TUCS

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 978-952-12-3516-0
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

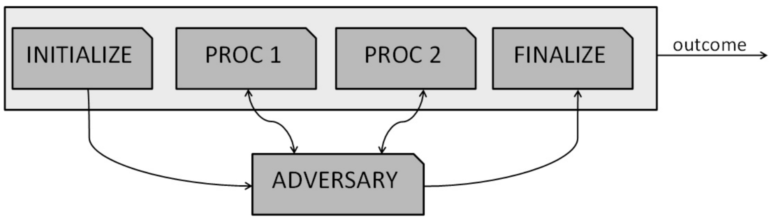


Figure 3.1: A game consists of codes for `INITIALIZE`, `FINALIZE` and other named procedures. An adversary playing the game is also treated as code which can interact with the other named procedures.

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

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

Garbled truth table (an example)

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

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).

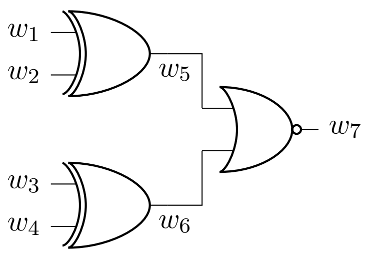


Figure 3.4: The circuit for testing the equality consists of two XOR gates and one NOR gate.

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 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.

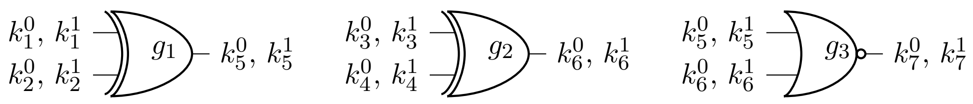


Figure 3.5: Each wire w_i is associated with two keys, k_i^0 and k_i^1.

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.

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

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

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

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 $e^3_{g_1}$, returning k^1_5. This is the case, because $e^3_{g_1} = e^{10}_{g_1} = \mathtt{En}_{k^1_1}(\mathtt{En}_{k^0_2}(k^0_5))$. Other three decryptions yield $\perp$. In similar way, Alice obtains k^1_6 by decrypting the values $e^0_{g_2}$, $e^1_{g_2}$, $e^2_{g_2}$ and $e^3_{g_2}$ with keys k^1_3 and k^0_4. In this case, the output k^1_6 is obtained from decrypting value $e^0_{g_2}$ whereas the other three decryptions yield $\perp$. As the final step, Alice finds out k^0_7 by decrypting values $e^0_{g_3}$, $e^1_{g_3}$, $e^2_{g_3}$ and $e^3_{g_3}$ with keys k^1_5 and k^1_6. In this case, k^0_7 is obtained from decrypting $e^3_{g_3}$; the other three decryptions again yield $\perp$. Then Alice sends k^0_7 to Bob, who then reveals that key k^0_7 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

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

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 $\texttt{AND}(1, 0) = 0$ and $\lambda_3 = \texttt{AND}(1, 0) \oplus \pi_3 = \pi_3$.

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 $x = x_1 \ldots x_n \in \{0,1\}^n$ and $x' = x'_1 \ldots x'_n \in$*

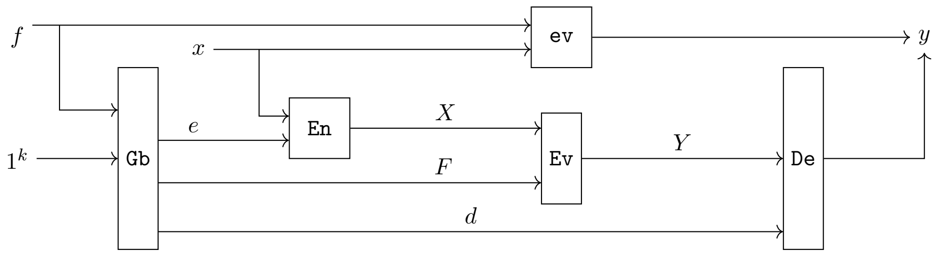


Figure 3.8: Idea behind garbling. The diagram shows that the final value obtained via garbling must coincide with the final value obtained by direct evaluation, i.e. $\texttt{ev}(f, x) = y = \texttt{De}(d, Y)$ where F is the garbled function, $Y = \texttt{Ev}(F, X)$ is the garbled value and $X = \texttt{En}(e,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

$\{0,1\}^n$ be any two arguments for f. Let **En** *be a non-deterministic algorithm and let r denote the explicit randomness value used in computation of* **En**. *We say that garbling scheme $\mathcal{G}$ is* projective *if the two garbled arguments $X = \textbf{En}(e, x, r)$ and $X' = \textbf{En}(e, x', r)$ 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 En of the garbling scheme. In this example, we provide an encryption algorithm that has the projectivity property. For constructing the encryption algorithm En, we use 128-bit AES. Let e be an encryption key generated by the garbling algorithm for encrypting arguments with 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 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 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 $\Phi(f)$.

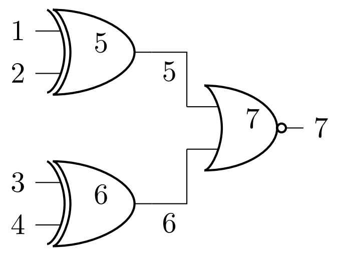
The concept of side-information provides another point-of-view to the leaked information, compared with the security notions privacy, obliviousness and matchability-only. The security notions tell which level of privacy a garbling scheme provides for f, x and y. For example, a garbling scheme achieving 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 `Inputs` $= \{1, \ldots, m\}$. The set of all wires in the circuit is `Wires` $= \{1, \dots, m+q\}$. The set of output wires is `Outputs` $= \{m+q-n+1, \dots, m+q\}$. Finally, the set of gates is `Gates` $= \{m+1, \dots, m+q\}$. 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 : `Gates` $\to$ `Wires`\\`Outputs`. The element B is a function that identifies the second incoming wire of a gate, i.e. B : `Gates` $\to$ `Wires`\\`Outputs`. Finally, the element G is a function that tells the functionality of each gate, i.e. G : `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 $\Phi_{\text{topo}}(f)$

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

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 $\Phi_{\text{topo}}(f_C)$.

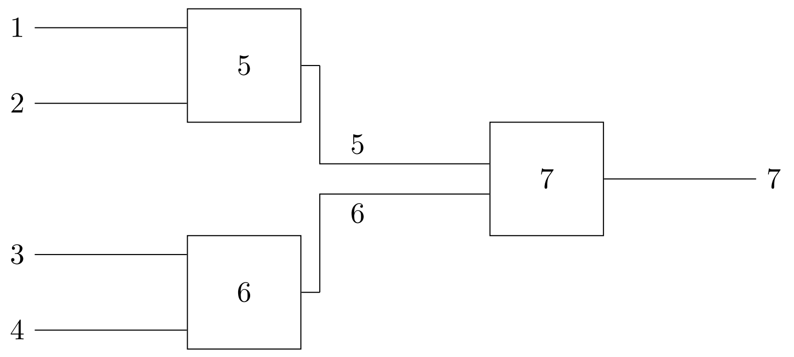


Figure 3.10: Side-information function Φ_{topo} leaks the topology of circuit from example 3.4. The functionality of the three gates is not leaked.

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* $\in \{\mathrm{Prv}, \mathrm{Obv}, \mathrm{Mao}\}$*, Yyy* $\in \{\mathrm{Sim}, \mathrm{Ind}\}$ *and Zzz* $\in \{\mathrm{Stat}, \mathrm{Adap}_\ell, \mathrm{Padap}_\ell, \mathrm{Radap}_\ell\}$*. 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 $\Pr\left[\mathcal{A} \text{ wins}\right]$ *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$_\ell$, Padap$_\ell$, Radap$_\ell$}. *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$_\ell$, Padap$_\ell$, Radap$_\ell$}. *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* $\in$ {prv, obv, mao}, *yyy* $\in$ {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 $(\Phi, \texttt{ev})$. We say that $(\Phi, \texttt{ev})$ is efficiently invertible, if there is an efficient algorithm which takes $\Phi(f)$ and $y = \texttt{ev}(f, x)$ as input and outputs (f', x') such that $\Phi(f') = \Phi(f)$ and $\texttt{ev}(f', x') = y = \texttt{ev}(f, x)$. In [18] it is shown that $(\Phi_{\text{topo}}, \texttt{ev}_{\text{circ}})$ and $(\Phi_{\text{size}}, \texttt{ev}_{\text{circ}})$ are efficiently invertible, whereas $(\Phi_{\text{circ}}, \texttt{ev}_{\text{circ}})$ is not. Here, $\texttt{ev}_{\text{circ}}$ denotes the usual evaluation algorithm for circuits. For justifying why $(\Phi_{\text{circ}}, \texttt{ev}_{\text{circ}})$ 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.

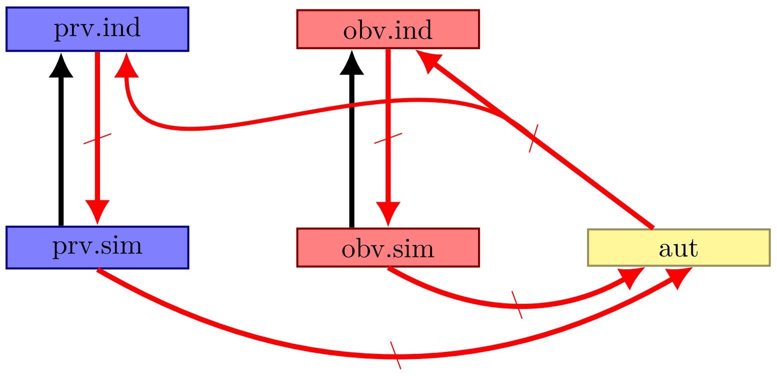


Figure 3.11: Relations between the classes of garbling schemes. The figure is adapted from [18, p.786].

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.

Contexts in which garbling is needed

- Cloud computing
- Distributed computing

Applications requiring privacy-preserving techniques

- Statistical analysis
- Data mining
- Machine learning
- Database query
- Search engines
- Remote monitoring
  - Surveillance/monitoring
  - Health monitoring
- Computer intrusion detection and virus protection

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.

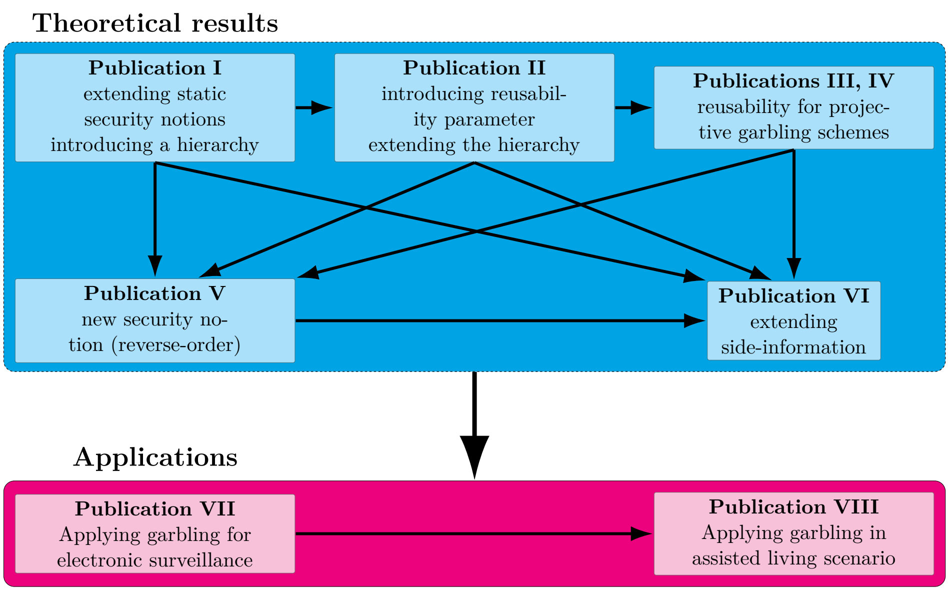

Theoretical results
Publication I
extending static security notions introducing a hierarchy
Publication II
introducing reusability parameter extending the hierarchy
Publications III, IV
reusability for projective garbling schemes
Publication V
new security notion (reverse-order)
Publication VI
extending side-information
Applications
Publication VII
Applying garbling for electronic surveillance
Publication VIII
Applying garbling in assisted living scenario

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 $F = \texttt{Gb}(f, 1^k)$, 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 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 ev called bit- and componentwise efficient invertibility of $(\Phi, \texttt{ev})$. The existence of a bit-

and componentwise efficient $(\Phi, \texttt{ev})$-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 $(\texttt{Gb}, \texttt{En}, \texttt{Ev}, \texttt{De}, \texttt{ev})$, we define a garbling scheme as a 6-tuple $(\texttt{KeyGen}, \texttt{Ga}, \texttt{En}, \texttt{Ev}, \texttt{De}, \texttt{ev})$. 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.

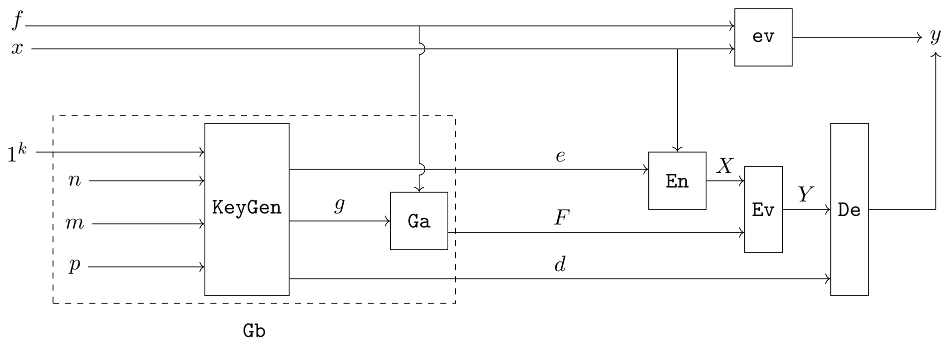


Figure 4.2: Generalized model of a garbling scheme. Compared to the definition in [18], the diagram is the same except of the garbling algorithm `Gb`, which in our model is split to two separate algorithms `KeyGen` and `Ga`.

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 $\texttt{ev}(f, x)$ 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.

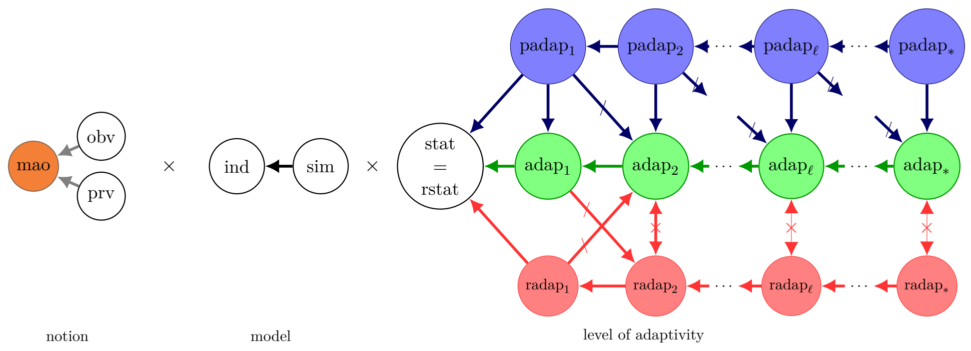


Figure 5.1: Hierarchy of garbling scheme classes represented as a Cartesian product of graphs. Lines $\rightarrow$ show the inclusions between security classes and lines $\nrightarrow$ show that there is no inclusion. Colored vertices give the new security classes introduced in this work.

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* 32nd *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=VirToolAccessed 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

Paper I

Classes of Garbling Schemes

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

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

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

Paper 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

Paper 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

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

Paper VII

How to Use Garbling for Privacy Preserving Electronic Surveillance Services

T. Meskanen and V. Niemi and N. Nieminen (2015). *Cyber Security and Mobility*, 4(1):41-64

VII

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

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

For example, if there exists a function f such that the evaluation of f, i.e. $\texttt{ev}(f, \cdot)$, is not injective then the assumption $(*)$ holds.

There are pathological cases where the assumption $(*)$ does not hold. For instance, if both Φ and $\texttt{ev}$ 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 $X = X_1 \ldots X_n$ is the fully specified garbled input which is obtained by using $\mathtt{En}$ of garbling scheme $\mathcal{G}$ whereas $\widehat{X} = \widehat{X_1} \ldots \widehat{X_n}$ is the fully specified argument encrypted with the independent key $\widehat{e}$. Finally, $S = \{s_1, \ldots s_n\}$ 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. $\square$

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 $\mathcal{F}_\ell \neq \mathcal{C}_\ell$. $\square$

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 $(e, \widehat{e}, x, \{s_1, s_2, \dots, s_t\})$ as input and outputs $(X, \widehat{X}, s)$. Here, the key $\widehat{e}$ is an encryption key from an independent, secure, symmetric, non-deterministic 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 $j \in \{1, \dots, t\}$. 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 $x = x_1 \dots x_n$ and $x' = x'_1 \dots x'_n$ 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 $\overline{X} = \overline{X_1} \dots \overline{X_n}$ and $\overline{X'} = \overline{X'_1} \dots \overline{X'_n}$ 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 $\texttt{GS}(\text{obv.sim.padap}_\ell) \subseteq \texttt{GS}(\text{mao.sim.padap}_\ell)$ which concludes the proof. $\square$

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 $\mathcal{G} = (\texttt{KeyGen}, \texttt{Ga}, \texttt{En}, \texttt{De}, \texttt{Ev}, \texttt{ev}) \in \bigcap_{\ell=1}^{\infty} \mathcal{R}_\ell$ 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. $\square$

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 $\Phi(f_0) \neq \Phi(f_1)$,

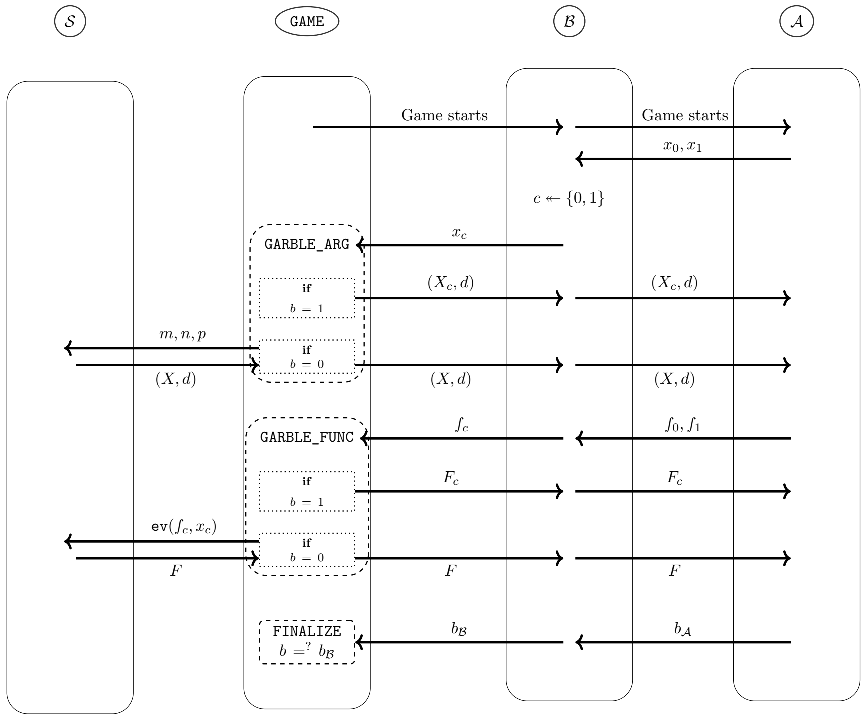


Figure 6.1: In the above figure, $\mathcal{B}$ is an adversary playing the game PrvSimRadap_ℓ denoted in the diagram by `GAME`. $\mathcal{S}$ is the simulator in this game. $\mathcal{A}$ is an adversary presuming to play game PrvIndRadap_ℓ, but actually adversary $\mathcal{B}$ uses $\mathcal{A}$ as a subroutine by trying to emulate the actual game PrvIndRadap_ℓ.

$\{x_0, x_1\} \not\subseteq \{0,1\}^m$ or $\mathtt{ev}(f_0, x_0) \neq \mathtt{ev}(f_1, x_1)$, $\mathcal{B}$ 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 $\mathtt{ev}(f_0, x_0) =^? \mathtt{ev}(f_1, x_1)$ 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 $\texttt{ev}(f_0, x_0) =^? \texttt{ev}(f_1, x_1)$. 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$_\ell$. 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 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$_\ell$ is also negligible, which proves the first inclusion in the claim.

Consider then the second inclusion $\texttt{GS}(\text{obv.ind.radap}_\ell, \Phi) \subseteq \texttt{GS}(\text{mao.ind.radap}_\ell, \Phi)$. Suppose that $\mathcal{G} \in \texttt{GS}(\text{obv.ind.radap}_\ell, \Phi)$. Let $\mathcal{A}$ be an arbitrary adversary playing the game MaoIndRadap$_\ell$. We construct an adversary $\mathcal{B}$ playing the game ObvIndRadap$_\ell$ which uses adversary $\mathcal{A}$ as his subroutine as follows.

The ObvIndRadap$_\ell$ game starts by the choice of the challenge bit b in INITIALIZE procedure. Then the ObvIndRadap$_\ell$ 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$_\ell$ game. Since adversary $\mathcal{A}$ presumes to play the actual MaoIndRadap$_\ell$ 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 $y_0 = \texttt{ev}(f_0, x_0)$ and $y_1 = \texttt{ev}(f_1, x_1)$ 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_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 $\text{obv.ind.radap}_\ell$ secure, the advantage of adversary $\mathcal{B}$ is negligible. This yields that the advantage of an arbitrary adversary playing game MaoIndRadap_ℓ is also negligible, which proves the second inclusion in the claim. □

Proof of Theorem 8: Our aim is to prove that $\texttt{GS}(\text{prv.sim.radap}_\ell) \cup \texttt{GS}(\text{obv.sim.radap}_\ell) \subseteq \texttt{GS}(\text{mao.sim.radap}_\ell)$. 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 $\text{prv.sim.radap}_\ell$ secure. Our aim is to prove that then $\mathcal{G}$ is also mao.sim.adap_ℓ secure. There is only one difference in the games PrvSimAdap_ℓ and MaoSimAdap_ℓ: in game PrvSimAdap_ℓ, GARBLE_ARG returns the decryption key d together with X whereas in game MaoSimAdap_ℓ, 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_ℓ game. This proves the first claim.

Then, consider the second claim $\texttt{GS}(\text{obv.sim.adap}_\ell, \Phi) \subseteq \texttt{GS}(\text{mao.sim.adap}_\ell, \Phi)$. Suppose that the garbling scheme $\mathcal{G}$ is obv.sim.adap_ℓ 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_ℓ game. Let $\mathcal{A}$ be an arbitrary adversary playing the MaoSimRadap_ℓ game and let $\mathcal{A}'$ be the corresponding adversary playing the ObvSimRadap_ℓ game. Adversary $\mathcal{A}'$ emulates the behavior of adversary $\mathcal{A}$ as before.

Because $\mathcal{G}$ is an $\text{obv.sim.radap}_\ell$ 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_ℓ 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_ℓ game and the advantage of $\mathcal{A}$ in game MaoSimRapad_ℓ 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 $\mathtt{GS}(\text{prv.sim.radap}_\ell) \cup \mathtt{GS}(\text{obv.sim.radap}_\ell) \subseteq \text{mao.sim.radap}_\ell$. $\square$

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 $x \overset{?}{\in} D_f$, 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

$$\texttt{GS}(\mathit{SIM.STAT}, \Psi_{\text{tot}}) \bigcap \texttt{GS}(\texttt{ev}_{\text{circ}}) \subseteq \texttt{GS}(\mathit{prv.sim.stat}, \Phi) \bigcap \texttt{GS}(\texttt{ev}_{\text{circ}}).$$

This concludes the proof. $\square$

Proof of Theorem 2: We prove only the equality

$$\texttt{GS}(\text{obv.ind.adap}_\ell, \Phi) \bigcap \texttt{GS}(\texttt{ev}_{\text{circ}}) = \texttt{GS}(\text{IND.ADAP}_\ell, \Psi_{\text{tot}}) \bigcap \texttt{GS}(\texttt{ev}_{\text{circ}}).$$

The other cases can be proven in similar manner. For simplicity we use notation `ev` instead of $\texttt{ev}_{\text{circ}}$ throughout the proof. Let us first assume that $\mathcal{G}$ is obv.ind.adap$_\ell$ secure over side-information function Φ. Our aim is to prove that then $\mathcal{G}$ is also IND.ADAP$_\ell$ 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 $X = \texttt{En}(e, x_b)$ 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. $\square$

Proof of Theorem 3:

We prove the claim in case

$$\texttt{GS}(\text{mao.sim.adap}_\ell, \Phi) = \texttt{GS}(\text{SIM.ADAP}_\ell, \Psi_{\text{tot}}).$$

The other cases are proven in similar manner.

Consider first the claim $\texttt{GS}(\text{mao.sim.adap}_\ell, \Phi) \subseteq \texttt{GS}(\text{SIM.ADAP}_\ell, \Psi_{\text{tot}})$. 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

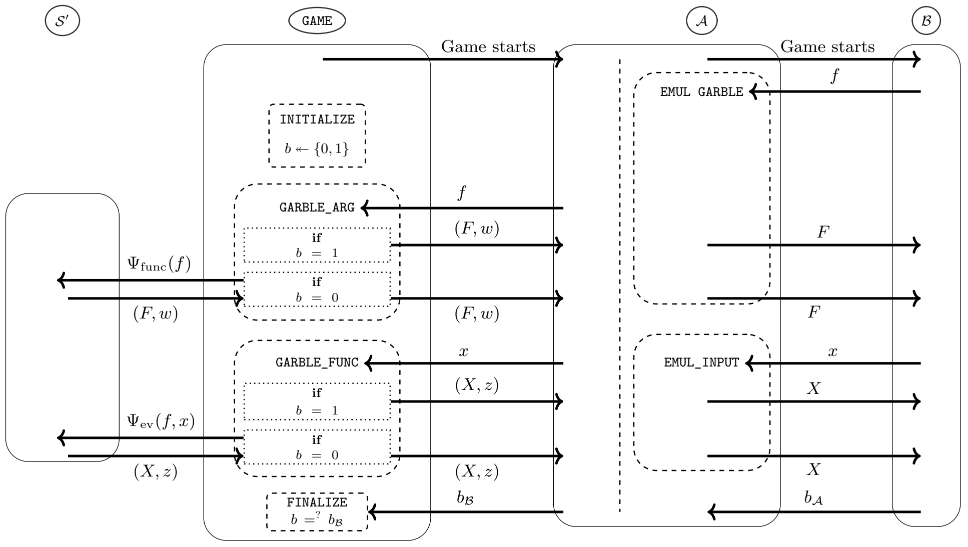


Figure 6.2: In the above figure, $\mathcal{A}$ is an adversary playing the game SIM.ADAP$_\ell$ denoted by `GAME` in the diagram. $\mathcal{S}'$ is the simulator in this game. $\mathcal{B}$ is an adversary presuming to play game MaoSimAdap$_\ell$. Actually adversary $\mathcal{A}$ uses $\mathcal{B}$ as 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 $\mathtt{GS}(\text{XXX.YYY}, \Phi) = \mathtt{GS}(\text{mao.xxx.yyy}, \Psi_{\text{tot}})$. $\square$

ISBN 978-952-12-3516-0
ISSN 1239-1883

Noora Nieminen Garbling Schemes and Applications

Noora Nieminen Garbling Schemes and Applications

Nooara Nieminen Garbling Schemes and Applications