (and no worse) than a guess. Right guess still has probability $\frac{1}{2}$. Moreover, $\mathcal{B}$ loses its game if and only if $\mathcal{A}$ wins its game. Consider for example the following scenario. If $b_{\mathcal{A}} = 0$ and $c = 0$ then $b_{\mathcal{B}} = 1$ because $b_{\mathcal{A}} = c$. It follows that adversary $\mathcal{A}$ wins its game ($b_{\mathcal{A}} = c$), but adversary $\mathcal{B}$ loses its game ($b_{\mathcal{B}} \neq b$). The three other possibilities end up in a similar situation where the other adversary wins its game if and only if the other loses.

On the other hand, if $b = 1$ then adversary $\mathcal{B}$ is able to emulate the actual XxxIndRadap_ℓ game because F and X are created by the actual garbling algorithm from either x_0, f_0 or x_1, f_1, giving adversary $\mathcal{A}$ a chance to figure out the correct answer $b_{\mathcal{A}}$. Therefore, $\mathcal{A}$ has an advantage $\mathbf{Adv}_{\mathcal{A}}$ of answering correctly to the challenge in its game. Adversary $\mathcal{B}$ wins the game only if $b_{\mathcal{A}} = c$ and $\mathcal{A}'s$ inputs pass all the tests made by adversary $\mathcal{B}$. The win probability of $\mathcal{B}$ is now $\frac{1}{2} + \frac{1}{2} \cdot \mathbf{Adv}_{\mathcal{A}}$.

This case analysis lets us derive the win probability of adversary $\mathcal{B}$ against any `PT` simulator $\mathcal{S}$ as follows.

$$\begin{aligned}\Pr[\mathcal{B} \text{ wins }] &= \frac{1}{2}\Pr[\mathcal{B} \text{ wins } | b = 1] + \frac{1}{2}\Pr[\mathcal{B} \text{ wins } | b = 0] \\ &= \frac{1}{2}\left(\frac{1}{2} + \frac{1}{2}\mathbf{Adv}_{\mathcal{A}}\right) + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2} + \frac{1}{4} \cdot \mathbf{Adv}_{\mathcal{A}}.\end{aligned}$$

Based on the definition of the advantage of the adversary, we have that

$$\mathbf{Adv}_{\mathcal{B}} = 2 \cdot \Pr[\mathcal{B} \text{ wins }] - 1.$$

From this we can derive the following identity:

$$\Pr[\mathcal{B} \text{ wins }] = \frac{1}{2} \cdot \mathbf{Adv}_{\mathcal{B}} + \frac{1}{2}$$

Now, using $\Pr[\mathcal{B} \text{ wins }] = \frac{1}{2} + \frac{1}{2} \cdot \mathbf{Adv}_{\mathcal{B}}$ and $\Pr[\mathcal{B} \text{ wins }] = \frac{1}{2} + \frac{1}{4} \cdot \mathbf{Adv}_{\mathcal{A}}$, we obtain $\mathbf{Adv}_A = 2 \cdot \mathbf{Adv}_{\mathcal{B}}$.

We assumed $\mathcal{G}$ to be prv.sim.adap$_\ell$ secure, so there is a simulator $\mathcal{S}$ such that the advantage is negligible for all $\mathcal{PT}$ adversaries. Especially, the advantage of adversary $\mathcal{B}$ is negligible. Because we have that $2 \cdot \mathbf{Adv}_{\mathcal{B}} = \mathbf{Adv}_{\mathcal{A}}$, the advantage of adversary $\mathcal{A}$ is also negligible. Adversary $\mathcal{A}$ is an arbitrary $\mathcal{PT}$ adversary, which lets us conclude that any $\mathcal{PT}$ adversary playing the security game in the indistinguishability-based model has a negligible advantage, proving that $\mathcal{G}$ is prv.sim.adap$_\ell$ secure. □

Proof of Theorem 7: Our aim is to prove that

$$\texttt{GS}(\text{prv.ind.radap}_\ell, \Phi) \cup \texttt{GS}(\text{obv.ind.radap}_\ell, \Phi) \subseteq \texttt{GS}(\text{mao.ind.radap}_\ell, \Phi).$$

To prove the claim, we have to prove that indistinguishability-based privacy implies indistinguishability-based matchability-only notion and that indistinguishability-based obliviousness implies indistinguishability-based matchability-only notion for garbling schemes achieving reverse-order adaptive security.

Consider first the inclusion $\texttt{GS}(\text{prv.ind.radap}_\ell, \Phi) \subseteq \texttt{GS}(\text{mao.ind.radap}_\ell, \Phi)$. Suppose that $\mathcal{G}$ is a prv.ind.radap$_\ell$ secure garbling scheme over Φ. Let $\mathcal{A}$ be an arbitrary adversary playing the game MaoIndRadap$_\ell$. We construct another adversary $\mathcal{B}$ for game PrvIndRadap$_\ell$ which uses $\mathcal{A}$ as a subroutine as follows.

When adversary $\mathcal{B}$ is asked to give his argument to the `GARBLE_ARG` procedure in game PrvIndRadap$_\ell$ game the adversary $\mathcal{B}$ challenges adversary $\mathcal{A}$ to start game MaoIndRadap$_\ell$ and provide an argument for the `GARBLE_ARG` procedure in this game. Adversary $\mathcal{A}$ presumes to play the actual MaoIndRadap$_\ell$ game so it sends arguments x_0 and x_1 to $\mathcal{B}$. Adversary $\mathcal{B}$ sends these arguments to the `GARBLE_ARG` procedure in his game. The `GARBLE_ARG` in PrvIndRadap$_\ell$ checks first whether the two arguments are of the same length. If they are not, then the procedure sends $\perp$ to $\mathcal{B}$. Adversary $\mathcal{B}$ then sends $\perp$ to $\mathcal{A}$. If the arguments x_0 and x_1 are of correct length, then the `GARBLE_ARG` procedure in game PrvIndRadap$_\ell$ garbles x_b based on the choice of the challenge bit b. The procedure sends (X, d) to adversary $\mathcal{B}$. Adversary $\mathcal{B}$ then sends only X to $\mathcal{A}$, keeping d as its own information.

Then adversary $\mathcal{A}$ starts sending the functions f_0 and f_1 to adversary $\mathcal{B}$. Adversary $\mathcal{B}$ sends these functions to its `GARBLE_FUNC` procedure. The

GARBLE_FUNC procedure performs the checks $\{x_0, x_1\} \in^? \{0,1\}^m$, $\Phi(f_0) =^? \Phi(f_1)$ and $\mathtt{ev}(f_0, x_0) =^? \mathtt{ev}(f_1, x_1)$. If any of these three checks fail, the GARBLE_FUNC procedure sends $\perp$ to $\mathcal{B}$. Adversary $\mathcal{B}$ then forwards $\perp$ to $\mathcal{A}$. If none of the checks fail, then GARBLE_FUNC procedure garbles the function f_b based on the choice of the challenge bit and sends F to adversary $\mathcal{B}$. Adversary $\mathcal{B}$ sends F to $\mathcal{A}$.

The queries to GARBLE_FUNC are repeated at most ℓ times. Then adversary $\mathcal{A}$ sends his answer $b_{\mathcal{A}}$ to adversary $\mathcal{B}$. Adversary $\mathcal{B}$ uses $b_{\mathcal{A}}$ as his own answer and sends it to his FINALIZE procedure in game PrvIndRadap_ℓ. Let us now analyze the different outcomes of the game. The adversary $\mathcal{B}$ is able to correctly emulate game MaoIndRadap_ℓ to adversary $\mathcal{A}$. In addition, the answers of $\mathcal{A}$ and $\mathcal{B}$ are identical, so both adversaries have the same winning probability to win their own games, implying that the advantages of both adversaries are equal. Since we assumed that the garbling scheme $\mathcal{G}$ is $\text{prv.ind.radap}_\ell$ secure, the advantage of adversary $\mathcal{B}$ is negligible. This yields that the advantage of an arbitrary adversary playing game MaoIndRadap_ℓ is also negligible, which proves the first inclusion in the claim.

Consider then the second inclusion $\mathtt{GS}(\text{obv.ind.radap}_\ell, \Phi) \subseteq \mathtt{GS}(\text{mao.ind.radap}_\ell, \Phi)$. Suppose that $\mathcal{G} \in \mathtt{GS}(\text{obv.ind.radap}_\ell, \Phi)$. Let $\mathcal{A}$ be an arbitrary adversary playing the game MaoIndRadap_ℓ. We construct an adversary $\mathcal{B}$ playing the game ObvIndRadap_ℓ which uses adversary $\mathcal{A}$ as his subroutine as follows.

The ObvIndRadap_ℓ game starts by the choice of the challenge bit b in INITIALIZE procedure. Then the ObvIndRadap_ℓ game asks adversary $\mathcal{B}$ to give its input arguments x_0 and x_1 to the GARBLE_ARG procedure. Instead of choosing the arguments himself, adversary $\mathcal{B}$ challenges $\mathcal{A}$ to play MaoIndRadap_ℓ game. Since adversary $\mathcal{A}$ presumes to play the actual MaoIndRadap_ℓ game, he sends x_0 and x_1 to $\mathcal{B}$. Adversary $\mathcal{B}$ sends (x_0, x_1) to GARBLE_ARG procedure in his game.

The procedure checks whether the arguments x_0 and x_1 are of correct length. If this is not the case, then the procedure sends $\perp$ to $\mathcal{B}$. Adversary $\mathcal{B}$ sends $\perp$ to $\mathcal{A}$. If the arguments are both of correct length, then the GARBLE_ARG procedure garbles argument x_b based on the choice of the challenge bit sends X to $\mathcal{B}$. Adversary $\mathcal{B}$ sends X to A.

Then adversary $\mathcal{A}$ starts sending functions f_0 and f_1 to $\mathcal{B}$ who then sends them to his GARBLE_FUNC procedure. First, the procedure performs the checks $\{x_0, x_1\} \in^? \{0,1\}^m$ and $\Phi(f_0) =^? \Phi(f_1)$. If any of these two checks fail, the GARBLE_FUNC procedure sends $\perp$ to $\mathcal{B}$. Adversary $\mathcal{B}$ then forwards $\perp$ to $\mathcal{A}$. If none of the checks fail, then GARBLE_FUNC procedure garbles the function f_b based on the choice of the challenge bit and sends F to adversary $\mathcal{B}$. Adversary $\mathcal{B}$ now computes $y_0 = \mathtt{ev}(f_0, x_0)$ and $y_1 = \mathtt{ev}(f_1, x_1)$ and checks whether $y_0 = y_1$. If this is not the case, then adversary $\mathcal{B}$ sends $\perp$ to $\mathcal{A}$. Otherwise, adversary $\mathcal{B}$ sends F to $\mathcal{A}$.

The queries to GARBLE_FUNC are repeated at most ℓ times. Then adversary $\mathcal{A}$ sends his answer $b_{\mathcal{A}}$ to adversary $\mathcal{B}$. Adversary $\mathcal{B}$ uses $b_{\mathcal{A}}$ as his own answer and sends it to his FINALIZE procedure in game PrvIndRadap$_\ell$. Let us now analyze the different outcomes of the game. The adversary $\mathcal{B}$ is able to correctly emulate game MaoIndRadap$_\ell$ to adversary $\mathcal{A}$. In addition, the answers of $\mathcal{A}$ and $\mathcal{B}$ are identical, so both adversaries have the same winning probability to win their own games, implying that the advantages of both adversaries are equal. Since we assumed that the garbling scheme $\mathcal{G}$ is obv.ind.radap$_\ell$ secure, the advantage of adversary $\mathcal{B}$ is negligible. This yields that the advantage of an arbitrary adversary playing game MaoIndRadap$_\ell$ is also negligible, which proves the second inclusion in the claim. □

Proof of Theorem 8: Our aim is to prove that GS(prv.sim.radap$_\ell$) ∪ GS(obv.sim.radap$_\ell$) ⊆ GS(mao.sim.radap$_\ell$). To prove the claim, we have to show that simulation-based privacy implies simulation-based matchability-only notion and simulation-based obliviousness implies simulation-based matchability-only notion for garbling schemes achieving reverse-order adaptive security.

First, assume that garbling scheme $\mathcal{G}$ is prv.sim.radap$_\ell$ secure. Our aim is to prove that then $\mathcal{G}$ is also mao.sim.adap$_\ell$ secure. There is only one difference in the games PrvSimAdap$_\ell$ and MaoSimAdap$_\ell$: in game PrvSimAdap$_\ell$, GARBLE_ARG returns the decryption key d together with X whereas in game MaoSimAdap$_\ell$, only the garbled argument is returned. Omitting the decryption key d from (X, d) does not increase the winning probability of an adversary playing the MaoSimRadap$_\ell$ game. This proves the first claim.

Then, consider the second claim GS(obv.sim.adap$_\ell, \Phi$) ⊆ GS(mao.sim.adap$_\ell, \Phi$). Suppose that the garbling scheme $\mathcal{G}$ is obv.sim.adap$_\ell$ secure. Intuitively, it is clear that the simulator's additional input y cannot make its task of producing a good output (F, X) more difficult in the MaoSimRadap$_\ell$ game. Let $\mathcal{A}$ be an arbitrary adversary playing the MaoSimRadap$_\ell$ game and let $\mathcal{A}'$ be the corresponding adversary playing the ObvSimRadap$_\ell$ game. Adversary $\mathcal{A}'$ emulates the behavior of adversary $\mathcal{A}$ as before.

Because $\mathcal{G}$ is an obv.sim.radap$_\ell$ secure garbling scheme, there is a simulator $\mathcal{S}'$ such that it makes the advantage of $\mathcal{A}'$ negligible. Our aim is to construct a simulator $\mathcal{S}$ for the MaoSimRadap$_\ell$ game that makes the advantage of $\mathcal{A}$ negligible. On input $(1^k, m, n, p)$ the simulator $\mathcal{S}$ simply calls $\mathcal{S}'$ on the same input. On input $(\Phi(f), y)$ the simulator $\mathcal{S}$ omits y and calls $\mathcal{S}'(\Phi(f))$ to get F. Now, the advantage of both adversaries is the same because both simulators and both adversaries behave identically. Since $\mathcal{A}'$ has a negligible advantage in its ObvSimRadap$_\ell$ game and the advantage of $\mathcal{A}$ in game MaoSimRapad$_\ell$ is the same as the advantage of $\mathcal{A}'$, the advantage of $\mathcal{A}$ is negligible.

We can now conclude that we have found a simulator $\mathcal{S}$ for an arbitrary adversary $\mathcal{A}$ such that the advantage of $\mathcal{A}$ is negligible. This proves the claim that $\mathcal{G}$ is also mao.sim.radap$_\ell$ secure.

Combining the two results above we have proven the original claim $\mathtt{GS}(\text{prv.sim.radap}_\ell) \cup \mathtt{GS}(\text{obv.sim.radap}_\ell) \subseteq \text{mao.sim.radap}_\ell$. □

6.3 Original publication VI

Proof of Theorem 1: We prove only the equality

$$\mathtt{GS}(\text{prv.sim.stat}, \Phi) \bigcap \mathtt{GS}(\mathtt{ev}_{\text{circ}}) = \mathtt{GS}(\text{SIM.STAT}, \Psi_{\text{tot}}) \bigcap \mathtt{GS}(\mathtt{ev}_{\text{circ}}).$$

The other cases can be proved in similar manner. For simplicity we use notation $\mathtt{ev}$ instead of $\mathtt{ev}_{\text{circ}}$ throughout the proof. Let us first assume that $\mathcal{G}$ is prv.sim.stat secure over side-information function Φ. Our aim is to prove that then $\mathcal{G}$ is also SIM.STAT secure over Ψ_{tot} such that $\Psi_{\text{ev}}(f, x) = (\mathtt{ev}(f, x), \Phi(f))$, $\Psi_{\mathtt{Gb}}(e, d, f) = \Psi_{\text{func}}(f) = \Phi(f)$ and $\Psi_{\text{rest}}(e, d, f, x) = d$ and these values are efficiently computable from $\Psi_{\text{tot}}(e, d, f, x)$.

Let $\mathcal{A}$ be an arbitrary adversary playing the SIM.STAT game against $\mathcal{G}$ with side-information function Ψ_{tot}. We construct an adversary $\mathcal{B}$ for game prv.sim.stat with side-information function Φ. Adversary $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine. This means that adversary $\mathcal{B}$ emulates the SIM.STAT game for adversary $\mathcal{A}$ and uses $\mathcal{A}$'s answers in its own prv.sim.stat game.

Because $\mathcal{G}$ is a prv.sim.stat secure garbling scheme, for every polynomial-time adversary there is a simulator such that the advantage of the adversary is negligible. Let $\mathcal{S}$ be such a simulator for adversary $\mathcal{B}$ in game prv.sim.stat. For the game SIM.STAT we construct a simulator $\mathcal{S}'$ as follows. Simulator $\mathcal{S}'$ takes $(1^k, \Psi_{\text{ev}}(f, x))$ as an input and first computes $\Phi(f)$ and $y = \mathtt{ev}(f, x)$ from $\Psi_{\text{ev}}(f, x)$. Then the simulator $\mathcal{S}'$ calls $\mathcal{S}$ with input $(1^k, y, \Phi(f))$. Simulator $\mathcal{S}'$ is used against adversary $\mathcal{A}$ in game prv.sim.stat.

The prv.sim.stat game starts with procedure `INITIALIZE` in which a value for the challenge bit is chosen uniformly at random. Next, the adversary $\mathcal{B}$ is expected to query procedure `GARBLE` with input (f, x). Instead of choosing (f, x) itself, adversary $\mathcal{B}$ asks adversary $\mathcal{A}$ for input for the `GARBLE` procedure in game SIM.STAT. Adversary $\mathcal{A}$ presumes to play real SIM.STAT game so it sends (f, x) to adversary $\mathcal{B}$. Adversary $\mathcal{B}$ first checks whether $x \in D_f$, in other words whether x is a bit string of length m where m is the number of input wires in circuit f. If $x \notin D_f$ then $\mathcal{B}$ sends $\perp$ to $\mathcal{A}$ and tells that procedure `GARBLE` is ended. Regardless of the result of test $x \in^? D_f$, adversary $\mathcal{B}$ sends (f, x) to its own `GARBLE` procedure. `GARBLE` procedure first checks whether $x \in \{0, 1\}^m$, i.e. whether the argument x is a bit string of length m where m is the number of input wires in circuit f. Note that this test is passed whenever the test

$x \in^? D_f$ in game SIM.STAT is passed. If $x \notin \{0,1\}^m$, then the GARBLE procedure in game prv.sim.stat is ended with $\perp$ which is sent to adversary $\mathcal{B}$. Otherwise, GARBLE prepares (F, X) based on the value of the challenge bit. If $b = 1$ then (F, X, d) is generated by the real garbling algorithms Gb and En and if $b = 0$ then the simulator $\mathcal{S}$ generates (F, X, d). In both cases GARBLE returns (F, X, d) to adversary $\mathcal{B}$. Now, adversary $\mathcal{B}$ computes $\Psi_{\text{tot}}(e, d, f, x)$ as follows: he first computes $Y = \texttt{Ev}(F, X)$ and $y = \texttt{De}(d, Y)$, where $y = \texttt{ev}(f, x)$. Then $\mathcal{B}$ sets $\Psi_{\text{ev}}(f, x) = (y, \Phi(f))$, $\Psi_{\text{func}}(f) = \Psi_{\texttt{Gb}}(e, d, f) = \Phi(f)$ and $\Psi_{\text{rest}}(e, d, f, x) = d$, which now form $\Psi_{\text{tot}}(e, d, f, x)$. Then $\mathcal{B}$ sets $z = \Psi_{\text{tot}}(e, d, f, x)$ and sends (F, X, z) to adversary $\mathcal{A}$. Now, adversary $\mathcal{A}$ sends $b_{\mathcal{A}}$ to $\mathcal{B}$. Value $b_{\mathcal{A}}$ represents $\mathcal{A}$'s candidate for the value of the challenge bit. Adversary $\mathcal{B}$ answers the same as $\mathcal{A}$, i.e. $b_{\mathcal{B}} = b_{\mathcal{A}}$.

Let us now consider the possible outcomes of the game. If $x \notin D_f$ then adversary $\mathcal{A}$ does not receive information about (F, X) or $\Psi_{\text{tot}}(e, d, f, x)$ because the query to GARBLE_ARG returns $\perp$. But in this case also adversary $\mathcal{B}$ gets $\perp$ from its GARBLE procedure (since x is not a bit string of length m). If $x \in D_f$, then adversary $\mathcal{B}$ is gets (F, X, d) from its GARBLE procedure and can use this information to construct z for $\mathcal{A}$. Therefore, adversary $\mathcal{B}$ is able to correctly emulate the SIM.STAT game with the simulator $\mathcal{S}'$ to adversary $\mathcal{A}$. Since the answers of both adversaries are the same, the win probability of adversary $\mathcal{A}$ is the same as the win probability of adversary $\mathcal{B}$. This implies that the advantage of adversary $\mathcal{A}$ is equal to the advantage of adversary $\mathcal{B}$. We assumed that $\mathcal{G}$ is prv.sim.stat secure over Φ, meaning that there is a simulator that makes the advantage of adversary $\mathcal{B}$ negligible in game prv.sim.stat. Let simulator $\mathcal{S}$ be such a simulator. But now the simulator $\mathcal{S}'$ constructed above makes the advantage of adversary $\mathcal{A}$ negligible. Thus, the garbling scheme $\mathcal{G}$ is also SIM.STAT secure. This now completes the proof of the claim

$$\texttt{GS}(\text{prv.sim.stat}, \Phi) \bigcap \texttt{GS}(\texttt{ev}_{\text{circ}}) \subseteq \texttt{GS}(\text{SIM.STAT}, \Psi_{\text{tot}}) \bigcap \texttt{GS}(\texttt{ev}_{\text{circ}}).$$

Conversely, let us assume that $\mathcal{G}$ is SIM.STAT secure over side-information function Ψ_{tot}. Our aim is to prove that $\mathcal{G}$ is prv.sim.stat secure over Φ. The idea of the proof is similar to the earlier part of this proof. Let adversary $\mathcal{B}$ be an arbitrary adversary playing game prv.sim.stat with side-information function Φ against $\mathcal{G}$. We construct an adversary $\mathcal{A}$ for game SIM.STAT with side-information function Ψ_{tot} against $\mathcal{G}$ as follows. Adversary $\mathcal{A}$ emulates game prv.sim.stat for $\mathcal{B}$ and uses $\mathcal{B}$ as a subroutine in the following way.

Because $\mathcal{G}$ is a SIM.STAT secure garbling scheme over Ψ_{tot}, there must be a simulator for any polynomial-time adversary such that the advantage of the adversary is negligible. Let $\mathcal{S}$ be such a simulator in game SIM.STAT for adversary $\mathcal{A}$. In game prv.sim.stat we use the following simulator $\mathcal{S}'$

against adversary $\mathcal{B}$. Simulator $\mathcal{S}'$ takes $(1^k, y, \Phi(f))$ as an input. First, $\mathcal{S}'$ computes $\Psi(f,x) = (\Phi(f), m, y)$. Note that it is assumed that m can always be computed from $\Phi(f)$. Then, $\mathcal{S}'$ calls $\mathcal{S}$ with input $(1^k, \Psi(f,x))$.

The SIM.STAT game starts with the choice of the challenge bit b. Then adversary $\mathcal{A}$ is asked to provide input to `GARBLE` procedure. Instead of choosing (f,x) itself, adversary $\mathcal{A}$ tells adversary $\mathcal{B}$ to provide input to `GARBLE` procedure in game prv.sim.stat. $\mathcal{B}$ presumes to play real prv.sim.stat game, so $\mathcal{B}$ sends (f,x) to $\mathcal{A}$. Now, $\mathcal{A}$ checks whether $x \in \{0,1\}^m$ where m is the number of input wires in circuit f. If $x \notin \{0,1\}^m$, $\mathcal{A}$ sends $\perp$ to $\mathcal{B}$. Regardless of the result of $x \in^? \{0,1\}^m$, adversary $\mathcal{A}$ sends (f,x) to its `GARBLE` procedure. Now, `GARBLE` in game SIM.STAT checks whether $x \in D_f$, i.e. whether x is a bit string of length m. The test fails exactly when $x \notin \{0,1\}^m$. In this case, adversary $\mathcal{A}$ gets $\perp$ from `GARBLE` . Otherwise, `GARBLE` returns (F,X,z) based on the value of the challenge bit b. If $b = 1$ then (F,X,z) is created with actual algorithms `Gb`, `En` and Ψ_{tot} and if $b = 0$ then (F,X,z) is generated by a simulator $\mathcal{S}$. Now, adversary $\mathcal{A}$ needs to find out d from z. This is possible, since adversary $\mathcal{A}$ is able to compute $\Psi_{\text{rest}}(e,d,f,x) = d$ from $z = \Psi_{\text{tot}}(e,d,f,x)$. Adversary $\mathcal{A}$ now sends (F,X,d) to adversary $\mathcal{B}$. Adversary $\mathcal{B}$ sends its answer $b_{\mathcal{B}}$ to $\mathcal{A}$. Adversary $\mathcal{A}$ answers the same as $\mathcal{B}$, i.e. $b_{\mathcal{A}} = b_{\mathcal{B}}$.

Let us now analyze the possible outcomes of the game. If $x \notin \{0,1\}^m$ then adversary $\mathcal{B}$ does not receive information about (F,X,d) because the query to `GARBLE_ARG` yields $\perp$. In this case, also adversary $\mathcal{A}$ gets $\perp$ since $x \notin D_f$ (x is not a bit string of length m). If $x \in \{0,1\}^m$ then adversary $\mathcal{A}$ is allowed to find out the information that adversary $\mathcal{B}$ needs in its game, namely (F,X,d). Since the answer of adversary $\mathcal{B}$ is the same as the answer of adversary $\mathcal{A}$, the win probability of $\mathcal{B}$ in game prv.sim.stat is the same as the win probability of $\mathcal{A}$ in game SIM.STAT. This in turn implies that $\mathbf{Adv}_{\mathcal{B}} = \mathbf{Adv}_{\mathcal{A}}$. The simulator $\mathcal{S}$ is such that the advantage of adversary $\mathcal{A}$ is negligible. But now simulator $\mathcal{S}'$ in game prv.sim.stat makes the advantage of adversary $\mathcal{B}$ equal to $\mathbf{Adv}_{\mathcal{A}}$, i.e. negligible. This proves that

$$\mathtt{GS}(SIM.STAT, \Psi_{\text{tot}}) \bigcap \mathtt{GS}(\mathtt{ev}_{\text{circ}}) \subseteq \mathtt{GS}(prv.sim.stat, \Phi) \bigcap \mathtt{GS}(\mathtt{ev}_{\text{circ}}).$$

This concludes the proof. □

Proof of Theorem 2: We prove only the equality

$$\mathtt{GS}(\text{obv.ind.adap}_\ell, \Phi) \bigcap \mathtt{GS}(\mathtt{ev}_{\text{circ}}) = \mathtt{GS}(\text{IND.ADAP}_\ell, \Psi_{\text{tot}}) \bigcap \mathtt{GS}(\mathtt{ev}_{\text{circ}}).$$

The other cases can be proven in similar manner. For simplicity we use notation `ev` instead of $\mathtt{ev}_{\text{circ}}$ throughout the proof. Let us first assume that $\mathcal{G}$ is obv.ind.adap_ℓ secure over side-information function Φ. Our aim is to prove that then $\mathcal{G}$ is also IND.ADAP_ℓ secure over Ψ_{tot}. From Ψ_{tot}

we can efficiently compute $\Psi_{\mathtt{Gb}}(e,d,f) = \Psi_{\mathrm{ev}}(f,x) = \Psi_{\mathrm{func}}(f) = \Phi(f)$ and $\Psi_{\mathrm{rest}}(e,d,f,x) = \varepsilon$.

Let $\mathcal{A}$ be an arbitrary adversary playing the IND.ADAP$_\ell$ game against $\mathcal{G}$ with side-information function Ψ_{tot}. We construct an adversary $\mathcal{B}$ for game obv.ind.adap$_\ell$ with side-information function Φ. Adversary $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine. This means that adversary $\mathcal{B}$ emulates the IND.ADAP$_\ell$ game for adversary $\mathcal{A}$ and uses $\mathcal{A}$'s answers in its own obv.ind.adap$_\ell$ game.

First, the obv.ind.adap$_\ell$ game starts by the choice of the challenge bit b. Then, adversary $\mathcal{B}$ is expected to provide two functions as input to `GARBLE_FUNC` procedure. Instead of choosing the functions itself, adversary $\mathcal{B}$ starts IND.ADAP$_\ell$ game with adversary $\mathcal{A}$. Since adversary $\mathcal{A}$ presumes to play a real IND.ADAP$_\ell$ game, it sends two functions, f_0 and f_1, to adversary $\mathcal{B}$. Adversary $\mathcal{B}$ sends the same functions, f_0 and f_1, to its game obv.ind.adap$_\ell$. The game first checks whether $\Phi(f_0) = \Phi(f_1)$ holds. If this is not the case, then the game sends $\perp$ to adversary $\mathcal{B}$. This is the correct behavior because $\Psi_{\mathrm{func}}(f_0) \neq \Psi_{\mathrm{func}}(f_1)$, following from the fact that $\Psi_{\mathrm{func}}(f_0) = \Phi(f_0)$, $\Psi_{\mathrm{func}}(f_1) = \Phi(f_1)$ and $\Phi(f_0) \neq \Phi(f_1)$. In this case, adversary $\mathcal{B}$ sends $\perp$ to $\mathcal{A}$.

If f_0 and f_1 pass the test $\Phi(f_0) = \Phi(f_1)$ in the obv.ind.adap$_\ell$ game, then the game garbles function f_b, based on the choice of the challenge bit. The garbled function F is sent to adversary $\mathcal{B}$. Now adversary $\mathcal{B}$ sends $(F, \Phi(f_b))$ to adversary $\mathcal{A}$: in the IND.ADAP$_\ell$ game, an adversary should get (F,w) where $w = \Psi_{\mathtt{Gb}}(e,d,f_b) = \Phi(f_b)$. After getting (F,w), adversary $\mathcal{A}$ chooses two arguments, x_0 and x_1, and sends them to adversary $\mathcal{B}$ to get output from procedure `GARBLE_ARG`. Adversary $\mathcal{B}$ sends (x_0,x_1) to its obv.ind.adap$_\ell$ game.

The game first checks again, whether the side-information $\Phi(f_0)$ coincides with the side-information $\Phi(f_1)$. If this is not the case, then the obv.ind.adap$_\ell$ game sends $\perp$ to $\mathcal{B}$. Adversary $\mathcal{B}$ sends $\perp$ to $\mathcal{A}$. If the first test in obv.ind.adap$_\ell$ game is passed, then the game tests whether the length of both arguments is suitable for functions f_0 and f_1. If this is not the case, then the game obv.ind.adap$_\ell$ sends $\perp$ to $\mathcal{B}$. Adversary $\mathcal{B}$ sends $\perp$ to $\mathcal{A}$.

We show next that adversary $\mathcal{B}$ is able to correctly emulate the `GARBLE_ARG` procedure of game IND.ADAP$_\ell$ to adversary $\mathcal{A}$. Adversary $\mathcal{B}$ receives $\perp$ from its game if the side-information Φ of functions f_0 and f_1 are different or if the arguments x_0 and x_1 are not bit strings of correct length. In this case, also adversary $\mathcal{A}$ receives $\perp$. The meaning of $\perp$ to adversary $\mathcal{A}$ is that either the arguments have not been of correct form (i.e. bit strings of correct length) or that the side-information Ψ_{ev} is different for functions f_0 and f_1. This is the correct behavior since $\Psi_{\mathrm{ev}}(f,x) = \Phi(f)$.

If both tests are passed in $\mathcal{B}$'s obv.ind.adap$_\ell$ game, then adversary $\mathcal{A}$ returns its answer $b_{\mathcal{A}}$ to adversary $\mathcal{B}$. Adversary $\mathcal{B}$'s answer is the same as $\mathcal{A}$'s answer. Let us now analyze the winning chance of both adversaries.

Adversary $\mathcal{B}$ is able to correctly emulate the game IND.ADAP$_\ell$ to adversary $\mathcal{A}$. The answer of both adversaries is the same. Furthermore, the answer b is the same in both games. Therefore, the probability of winning the game is the same for both adversaries $\mathcal{A}$ and $\mathcal{B}$.

Since we assumed that $\mathcal{G}$ is obv.ind.adap$_\ell$ secure over Φ, the advantage of any $\mathcal{PT}$ adversary is negligible. Especially, the advantage of adversary $\mathcal{B}$ is negligible. We assumed that the adversary $\mathcal{A}$ is an arbitrary $\mathcal{PT}$ adversary playing the game IND.ADAP$_\ell$. Now we have that the advantage of an arbitrary adversary $\mathcal{A}$ in game IND.ADAP$_\ell$ is negligible, proving that $\mathcal{G}$ is IND.ADAP$_\ell$ secure over the chosen side-information function $\Psi_{\text{tot}}(e, d, f, x)$. This concludes the proof of the claim

$$\texttt{GS}(\text{obv.ind.adap}, \Phi) \bigcap \texttt{GS}(\texttt{ev}_{\text{circ}}) \subseteq \texttt{GS}(\text{IND.ADAP}_\ell, \Psi_{\text{tot}}) \bigcap \texttt{GS}(\texttt{ev}_{\text{circ}}).$$

Conversely, let $\mathcal{G}$ be an IND.ADAP$_\ell$ secure garbling scheme over side-information function Ψ_{tot}. Our aim is to prove that $\mathcal{G}$ is obv.ind.adap$_\ell$ secure over Φ. The idea of the proof is similar to the earlier part of this proof. Let adversary $\mathcal{B}$ be an arbitrary adversary playing game obv.ind.adap$_\ell$ with side-information function Φ against $\mathcal{G}$. We construct an adversary $\mathcal{A}$ for game IND.ADAP$_\ell$ with side-information function Ψ_{tot} against $\mathcal{G}$ as follows. Adversary $\mathcal{A}$ emulates game obv.ind.adap$_\ell$ for $\mathcal{B}$ and uses $\mathcal{B}$ as a subroutine in the following way.

The IND.ADAP$_\ell$ game starts with `INITIALIZE` procedure in which the challenge bit b is chosen uniformly at random. In the next step, adversary $\mathcal{A}$ is expected to give two functions as input to procedure `GARBLE_FUNC`. Instead of choosing f_0, f_1 itself, adversary $\mathcal{A}$ tells adversary $\mathcal{B}$ that game obv.ind.adap$_\ell$ game has been started and an input to `GARBLE_FUNC` is expected. Since $\mathcal{B}$ presumes to play the actual, not emulated, obv.ind.adap$_\ell$ game, it chooses f_0, f_1 and sends them to $\mathcal{A}$. Adversary $\mathcal{A}$ sends the same functions f_0, f_1 to the `GARBLE_FUNC` in its IND.ADAP$_\ell$ game. The procedure in IND.ADAP$_\ell$ game first tests whether $\Psi_{\text{func}}(f_0) = \Psi_{\text{func}}(f_1)$ is satisfied. If this is not the case, then $\perp$ is sent to adversary $\mathcal{A}$. Adversary $\mathcal{A}$ sends $\perp$ to $\mathcal{B}$, to indicate that $\Phi(f_0) \neq \Phi(f_1)$ is not satisfied in obv.ind.adap$_\ell$ game. This is the correct behavior in the emulated game because $\Psi_{\text{func}}(f) = \Phi(f)$.

If the test $\Psi_{\text{func}}(f_0) = \Psi_{\text{func}}(f_1)$ is passed, then `GARBLE_FUNC` in IND.ADAP$_\ell$ game prepares a garbled function F based on the choice of the challenge bit b: $F = \texttt{Gb}(1^k, f_b)$. Procedure `GARBLE_FUNC` also computes $w = \Psi_{\texttt{Gb}}(e, d, f_b) = \Phi(f_b)$. Finally, `GARBLE_FUNC` sends (F, w) to $\mathcal{A}$. Adversary $\mathcal{A}$ now extracts F from (F, w) and sends it to adversary $\mathcal{B}$.

Then, adversary $\mathcal{B}$ starts its calls to procedure `GARBLE_ARG` by sending two arguments, x_0 and x_1, to adversary $\mathcal{A}$. Adversary $\mathcal{A}$ sends the same arguments to `GARBLE_ARG` in its IND.ADAP$_\ell$ game. The `GARBLE_ARG` first checks whether the arguments belong to the right domain, i.e. whether they

are bit strings of correct length. If this is not the case, GARBLE_ARG sends $\perp$ to $\mathcal{A}$. Adversary $\mathcal{A}$ sends $\perp$ to $\mathcal{B}$. If the arguments are bit strings of correct length, then the IND.ADAP$_\ell$ game performs the next test $\Psi_{\text{ev}}(f_0, x_0) =^? \Psi_{\text{ev}}(f_1, x_1)$. If this test fails, then the game sends $\perp$ to $\mathcal{A}$. Adversary $\mathcal{A}$ sends $\perp$ to $\mathcal{B}$.

We show next that adversary $\mathcal{A}$ is able to correctly emulate the test in obv.ind.adap$_\ell$ game. In the obv.ind.adap$_\ell$ game, there are two tests. First, the game tests whether the side-information of function f_0 is the same as the side-information of function f_1. The result of the second test tells whether the arguments x_0 and x_1 are bit strings of correct length. If either of the tests in IND.ADAP$_\ell$ game fails, then adversary $\mathcal{A}$ gets $\perp$ and consequently also adversary $\mathcal{B}$ gets $\perp$. If $\mathcal{A}$ gets $\perp$ in its game, then it means that either the arguments are not of correct form or that the functions have different partial side-information Ψ_{ev}. Because $\Psi_{\text{ev}}(f, x) = \Phi(f)$, for adversary $\mathcal{B}$ playing the emulated game, getting $\perp$ in these cases is the correct behavior: either the arguments are not of correct form or the functions have different side-information.

If both tests are passed in IND.ADAP$_\ell$ game then GARBLE_ARG computes the garbled argument $X = \mathtt{En}(e, x_b)$ and $z = \Psi_{\text{tot}}(e, d, f_b, x_b)$. Then, (X, z) is sent to adversary $\mathcal{A}$. Adversary $\mathcal{A}$ extracts X from (X, z) and sends it to $\mathcal{B}$. Adversary $\mathcal{B}$ may now provide its answer $b_{\mathcal{B}}$ to $\mathcal{A}$ or make a new GARBLE_ARG query. Both adversaries make the same amount of GARBLE_ARG queries, both at most ℓ of them.

Finally, $\mathcal{B}$ answers $b_{\mathcal{B}}$ to $\mathcal{A}$, which now uses this answer also as its own answer (i.e. $b_{\mathcal{A}} = b_{\mathcal{B}}$). Let us now analyze the probability that $b_{\mathcal{B}} = b_{\mathcal{A}} = b$ and both adversaries win their games. First of all, adversary $\mathcal{A}$ is able to correctly emulate the obv.ind.adap$_\ell$ game to adversary $\mathcal{B}$ (with the same value of b). Secondly, both adversaries answer the same bit in their games. Therefore, both adversaries have exactly the same chance to win their game. This yields that the advantage of both adversaries in their own games is also the same. Since we assumed that $\mathcal{G}$ is IND.ADAP$_\ell$ secure over Ψ_{tot}, the advantage of adversary $\mathcal{A}$ is negligible. Then, also the advantage of $\mathcal{B}$ is negligible, which now completes the proof. □

Proof of Theorem 3:

We prove the claim in case

$$\mathtt{GS}(\text{mao.sim.adap}_\ell, \Phi) = \mathtt{GS}(\text{SIM.ADAP}_\ell, \Psi_{\text{tot}}).$$

The other cases are proven in similar manner.

Consider first the claim $\mathtt{GS}(\text{mao.sim.adap}_\ell, \Phi) \subseteq \mathtt{GS}(\text{SIM.ADAP}_\ell, \Psi_{\text{tot}})$. To prove the claim, let $\mathcal{G}$ be a mao.sim.adap$_\ell$ secure garbling scheme over side-information function $\Phi(f)$. Our aim is to prove that $\mathcal{G}$ is also SIM.ADAP$_\ell$ secure over side-information function $\Psi_{\text{tot}}(e, d, f, x)$.

Let $\mathcal{A}$ be an arbitrary adversary playing the game SIM.ADAP$_\ell$ game against garbling scheme $\mathcal{G}$. We construct an adversary $\mathcal{B}$ who plays the game MaoSimAdap$_\ell$ against $\mathcal{G}$. Adversary $\mathcal{B}$ uses adversary $\mathcal{A}$ as a subroutine by emulating game SIM.ADAP$_\ell$ to adversary $\mathcal{A}$.

Let the simulator in the game MaoSimAdap$_\ell$ be $\mathcal{S}$. The simulator $\mathcal{S}$ takes $(1^k, \Phi(f))$ as input when it is called in procedure `GARBLE_FUNC` and y when the simulator is called in procedure `GARBLE_ARG`. Using the simulator $\mathcal{S}$ we construct another simulator $\mathcal{S}'$ for the game SIM.ADAP$_\ell$. In procedure `GARBLE_FUNC`, the simulator $\mathcal{S}'$ is called with input $(1^k, \Psi_{\text{func}}(f))$. Because $\Psi_{\text{func}}(f) = \Phi(f)$ according to our assumption, simulator $\mathcal{S}'$ can call simulator $\mathcal{S}$ with the same input $(1^k, \Psi_{\text{func}}(f))$. In procedure `GARBLE_ARG`, the simulator $\mathcal{S}'$ takes input $\texttt{ev}(f, x)$. Simulator $\mathcal{S}'$ now calls simulator $\mathcal{S}$ with input $(\texttt{ev}(f, x), \Phi(f))$ which is $\Psi_{\text{ev}}(f, x)$ according to our assumptions.

First, the game MaoSimAdap$_\ell$ tells adversary $\mathcal{B}$ to start its game and to provide a function to procedure `GARBLE_FUNC`. Adversary $\mathcal{B}$ does not choose f itself; instead it asks adversary $\mathcal{A}$ to start SIM.ADAP$_\ell$ game. Adversary $\mathcal{A}$ presumes to play a real SIM.ADAP$_\ell$ game. Therefore, he chooses a function f and sends it to adversary $\mathcal{B}$. Adversary $\mathcal{B}$ sends f to his `GARBLE_FUNC` procedure.

Based on the challenge bit chosen in the `INITIALIZE` procedure of game MaoSimAdap$_\ell$, `GARBLE_FUNC` either uses the algorithm `Gb` or the simulator $\mathcal{S}'$ to compute the garbled function F. The garbled function F is sent to adversary $\mathcal{B}$ who sends F to adversary $\mathcal{A}$.

Then, adversary $\mathcal{A}$ sends his argument x to adversary $\mathcal{B}$. Adversary $\mathcal{B}$ sends argument x to his `GARBLE_ARG` procedure. The procedure `GARBLE_ARG` checks first whether the argument is of correct length, i.e. whether x is a bit string of length m. If x is not a bit string of length m then `GARBLE_ARG` procedure is exited with $\perp$ as the return value to adversary $\mathcal{B}$. In this case, adversary $\mathcal{B}$ sends $\perp$ to $\mathcal{A}$. If x is a bit string of length m, then the procedure `GARBLE_ARG` computes the garbled argument using either the encryption algorithm `En` or the simulator $\mathcal{S}'$ based on the value of the challenge bit.

Queries to obtain garbled arguments can be performed at most ℓ times. Then adversary $\mathcal{A}$ gives his answer $b_{\mathcal{A}}$ to adversary $\mathcal{B}$. Adversary $\mathcal{B}$ uses $\mathcal{A}'s$ answer as his own answer in the MaoSimAdap$_\ell$ game.

Let us now consider the possible outcomes of the game. Adversary $\mathcal{B}$ receives $\perp$ in his game exactly in the same cases as $\mathcal{A}$ gets $\perp$ in his game. Therefore, adversary $\mathcal{B}$ is able to correctly emulate the SIM.ADAP$_\ell$ game to $\mathcal{A}$. Furthermore, the answers of $\mathcal{A}$ and $\mathcal{B}$ in their games are equal. The answer of $\mathcal{A}$ and $\mathcal{B}$ are both correct or incorrect at the same time since the correct answer is b in both games.

To conclude, both adversaries have an equal probability to win their games. This implies that the advantage of both adversaries in their games are equal. Since $\mathcal{G}$ is mao.sim.adap$_\ell$ secure over Φ there is a simulator such

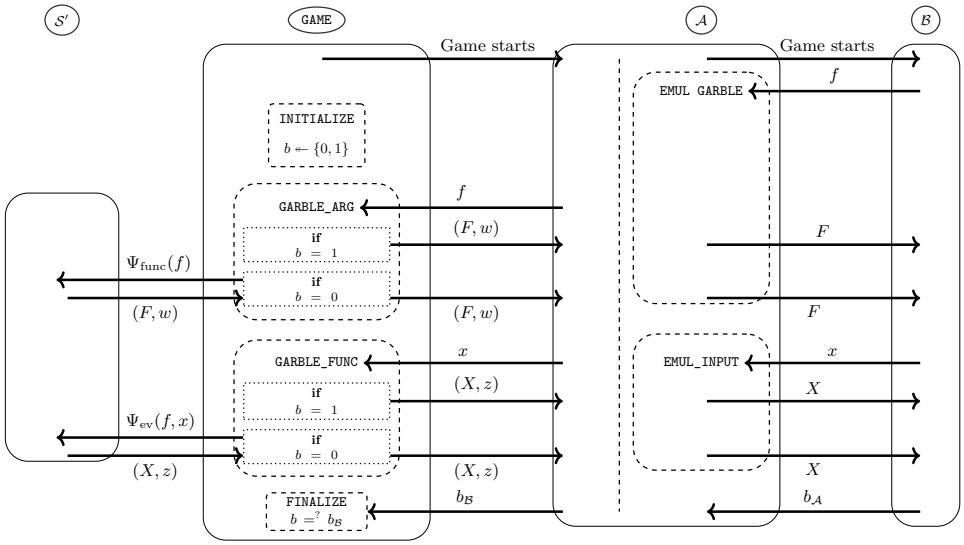


Figure 6.2: In the above figure, $\mathcal{A}$ is an adversary playing the game SIM.ADAP$_\ell$ denoted by GAME in the diagram. $\mathcal{S}'$ is the simulator in this game. $\mathcal{B}$ is an adversary presuming to play game MaoSimAdap$_\ell$. Actually adversary $\mathcal{A}$ uses $\mathcal{B}$ as a subroutine by trying to emulate the game MaoSimAdap$_\ell$.

that advantage of $\mathcal{B}$ is negligible. Let simulator $\mathcal{S}$ be such a simulator. We have constructed simulator $\mathcal{S}'$ for game SIM.ADAP$_\ell$ which uses $\mathcal{S}$ in such way that the advantage of $\mathcal{A}$ in game SIM.ADAP$_\ell$ is negligible. This now shows that there is a simulator $\mathcal{S}'$ such that the advantage of an arbitrary adversary $\mathcal{A}$ is negligible. Thus, $\mathcal{G}$ is SIM.ADAP$_\ell$ secure over Ψ_{tot} which now concludes the proof of the claim $\texttt{GS}(\text{mao.xxx.yyy}, \Phi) \subseteq \texttt{GS}(\text{XXX.YYY}, \Psi_{\text{tot}})$.

Conversely, let us consider the inclusion $\texttt{GS}(\text{SIM.ADAP}_\ell, \Psi_{\text{tot}}) \subseteq \texttt{GS}(\text{mao.sim.adap}_\ell, \Phi)$. Let $\mathcal{G}$ be a garbling scheme in the class $\texttt{GS}(\text{SIM.ADAP}_\ell, \Psi_{\text{tot}})$. Our aim is to prove that $\mathcal{G}$ belongs also to the class $\texttt{GS}(\text{mao.sim.adap}_\ell, \Phi)$.

Let $\mathcal{B}$ be an arbitrary adversary playing the game MaoSimAdap$_\ell$ game against garbling scheme $\mathcal{G}$. We construct an adversary $\mathcal{A}$ playing the game SIM.ADAP$_\ell$ who uses adversary $\mathcal{B}$ as subroutine. Figure 6.2 illustrates how the games are played when $\mathcal{A}$ plays its own game and emulates $\mathcal{B}$'s game.

Let $\mathcal{S}'$ be the simulator in game SIM.ADAP$_\ell$. The simulator $\mathcal{S}'$ takes $(1^k, \Psi_{\text{func}}(f))$ as input in GARBLE_FUNC procedure and $\Psi_{\text{ev}}(f, x)$ in GARBLE_ARG procedure. We construct a simulator $\mathcal{S}$ for game MaoSimAdap$_\ell$. In the game MaoSimAdap$_\ell$, the simulator $\mathcal{S}$ takes $(1^k, \Phi(f))$ as input in INPUT procedure. Simulator $\mathcal{S}$ calls simulator $\mathcal{S}'$ with input $(1^k, \Phi(f))$. Input $(1^k, \Phi(f))$ is a valid input to simulator $\mathcal{S}'$ because we assumed that $\Psi_{\text{func}}(f) = \Phi(f)$. In procedure GARBLE, the simulator $\mathcal{S}$ takes y as input

and makes call $\mathcal{S}'(y, \Phi(f))$. Input $(y, \Phi(f))$ is a valid input to simulator $\mathcal{S}'$ because we assume that $\Psi_{\text{ev}}(f, x) = (\texttt{ev}(f, x), \Phi(f))$.

First, the game SIM.ADAP$_\ell$ tells adversary $\mathcal{A}$ to start its game and to provide a function to procedure `GARBLE_FUNC`. Adversary $\mathcal{A}$ does not choose f itself; instead it asks adversary $\mathcal{B}$ to start MaoSimAdap$_\ell$ game. Adversary $\mathcal{B}$ presumes to play a real MaoSimAdap$_\ell$ game. Therefore, he chooses a function f and sends it to adversary $\mathcal{A}$. Adversary $\mathcal{A}$ sends f to his `GARBLE_FUNC` procedure.

Based on the challenge bit chosen in the `INITIALIZE` procedure of game SIM.ADAP$_\ell$, `GARBLE_FUNC` either uses the algorithm `Gb` or the simulator $\mathcal{S}$ to compute the garbled function F. The output of `GARBLE_FUNC` procedure, (F, w) is sent to adversary $\mathcal{A}$ who then sends F to adversary $\mathcal{B}$.

Then, adversary $\mathcal{B}$ sends his argument x to adversary $\mathcal{A}$. Adversary $\mathcal{A}$ sends argument x to his `GARBLE_ARG` procedure. The procedure `GARBLE_ARG` checks first whether the argument is of correct length, i.e. whether x is a bit string of length m. If x is not a bit string of length m then `GARBLE_ARG` procedure is exited with $\perp$ as the return value to adversary $\mathcal{A}$. In this case, adversary $\mathcal{A}$ sends $\perp$ to $\mathcal{B}$. If x is a bit string of length m, then the procedure `GARBLE_ARG` computes the garbled argument using either the encryption algorithm `En` or the simulator $\mathcal{S}$ based on the value of the challenge bit.

Queries to obtain garbled arguments can be performed at most ℓ times. Then adversary $\mathcal{B}$ gives his answer $b_{\mathcal{B}}$ to adversary $\mathcal{A}$. Adversary $\mathcal{A}$ uses $\mathcal{B}'s$ answer as his own answer in the SIM.ADAP$_\ell$ game.

The above construction shows that adversary $\mathcal{A}$ is able to correctly emulate mao.sim.adap$_\ell$ game with simulator $\mathcal{S}$ to adversary $\mathcal{B}$. Now consider the win probabilities of both adversaries.

Consider first the case $b = 0$. In this case, the simulator $\mathcal{S}'$ has been used for constructing F and X. The simulator $\mathcal{S}$ in $\mathcal{B}$'s game and the simulator $\mathcal{S}'$ in $\mathcal{A}$'s game behave in exactly similar manner: Both simulators use the same inputs and $\mathcal{S}'$ is constructed using simulator $\mathcal{S}$. Outputs F and X generated by $\mathcal{S}'$ are exactly the same as those generated by $\mathcal{S}$. In addition, both adversaries have the same answer in their games, so the winning probability is the same for both adversaries.

Then consider the case $b = 1$. In this case, the actual garbling algorithms `Gb` and `En` are used to construct F and X. In this case, adversary $\mathcal{B}$ is able to correctly emulate mao.sim.adap$_\ell$ game to adversary $\mathcal{A}$. Both adversaries have the same answer in their games, so both adversaries have the same winning probability.

In both cases $b = 0$ and $b = 1$, the winning probability of adversary $\mathcal{B}$ is the same as the winning probability of $\mathcal{A}$. In terms of advantages, the advantage of adversary $\mathcal{B}$ is equal to the advantage of adversary $\mathcal{A}$. Since $\mathcal{G}$ is SIM.ADAP$_\ell$ secure over Ψ_{tot} there is a simulator such that the advantage of $\mathcal{A}$ is negligible. Let simulator $\mathcal{S}'$ be such a simulator. Using simulator $\mathcal{S}'$

we have constructed a simulator $\mathcal{S}$ such that the advantage of an arbitrary adversary $\mathcal{B}$ is negligible. This proves that $\mathcal{G}$ is mao.sim.adap$_\ell$ secure over Φ.

This now concludes the proof of the claim $\mathtt{GS}(\mathrm{XXX.YYY}, \Phi) = \mathtt{GS}(\mathrm{mao.xxx.yyy}, \Psi_{\mathrm{tot}})$. $\square$

Turku
Centre *for*
Computer
Science

Joukahaisenkatu 3-5 B, 20520 Turku, Finland | www.tucs.fi

University of Turku
- Department of Information Technology
- Department of Mathematics

Åbo Akademi University
- Department of Information Technologies

Turku School of Economics
- Institute of Information Systems Sciences

ISBN 978-952-12-3515-3
ISSN 1239-1883