Chapter 1
Obfuscation and Diversification for Securing Cloud Computing

Shohreh Hosseinzadeh, Samuel Laurén, Sampsa Rauti, Sami Hyrynsalmi, Mauro Conti, Ville Leppänen

May 24, 2016

Springer

Contents

Abstract The evolution of cloud computing and advancement of its services has motivated the organizations and enterprises to move towards the cloud, in order to provide their services to their customers, with greater ease and higher efficiency. Utilizing the cloud-based services, on one hand has brought along numerous compelling benefits and, on the other hand, has raised concerns regarding the security and privacy of the data on the cloud, which is still an ongoing challenge. In this regard, there has been a large body of research on improving the security and privacy in cloud computing. In this chapter, we first study the status of security and privacy in cloud computing. Then among all the existing security techniques, we narrow our focus on *obfuscation* and *diversification* techniques. We present the state-of-the-art review in this field of study, how these two techniques have been used in cloud computing to improve security. Finally, we propose an approach that uses these

two techniques with the aim of improving the security in cloud computing environment and preserve the privacy of its users.
Keywords: cloud computing, enterprise security, security, privacy, obfuscation, diversification

1.1 Introduction

The recent changes in the business world have made the organizations and enterprises more interested in using cloud to share their services and resources remotely to their users. For this purpose, cloud computing offers three different models (Mell and Grance, 2011): Software as a service (SaaS), Platform as a service (PaaS), and Infrastructure as a service (IaaS). In IaaS model the services that are offered by the service provider include computing resources, storage, and virtual machines. The PaaS model presents computing platforms to the business and its end users. In SaaS, the main services offered by the service providers are the applications that are hosted and executed on the cloud and are available to the customers through the network, typically over the Internet. Depending on the need of the enterprise, a suitable delivery model is deployed.

The advancements in the cloud computing have facilitated the business, organizations, and enterprises with services providing lower cost and higher performance, scalability, and availability. Due to these advantages, cloud computing has become a highly demanded technology and organizations are relying on cloud services more and more. However, by using the cloud, more data is stored outside the organization's perimeters, which raises concerns about the security and privacy of the data. Therefore, it is significant for the cloud service providers to employ effective practices to secure the cloud computing infrastructure and preserve the privacy of its users. In the context of the cloud computing security, there exist many different measures that protect the cloud infrastructure. Some of these measures consider the cloud as an untrusted or malicious infrastructure that the user's data should be protected from (e.g., the cloud uses the data without the user's consent). While some other measures protect the infrastructure from the external intrusions.

Obfuscation and diversification are two propitious software security techniques that have been employed in various domains, mainly to impede malware (i.e., malicious software) (Skoudis, 2004). These techniques have also been used to provide security in cloud computing as well. In a previous work[1], we have systematically surveyed the studies that use obfuscation and diversification techniques with the aim of improving the security in cloud computing environment (see the details of this study in Section 1.3.1). By analyzing the collected data, we managed to identify the areas that have gained more

[1] This book chapter is a re-written extended version of our previous study (Hosseinzadeh et al., 2015)

attention by the previous research, and also the areas that have remained intact and potential for further research. The results of the survey motivated us to propose a diversification approach, aiming at improving the security in cloud computing. We demonstrate this approach by applying obfuscation on client-side JavaScript components of an application. As such, we make it complicated for a piece of malware to gain knowledge about the internal structure of the application and perform its malicious attack. Moreover, we distribute unique versions of the application to the computers, which in the end mitigates the risk of massive-scale attacks (more detail in Section 1.4).

This book chapter is structured as follows: Section 1.2 discusses the security and privacy in domain of cloud computing. The available security threats and different aspects of security, concerning the cloud computing technology. In Section 1.3 we introduce the terms and techniques that are used in the proposed approach, and we present the state of this field of study, i.e., how these techniques have previously been used to boost the security of cloud computing. In Section 1.4, we present our proposed approach in detail. Conclusions come in Section 1.5.

1.2 Security and privacy in cloud computing

Cloud computing is an evolving technology with new capabilities and services that have remarkable benefits when compared to more traditional service providing approaches. The services are delivered with lower cost (in usage as it is pay-for-use, in disaster recovery, in data storage solutions), greater ease, less complexity, higher availability and scalability, and also faster deployment. These compelling benefits have motivated the enterprises to adopt cloud solutions in their architectures and deliver their services over cloud. Depending on the need of the enterprise and how large the organization is, different deployment models are available, including public clouds, private clouds, community cloud, and hybrid clouds (Mell and Grance, 2011; Mather et al., 2009). Again, depending on the need of the enterprise, cloud providers offer three different delivery models, i.e., SaaS, IaaS, and PaaS. In the following, we discuss various deployment models, and various business models in cloud computing (CSA, 2016).

In a *public cloud* (or external cloud), a third party vendor is responsible for hosting, operating, and managing the cloud. A common infrastructure is used to serve multiple customers, which means that the customers are not required to acquire for any software, hardware, and network devices. This makes the public cloud a suitable model for the enterprises that wish to invest less and manage the costs efficiently. The security in a public cloud is managed by the third party, which leaves less control for the organization and its users over the security (Rhoton et al., 2013). A kind of opposite solution to that is *private cloud* (or internal cloud), where the organization's customers are

in charge of managing the cloud. The storage, computing, and network are dedicated to the organization owning the cloud, and not shared with other organizations. This enables the customers to have a higher control on security management and have more insight about logical and physical aspects of the cloud infrastructure. *Community cloud* refers to the type of clouds that are used exclusively by a community of customers from enterprises with common requirements and concerns (e.g., policies, security requirements, and compliance considerations). The last model is *hybrid cloud* that is the composition of several clouds (private, public, and community). According to the needs and budget of the enterprise and how critical its resources are, a suitable deployment model is chosen that can serve the enterprise's needs the best. For instance, an enterprise pays more for private cloud and has better security control over its shared resources, while it spends less on a public cloud for which it has less security control (CSA, 2016; Mather et al., 2009).

As mentioned before, cloud service providers use three different business models to deliver their services to the end users: IaaS, PaaS, and SaaS. IaaS is the foundation of the cloud, PaaS comes on top of that, and SaaS is built upon PaaS. Each of these delivery models has different security issues, and is prone to different types of security threats, and therefore, it requires different levels of security.

The IaaS service model offers capabilities such as storage, processing, networks, and computing resources to the consumers. The end user does not manage the underlying infrastructure of the cloud, but has control over applications, operating system and storage. IaaS has made it a lot easier for the enterprises to deliver their services, in a way that they no longer worry about provisioning and managing the infrastructure and dealing with the underlying complexities. In addition to that, it has made it cheaper for businesses, in a way that instead of paying for the data centers and hosting companies, they only need to spend for the resources they consume to IaaS providers (Mell and Grance, 2011; Mather et al., 2009).

The PaaS model is built upon IaaS and offers a development environment to the developers to develop their applications, without worrying about the underlying infrastructure. The offered services consist of a complete set of software development kit, ranging from design to testing and maintenance. The consumers of this model do not have control on the beneath services (e.g. the operating system, server, network, and the storage), but they can manage the application-hosting environment (Mell and Grance, 2011; Mather et al., 2009). The dark side of these advantages is that the PaaS infrastructure can also be used by a hacker to malicious purposes (e.g., running the malware codes and commands).

In the SaaS model, the service providers host the applications remotely and make them available to the users, when requested, over the Internet. SaaS is an advantageous model for the IT enterprises and their customers, as it is more cost-effective and has better operational efficiency. However, there are still concerns about the security of the data store and software, which the

vendors are required to address them (Mell and Grance, 2011; Mather et al., 2009).

In addition to these three main service delivery models, the cloud offers other models and the infrastructure is utilized these days for many other purposes, such as Security-as-a-Service (SECaaS). Using this service model, many security vendors deliver their security solutions using cloud services. That is to say, the security management services are outsourced to an external service provider, and delivered to the users over the Internet. SECaaS applications can be in the form of anti-virus, anti-spam, and malware detection programs. The programs operate on the cloud, instead of client-side installed software, and with no need for on-premises hardware (Varadharajan and Tupakula, 2014)

Employing cloud services by an enterprise is a two-edged sword, meaning that it has both positive and negative impacts. On one hand, by outsourcing and shifting some responsibilities from the enterprise to the cloud, fewer unwanted incidents are expected to occur. This is due to the fact that the cloud providers have a more advanced and experienced position in offering more secure services that are supported by their specialized staff, and also there are incident management plans for the case of break outs. However, this transferring of the responsibilities, on the other hand, decreases the control of the enterprise over the critical services. In addition to that, storing the data outside the organization's firewalls raises concerns about potential vulnerabilities and possible leaks. For instance, if the information fall into wrong hands (e.g., exposed to hackers or competitors), it results in loss of customer's confidence, damage to the organization's reputation, and even legal and financial penalties for the organization. On this basis, enterprises that are planning to adopt cloud services put together the positive and negative impacts, weigh them up, and do risk assessment (Rhoton et al., 2013).

In spite of the benefits of adopting cloud-based services in the enterprises, there exist still some barriers. Among all, security and privacy are the most significant barriers. The fundamental challenges related to the cloud security are the security of the data storage, security in data transmission, application security, and security of the third party resources (Subashini and Kavitha, 2011). Among all the security risks associated with the cloud, the followings are the top severe security threats reported by the Cloud Security Alliance (CSA) (Top Threats Working Group, 2013):

- Data breach: As a result of a malicious intrusion the (sensitive) data may be disclosed to unwanted parties, including the attacker and competitors.
- Data loss: The data stored on the cloud could be lost due to an attack, unintentional/accidental deletion of the data by the service provider, and physical corruption of the system infrastructure.
- Hijacking of the accounts and traffic: Attackers by getting access to the users' credentials, through phishing and exploiting the software vulnerabilities can read or alter the users' activities. This consequently puts the confidentiality, integrity, and availability of the system at risk.

- Insecure Application Program Interfaces (API): Clients use the SW interfaces to interact with the cloud. These APIs should be sufficiently secure to protect both the consumer and the service provider.
- Denial of service: An intruder by sending illegitimate requests to the service provider attempts to occupy the resources, so to disable/slow down the cloud to process the legitimate requests.
- Malicious insider: The adversary is not always an outsider, but can be a person inside the cloud system who has an authorized access to the data and intentionally misuses such authorization.
- Abuse of the cloud service: The cloud computing serves the organizations with extensive computational power; however, this power could be misused by a malicious user to perform his belligerent action.
- Insufficient due diligence: Before the organizations move their services to the cloud, it is significant to have a proper understanding of the capabilities and adaptabilities of their resources with the cloud technologies.
- Shared technology issues: Sharing platforms, infrastructures, and applications has made the delivery of the cloud services feasible; however, such sharing has the drawback that vulnerability in a single piece of shared component can be propagated potentially to the entire cloud.

In securing the cloud computing environment various aspects should be taken into account. The International Information Systems Security Certification Consortium $(ISC)^2$ (ISC, 2016) has presented taxonomy of the security domains concerning the cloud computing, which covers the following aspects: physical security, access control, telecommunications and network security, cryptography, application security, operation security, information security and risk management practices, and business continuity and disaster recovery planning.

Security and privacy come hand in hand, in other words, a more secure system better protects the privacy of its users. Therefore, while integrating cloud in organization's architecture, it is highly significant for the enterprise to assure that a cloud service provider is considering all the security aspects, and adequately addressing the privacy regulations.

Considering the fact that the proposed approach in this study is aimed at securing the application, and protecting SaaS and PaaS models, in the following section we study the state of cloud application security.

1.2.1 Application security in cloud computing

Talking about the IaaS model, it is more straightforward to provide protection by hardening the platform through allowing the traffic from trusted IP addresses, running anti-virus programs, applying security patches, and so on. However, when it comes to PaaS and SaaS models, this may not be the case;

since in these models, ensuring the security of the platform is the service provider's responsibility (Rhoton et al., 2013). Moreover, in SaaS, there is less transparency and visibility about how the data is stored and secured. This makes it more difficult for the enterprises to trust the service provider.

Application security covers the measures and practices taken throughout the software development life-cycle to reduce the vulnerabilities and flaws. Because of the fact that the cloud-base applications are connected directly to the Internet, cloud offers less physical security compared to traditional data centers and service providers. Also because of the co-mingled data and multi-tenancy behavior of the cloud, the cloud's applications are prone to additional attack vectors.

Recent security incidents clearly show that the exploits by taking advantage of the software flaws and vulnerabilities, make the web applications the leading targets for attacks. Web applications are the simplest and the most profitable targets, from the attackers' perspective. Especially, in the case of cloud computing that the applications are accessed through the user's browser, website security is the sole means to impede the attacks. Moreover, the security breaches through exploiting the applications and web services have shown to be pretty severe and have lead to big losses. Stuxnet (Chen and Abu-Nimeh, 2011) is one example of infecting the software with the aim of affecting critical physical infrastructures and industrial control systems. The other example is using SQL injection to steal the debit/credit card numbers, which in the end resulted in $1 million withdrawals from the ATM machines worldwide (CSA, 2016).

In spite of the significance of the application security, it has been considered as an afterthought in many enterprises. In other words, application security has seldom been the top priority and the main focus for neither the security practitioners and nor the enterprises and less security budget has been allocated on it (CSA, 2016).

On this basis, more consideration is required both from the business side by shifting more budget to application security and also from the security team to concentrate more on securing the web applications, the most exposed component of the business.

As in other domains, application security in cloud computing is a crucial component in operational IT strategy. Regardless of where an application is residing, the enterprise is responsible for ensuring the effectiveness of the security practices to protect the application. Also, as we discussed earlier, the nature of cloud computing environment introduces additional risks compared to on-premise applications and web services.

1.3 Obfuscation and diversification for securing cloud computing

Code obfuscation refers to the deliberate act of scrambling the program's code and transforming it in a way that it becomes harder to read (Collberg et al., 1997). This new version of the code is functionally similar to the original code, while syntactically different. This means that even though the obfuscated code has different implementation, given the same input, it produces the same output. The main purpose of code obfuscation is to make the understanding of code and its functionality more complicated and to prevent the act of malicious reverse engineering.

a)

```
function setText(data) {
  document.getElementById("myDiv").innerHTML = data;
}
```

b)

```
function ghds3x(n) {
  h = "\x69\u006En\u0065r\x48T\u004DL";
  a="s c v o v d h e , n i";x=a.split(" ");b="gztxleWentBsyf";
  r=b.replace("z",x[7]).replace("x","E").replace("s","").replace("f","I")
  ["repl" + "ace"]("W","m")+"d"; c="my"+String.fromCharCode(68)+x[10]+"v";
  s=x[5]+x[3]+x[1]+"um"+x[7]+x[9]+"t";d=this[s][r](c);if(+!![])
  { d[h]=n; } else { d[h]=c; }
}
```

Fig. 1.1 a) a piece of JavaScript code, and b) an obfuscated version of the same code

Figure 1.1 is an example of obfuscated code that clearly shows how much harder it can become to read and comprehend the code after it is obfuscated. With no doubt, within a given time an attacker may succeed in reverse engineering the obfuscated code and breaking it: however, it is harder and costlier now, compared to the original code.

In the literature, many different obfuscation mechanisms have been proposed (Popov et al., 2007; Linn and Debray, 2003). Each of these mechanisms targets various parts of the code to apply the obfuscation transformation. Among all, the techniques that attempt to obfuscate the control of the program (Nagra and Collberg, 2009), are the most commonly used. These techniques alter the control flow of the program, or generate a fake one, so it would be more challenging for a malicious analyzer to understand the code. To this end, bogus insertion (Drape and Majumdar, 2007), and opaque predicates (Collberg et al., 1998) are effective control flow obfuscation techniques.

Software diversification aims at generating unique instances of software in a way that they appear with different syntax but equivalent functionality (Cohen, 1993). Diversification breaks the idea of developing and distributing the software in a monoculture manner, and introduces multiculturalism to software design. In the other words, the identical designs of the software in-

stances make them have similar vulnerabilities and are prone to similar types of security threats. This offers the opportunity to an attacker to design an attack model to exploit those vulnerabilities and easily compromise a wide number of execution platforms (e.g., computers). The risk of this kind of massive-scale attacks can be mitigated through diversifying the software versions, so that the same attack model will not be effectual on all instances. The way a program is diversified is kept secret and pieces of malware that do not know the secret cannot interact with the environment and eventually become ineffective. However, the created secret has to be propagated to the trusted applications, so it will still be feasible for them to access the resources. In the worst case scenario, even if the attacker gains the secret of diversification of one instance, that secret is specified to that computer and a costly analysis is required to find out other secrets to attack other computers. There has been survey studies surveying software diversity (Larsen et al., 2014; Baudry and Monperrus, 2015)

A particular version of diversification is *interface diversification* which is applied to internal interfaces of software (APIs or instruction sets of languages). For example, the system call interface (for accessing all kinds of resources of a system) is one typical *internal* interface which can be changed without sharing the details of new internal interface to external parties (e.g. malware) (Rauti et al., 2014). Of course, the details on diversified internal interface need to be propagated to all legal applications so that those programs can still use the system's resources (Laurén et al., 2014).

Figure 1.2 illustrates distribution of diversified versions of program P among the users. Each of the programs P1, P2, and P3 are unique in structure and diversified differently. Thus, one single attack model does not work for multiple systems, and attack models need to be designed to be system-specific. When program P3 is attacked, other versions are still safe.

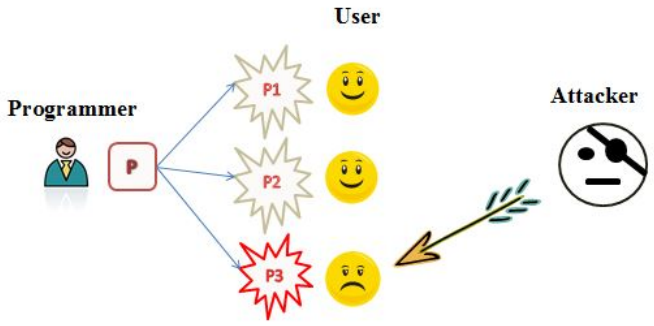

Fig. 1.2 Diversification generates unique versions of software. Therefore, even if one copy of software is breached, other copies are safe.

Program bugs left at the development time are inevitable and cause software vulnerabilities. Some of these vulnerabilities are not known, while releasing the software. Later on, a malicious person can gain knowledge about the system and its vulnerabilities, and write a piece of malware to exploit those vulnerabilities performing a successful attack. Especially, the interface diversification techniques can be helpful in preventing such zero day type of attacks, since the malicious person no longer automatically know the necessary (for malware) internal interfaces for accessing resources.

In general, diversification and obfuscation techniques do not attempt to remove these vulnerabilities, but attempt to prevent (or make it hard, at least) the attacker/malware to taking advantage of them to run its malicious code. Obfuscating and diversifying the internal interfaces of the system makes it challenging for malware to attain the required knowledge about the system, how to call the systems interfaces, in order to execute its malicious code.

1.3.1 Related work on security of cloud through obfuscation and diversification

As mentioned before, diversification and obfuscation techniques have been used in different domains to provide security, including cloud computing. In a previous study (Hosseinzadeh et al., 2015), we systematically studies in what ways these two techniques have been used in cloud computing environment with the aim of improving the security. As the result of the search, we collected 43 studies that were discussing diversification and obfuscation as the techniques for improving the security in the cloud and protect the privacy of its users, and we classified them based on how the techniques are used to this aim. After extracting data from those studies, we identified that the obfuscation and diversification techniques are used in nine different ways to boost the security and privacy of the cloud, including: 1) generating noise obfuscation, 2) client-side data obfuscation as a middleware, 3) general data obfuscation, 4) source code obfuscation, 5) location obfuscation, 6) file splitting and storing on separate clouds, 7) encryption as obfuscation, 8) diversification, and 9) cloud security by virtue of securing the browser. Figure 1.3 illustrates these categories with the number of studies in each group.

Many of the cloud service providers are complying with the policies and regulations in order to protect the privacy of their customers. However, there exist a wide number of service providers that may record the collected data from the customers, deduce and misuse the private information without user's consent. Hence, there is a need for practices to be taken at client side (without service provider's interference) to protect the privacy. Obfuscation and diversification techniques were employed to protect the data from the cloud. In majority of studied works, the cloud service provider was considered as untrusted/malicious.

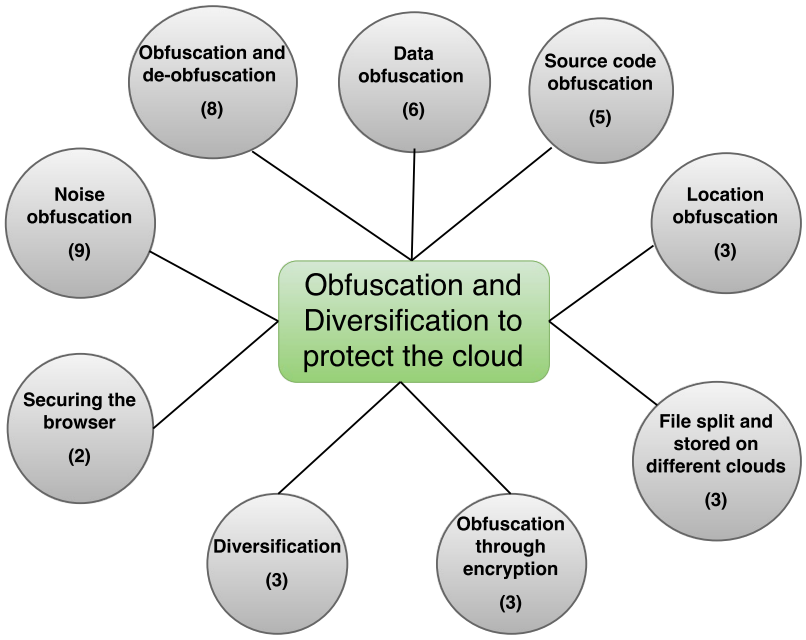

Fig. 1.3 The related studies on security and privacy in cloud computing through obfuscation and diversification techniques.

In the following, we explain the nine different ways that obfuscation/diversification have been used in the literature for protecting the cloud from security threats, and also protecting the user's privacy from the malicious cloud.

- Generating noise obfuscation: This approach resides on the client side and tries confuses the malicious cloud by injecting irrelevant requests that are similar to legitimate requests of the user (called noise) into the user's service requests, i.e., the requests sent from the customer to the cloud. In this way, the occurrence probability of the legitimate requests and the noise requests become the same, so it becomes difficult for the cloud to distinguish the real request. Noise generation strategy, conceals real requests coming from the users, and therefore, lessens the probability of request being revealed. As the result, the privacy of the customer is protected.
- Client-side data obfuscation as a middleware: This method protects the data from the untrusted service provider while the data is stored or processed on the cloud. A privacy managing middleware on the client side obfuscates the (sensitive) data using a secret key which is chosen and kept by the user. The obfuscated data is sent to the cloud and is processed on the cloud without being de-obfuscated. This is because the key is kept secret to the user and the cloud does not have the key to de-obfuscate the

data. The result of the process is sent to the user and is de-obfuscated on the client side, so the user sees the plain data.

- General data obfuscation: In this class obfuscation, some transformations that are made into the user's data, which make them harder to read/understood. This method can be used to protect the user's identity information, the data stored on the database of the cloud, and the user's behavioral pattern. Data obfuscation makes the user's confidential harder to be exposed, and therefore, protects the user's data privacy.
- Source code obfuscation: As explained before, source code obfuscation is a technique for protecting the software from reverse engineering. This method can also be used for securing the cloud's software from attacks and risk of malware.
- Location obfuscation: As we know, there exist services that rely on the physical information of the user to provide the services. This includes privacy concerns on revealing the precise location of the user (e.g., concerns about locating and tracking down the user). To this end, obfuscating the location information is a technique to make the exact location imprecise through generalizing, or slightly altering the precise location to avoid the actual position being exposed and consequently, preserve the location privacy.
- File splitting and storing on separate clouds: The idea in this obfuscation strategy is to divide the data/files into different sectors and store them on different clouds. This approach ensures not only the security, but also the availability of the data. If one cloud is attacked, only one part will be leaked, and the other parts are safe.
- Encryption as obfuscation: Obfuscation could be attained through cryptographic techniques. For instance, homomorphic encryption and one-way hash function are examples of this obfuscation strategy. Obfuscating the data reduces the risk of data leakage and even if the data is leaked, it is quite useless, as it is scrambled. Therefore, it is a beneficial technique in preserving the confidentiality of the data on the cloud.
- Diversification: As discussed before, different components could be the target for diversification depending on the security need of the system. In cloud computing paradigm, it is proposed to continuously diversify the execution environment, so to shorten the time for the attacker to learn the execution environment and the vulnerabilities of it. The execution environment changes to a new environment, before the attacker gets the chance to obtain sufficient knowledge about it.
- Cloud security through securing the browser: In this idea a plug-in is embedded in the user's web browser, which has the capability of data obfuscation and hybrid authentication. Therein, the security and the privacy of the data are addressed in the web browser.

Table 1.1 lists all the papers that are discussing diversification and obfuscation as the promising security techniques in cloud computing environment.

Based on how the studies use diversification/obfuscation is in cloud computing, they fall into nine different categories.

Table 1.1: List of the studies

No.	Description of method
	Category 1: Generating noise obfuscation
1	(Zhang et al., 2013): Injecting noise requests in user's request makes it difficult for the cloud to distinguish the legitimate request. The paper considers noise obfuscation as a way for privacy-leakage-tolerance.
2	(Yang et al., 2013): This paper proposes noise generation approach as a way to obfuscate the data while the characteristic of the data is not changed. The main goal is to protect the privacy in the domain of data statics and data mining.
3	(Zhang et al., 2012b): In this paper, Time-series Pattern Strategy Noise Generation (TPNGS) is used to create a pattern based on the previous requests that the user has made, and with the help of this pattern predict the occurrence probability of the future requests. This approach makes the real requests of the user vague, and protects the privacy of the client from a malicious cloud.
4	(Zhang et al., 2015): In this work, noise obfuscation approach considers occurrence probability fluctuation as a way to disguise the customer's data.
5	(Zhang et al., 2012c): Noise injection is discussed in this paper as a method to confuse the malicious cloud provider, with the aim of privacy protection.
6	(Zhang et al., 2012a): Injecting noise (=irrelevant requests) into the user's request makes the occurrence possibility of the real and the noise requests the same, and thus make them indistinguishable.
7	(Lamanna et al., 2012): This paper considers homomorphic encryption, oblivious transfer, and query obfuscation in the proxy as the techniques to protect the information from an untrusted cloud. Query obfuscation aims at generating random noisy/fake queries and confusing for the cloud.
8	(Zhang et al., 2012d): Noise obfuscation disguises the occurrence probability of the user's requests. In this way, the user's personal information is kept safe, and therefore, the privacy is conserved.
9	(Liu et al., 2012): In this paper, generating noise in user's requests is discussed as a way to protect the privacy.
	Category 2: Client-side data obfuscation as a middleware
10	(Arockiam and Monikandan, 2014): Before sending the data to the cloud, encryption and obfuscation techniques are used to ensure the confidentiality of the data. Obfuscation is used for the numerical data types, while encryption is applied on alphabetical type of data.
11	(Tian et al., 2011): This paper suggests that the user's information be encrypted before being sent to the cloud. This encrypted data is decrypted only on the client side.

Continued on the next page

Table 1.1 – continued from the previous page

No.	Description of method
12	(Yau and An, 2010): The proposed approach protects the customer's data from malicious cloud through data obfuscation, information hiding, and separating the software and the infrastructure of the service provider.
13	(Mowbray et al., 2012): This paper introduces a privacy manager that protects the user's private information by obfuscating them before delivering to the cloud. The key used for this purpose is selected by the privacy manager. The same key is used to de-obfuscate the processed data received from the cloud. They use the term obfuscation rather than encryption, since the data is partially obfuscated and some parts remain intact.
14	(Pearson et al., 2009): This work presents a mathematical formulation for obfuscation, and also a privacy manager founded on obfuscation and de-obfuscation approaches.
15	(Mowbray and Pearson, 2009): This paper proposes a privacy managing technique based on obfuscation and de-obfuscation approaches, to control the data transferred to the cloud. User's information is obfuscated using the key selected by user, and then sent to the cloud. This key is kept secret by the user, so the cloud is never able to de-obfuscate the data.
16	(Govinda and Sathiyamoorthy, 2012): In this approach customer's confidential data is obfuscated before being sent to the cloud service provider. The result of the processed data is sent back to the customer. There, the data is de-obfuscated on the client side using the user's secret key.
17	(Patibandla et al., 2012): In this work, a privacy manager software is presented that obfuscates the user's sensitive data, prior to send to the cloud, based on the user's preferences.
	Category 3: General data obfuscation
18	(Reiss et al., 2012): This paper proposes a systematic obfuscation approach that aims at protecting personal data. The obfuscation techniques used are: a) transforming: changing the information into another format, b) sub-setting: selecting a particular fraction of data, c) culling: deleting a particular fraction of data, and d) aggregation.
19	(Kuzu et al., 2014): Data obfuscation is an advantageous solution to protect the data that is stored in the cloud's database. Besides, the access patterns could be obfuscated and protected as well.
20	(Vleju, 2012): The confidential information of the user can be protected through obfuscation. For instance, obfuscating the identification information conceals the user's real identity. After obfuscation is applied, the data can be deciphered only by the user.
21	(Li et al., 2011): To protect the data privacy in SaaS, data obfuscation is proposed as a beneficial technique.
22	(Qin et al., 2014): This paper proposes an algorithm based on obfuscation techniques to protect the confidential information that exist in CNF (Conjunctive Normal Form) format.

Continued on the next page

Table 1.1 – continued from the previous page

No.	Description of method
23	(Tapiador et al., 2012): Typically, a user's decisions and behavior follow a similar pattern. Analyzing this pattern helps in foreseeing the future behavior which raises privacy concerns. Obfuscating the user's behavioral pattern, make this information inaccessible, or at least makes it harder to access.
24	(Kansal et al., 2015): This paper proposes image obfuscation as a technique to hide and obfuscate an image (for instance by hiding the position of the pixels or the colors). For this purpose, the paper integrates the compression and secret sharing to produce multiple numbers of shadow images.
	Category 4: Source code obfuscation
25	(Bertholon et al., 2013a): JavaScript is the language that is widely used in today's web services. To protect the JavaScript code, obfuscation is proposed to make it harder to reverse engineer.
26	(Bertholon et al., 2013b): The paper presents a framework that transforms/obfuscates the source code of the C program into a jumbled form.
27	(Hataba and El-Mahdy, 2012): This paper is a survey of existing obfuscation techniques that aim at making the reverse engineering harder.
28	(Bertholon et al., 2014): JSHADOF framework is designed to obfuscate the JavaScript code. The target of the transformation is the source code in cloud computing web services.
29	(Omar et al., 2014): This paper uses control-flow obfuscation and junk code insertion to present a threat-based obfuscation technique.
	Category 5: Location obfuscation
30	(Karuppanan et al., 2012): This paper states that the user's private information needs to be protected from being disclosed. Location obfuscation is proposed to conceal the user's location.
31	(Skvortsov et al., 2012): The paper states that the Location Services (LS) present services based on the location information of the user, which brings along privacy concerns. Location obfuscation solves this problem by making this information appear imprecise.
32	(Agir et al., 2014): This paper considers location obfuscation as a way to confuse the server about the location of the user.
	Category 6: File splitting and storing on separate clouds
33	(Celesti et al., 2014): In this work, data obfuscation is done through dividing the files and storing them on multiple clouds. In this way, each cloud has partial view to the file.
34	(Ryan and Falvey, 2012): In order to obfuscate the data, this work proposes splitting the data and storing them on geographically separated data stores.
35	(Villari et al., 2013): To keep the data confidential, it is proposed to spread the data over various clouds.
	Category 7: Encryption as obfuscation

Continued on the next page

Table 1.1 – continued from the previous page

No.	Description of method
36	(Padilha and Pedone, 2015): The most common way to achieve obfuscation is to employ cryptographic approaches. Secret sharing is one practical example, in this regard.
37	(Gao-xiang et al., 2013): This paper proposes achieving the obfuscation through homomorphic encryption.
38	(Furukawa et al., 2013): This paper studies point function obfuscation which relies on one way hash functions.
	Category 8: Diversification
39	(Tunc et al., 2014): Moving target defences aim at continuously altering the execution environment of the system and its configurations, in order to make it challenging and costly for the intruder to learn about the environment and discover its vulnerabilities. This paper proposes diversification of the cloud's execution environment.
40	(Yang et al., 2014): This paper proposes to continuously change the execution environment and also the platforms used to execute them. Hence, till the time that the attacker learns the execution environment, it has changed. Moreover, hardware redundancy is introduced as a way to increase the tolerance to the attacks.
41	(Guo and Bhattacharya, 2014): Design diversity is proposed in this paper for the cloud infrastructure. The target of the diversification is the configuration of virtual replicas. This increases the resiliency of the service, in case of possible attacks.
	Category 9: Cloud security through securing the browser
42	(Prasadreddy et al., 2011): This paper proposes a plug-in for the user's web browser that offers double authentication and hybrid obfuscation for the data, and protects the security and privacy of the cloud in this way.
43	(Palanques et al., 2012): In this work, obfuscation is used to extend the session's life time.

1.4 Enhancing the security of cloud computing using obfuscation and diversification

1.4.1 Motivation behind our idea

As discussed in Section 1.2.1 about the importance of application security in cloud computing environment and the big losses that might happen as the consequence of insecure applications, we were motivated to propose an approach to improve the cloud's security through securing the applications. Considering the fact that obfuscation and diversification techniques have

shown success in impeding the malware in various domains to lessen the risk of harmful damage, we were motivated to use this techniques in our approach. To this aim, first we investigated "how these two techniques are used in cloud computing with the goal of boosting security" (Hosseinzadeh et al., 2015). We systematically reviewed all the studies that were trying to answer this research question. By answering this question, we were aiming at identifying the research gaps which lead us in our future research. After collecting and analyzing the data, we concluded that: there is a growing interest in this field of study, as the number of publications was increasing year by year. Obfuscation and diversification techniques have been used in the literature in different ways, that we presented a classification of this studies based on the way they use these techniques. The classification is presented in Section 1.3.1. Furthermore, as the result of this survey, we realized that the majority of the studied works have proposed approaches using obfuscation techniques, and few were focusing on diversification techniques. This implies that there is a room for more research on the use of diversification as a beneficial technique to bring security to cloud computing.

The previous survey shed more light on the areas that are still potential targets for further research, which motivated us to propose an efficient approach with the help of diversification and obfuscation techniques to secure the cloud's applications. We discuss the details of the proposed approach in Sections 1.4.3.

1.4.2 Threat model

To make using and deploying SaaS applications easy, these applications are usually available in web environment. A significant proportion of the code of these applications is usually run on the client side, which makes them vulnerable to client-side attacks. Also, the client-side interfaces are often a natural weak point that an adversary can utilize to launch an attack. In what follows, we will concentrate on this threat.

In a man-in-the-browser attack (MitB), the adversary has successfully compromised the client's endpoint application, usually the browser, by getting malware into user's system. The malware can then modify how the browser represents certain web sites and how the user can interact with them. Because the malware is operating inside user's browser, it is able to perform actions using user's authentication credentials by exploiting active log-in sessions (Gühring, 2006; Laperdrix et al., 2015).

To be more specific, the malicious program infects the computer's software. The malware – often implemented as a browser extension – then waits for the user to submit some interesting data. As the data is input in the application, the malware intercepts this delivery and extracts all the data using the interfaces provided by the browser (usually by accessing the DOM

interface using JavaScript) and stores the values. The malware then modifies the values using browser's interface. The malware then tells the browser to continue submitting the data to the server (or just store it locally in the web application) and the browser goes on without knowing the data has been tampered with. The modified values are now stored by the server (or locally), but neither the user or the server knows that they are not the original values.

In the case the server generates a receipt of the performed transaction or otherwise shows the previously sent values to the user, the malware again transforms them to the original ones. The user thinks everything is fine, because it appears that the original transaction was received and stored intact. In reality, however, the stored values have been fabricated by the malicious adversary.

It is important to note that attacks of this kind have been seen in the wild (Binsalleeh et al., 2010) and there is no completely satisfactory solution to prevent them. Therefore, mitigating these attacks has become an important goal.

1.4.3 Our proposed approach

For this work, we decided to evaluate integrating source code level obfuscation into an existing web application written in JavaScript. Our solution is a proactive and transparent method that protects applications from data manipulation. Although it does not guarantee to completely prevent all tampering, it significantly mitigates the attack scenario we described.

An important key observation in our solution is the fact that a malicious program in the user's browser needs knowledge about the web application's internal structure in order modify the data provided by the user. We therefore, change the application that is being executed on the user's web browser in a way that will make it very difficult for a harmful program to compromise it.

After we have applied unique obfuscation to the program, the code is unique on each user's computer. Generic and automatic large-scale malware attacks become infeasible, since the adversary needs to know what to change in the target application's code.

Given enough time, however, the attacker may be able to break the obfuscation. Taking this possibility into account, we could make attacking the web application even harder by continuously re-obfuscating it during its execution. As the internal structure of the web application is dynamically changed like this, a malicious program has only a short time to analyze it in order to modify the data.

In web environment, certain obfuscation methods can also be used to obfuscate the HTML code on the web page that is the target of protection. This makes it even harder for a piece of malware to attach itself to the web application (for example, by using known attribute names of HTML elements). In

is also worth noting that in our scheme, we scramble HTML and JavaScript code but not to the actual data that is transmitted over the network. The usual cryptographic protocols like Transport Layer Security (TLS) (Dierks, 2008), are still applied to this data on most web pages handling private data.

It is worth noting that when the obfuscation has been performed, the user of a web application will not notice any changes in the functionality of the application. Obfuscation is transparent to the user. We also want our solution to be transparent for the web application developer. The obfuscation is performed automatically after the code is written so the developer does not have to worry about it.

Data modification attacks are often highly dependent on the known structure of a web application. For example, the adversary might try to edit some function in the JavaScript code based on its known name. Our approach should therefore effectively mitigate these kinds of attacks by obscuring the structure of executable code.

1.4.4 Choice of application

For the choice of application, we had the following criteria that the selected application had to fulfill:

1. Availability of production-ready obfuscation tooling and libraries. It can be argued, that source code level obfuscation tooling is still in its infancy and, at least in our experience gathered from this exercise, such tools are simply non-existent for many languages and environments. However, for some languages and environments – like JavaScript run in the user's browser – several obfuscation tools and libraries exist today.
2. The application had to be implemented using technologies that are common in today's web development environment. Using commonplace technologies was especially important because we wanted the experiences to be applicable to real-world web application deployment scenarios. In short, we wanted our choice of application be representative of a generic web application.

In the end, we decided to obfuscate Laverna (lav, 2016), a simple note taking application that relies entirely for client side scripting for its functionality. Since Laverna contains essentially no server-side components, the main security risk it faces comes from man-in-the-browser attacks.

1.4.5 Implementation

Ideally, we would like the obfuscation to be seamlessly integrated into project's development work-flow. It is common for modern web-applications already contain a complex build process: resource compression, source code transpiling and request count optimization are just few of the steps that a typical application might employ. Orchestrating all these interdependent operations is a challenging task that has given a rise for a cornucopia of different build automation tools targeting the web platforms.

Laverna is not an exception on this front. The project makes heavy use of Gulp (gul, 2016a), Browserify (bro, 2016), and npm (npm, 2016) to automate its build process and manage the complex web of dependencies required for building the application. As a part of the standard build process, Gulp transpiles stylesheets written in Less (les, 2016) into CSS, compresses the HTML, produces caching manifest and runs various code quality checkers on the project.

We wanted the obfuscation to be as transparent to the software developer as possible. To this end, we decided to integrate the source code obfuscation as an additional step in Gulp's project build specification. Using common tools for the deployment process served our overall goal: evaluating the real-world challenges related to deploying obfuscation.

The concrete obfuscation implementation is composed of three third-party components: `gulp-js-obfuscator` (gul, 2016b), `js-obfuscator` (jso, 2016), and the service provided by `javascriptobfuscator.com` (jav, 2016). The last of which, provides the actual source code transformations in a *software-as-a-service* like manner. `js-obfuscate` implements a programmatic API around the the service and `gulp-js-obfuscate` provides integration with Gulp's build pipeline architecture.

The results of the diversification experiment we performed on Laverna applications with our tool indicate that the program would indeed be much more difficult to understand and tamper with after diversification has been applied. For example, the average Halstead difficulty (the difficulty of understanding a given program) was X for the functions of the diversified version of the program, compared to Y for the original code. Naturally, even better results would be achieved with a framework that would use a larger set of even more resilient obfuscation transformations.

1.4.6 Limitations of the approach

Tooling for analyzing errors in program code is obviously important from a software development standpoint and when it comes to web development, most popular browsers come with built-in debugging capabilities. Setting break-points, single-stepping through the program code, and inspecting ob-

```
74\x49\x74\x65\x6D"];define([_0xa6ab[0],_0xa6ab[1],_0xa6ab[2]],
function(_0x27cax1,_0x27cax2,_0x27cax3){_0xa6ab[3];var _0x27cax4
={dbs:{},getDb:function(_0x27cax5){var _0x27cax6=_0x27cax5[
_0xa6ab[4]]+_0xa6ab[5]+_0x27cax5[_0xa6ab[6]];this[_0xa6ab[7]][
_0x27cax6]=this[_0xa6ab[7]][_0x27cax6]||_0x27cax3[_0xa6ab[9]]({
name:_0x27cax5[_0xa6ab[4]]||_0xa6ab[8],storeName:_0x27cax5[
_0xa6ab[6]]});return this[_0xa6ab[7]][_0x27cax6]},find:function(
_0x27cax7){var _0x27cax8=_0x27cax2[_0xa6ab[10]]();this[_0xa6ab
[17]](_0x27cax7[_0xa6ab[16]])[_0xa6ab[15]](_0x27cax7[_0xa6ab
[11]],function(_0x27cax9,_0x27cax7){if(_0x27cax9){return
_0x27cax8[_0xa6ab[12]](_0x27cax9)};if(!_0x27cax7){_0x27cax8[
_0xa6ab[12]](_0xa6ab[13])};return _0x27cax8[_0xa6ab[14]](
_0x27cax7)});return _0x27cax8[_0xa6ab[18]]},findAll:function(
_0x27cax7){var _0x27cax8=_0x27cax2[_0xa6ab[10]](),_0x27caxa=this;
this[_0xa6ab[17]](_0x27cax7[_0xa6ab[16]])[_0xa6ab[23]](function(
_0x27cax9,_0x27caxb){if(!_0x27caxb||!_0x27caxb[_0xa6ab[19]]){
return _0x27cax8[_0xa6ab[14]]([])};
```

Fig. 1.4 Excerpt from obfuscated piece of JavaScript code.

jects are common requirements. Unfortunately, application of source code level obfuscation makes utilizing available tooling challenging, to say the least. The problem arises because the developer interacts with the original, unobfuscated source code, but the browser only has access to the obfuscated version of the code. This is not a problem that only affects obfuscation related tooling, source-to-source transpilers have long faced similar problems. However, it could be argued that the problem is magnified for by the very nature of obfuscation, desire to make programs harder to understand.

Source maps is a technique created to solve the aforementioned problem of debugging (sou, 2016). Source maps provide the browser with auxiliary debugging information about the obfuscated scripts, allowing it to map the executed statements to statements in the original source. Unfortunately, our current setup did not provide support for source maps. This problem can be somewhat remedied by applying obfuscation only to release builds of the software. While this approach works, with the added benefit of making the build process faster, it does not allow analyzing problems that might arise due to the application of obfuscation. It also limits software developer's ability to analyze possible error reports from end users.

When obfuscating any application, preserving good performance is also an important goal and a challenge. Because of the requirement for transparency, large performance losses clearly noticed by the user are not acceptable. As a response to increasing use of JavaScript frameworks and ongoing competition between web browser manufacturers, performances of the JavaScript engines have gone up in recent years. Acceptable performance and good transparency

to the user are usually feasible goals even when using several obfuscation techniques in combination and the obfuscation is dynamically changed.

Employing obfuscation also often increases bandwidth consumption as the executable code grows longer. Dynamically updating the code during execution – a feature not implemented in our current proof-of-concept implementation – would also significantly increase the network traffic. All obfuscation techniques do not increase the size of code that much, though. For example, simply renaming functions does not really affect the bandwidth consumption.

The SaaS-based obfuscation backend provided by `javascriptobfuscator.com` supports a number of obfuscating transformations. The basic settings employ standard techniques such as variable renaming and string encoding, but more complicated transformation options are also available. Still, we felt that the service-oriented solution limited the amount of control over how the code was to be modified. Figure 1.4 gives an idea of what the end result of the obfuscation looks like.

1.5 Conclusion

Cloud computing is becoming an essential part of today's Information Technology. Almost all enterprises and businesses, in all sizes, have deployed (or are planning to deploy) cloud solutions for delivering their services to customers. Cloud adoption is accelerating because of the advantages that cloud computing has brought along, such as higher flexibility and capability of the infrastructures, lower costs of operation and maintenance, wider accessibility, and improved mobility and collaboration (Mather et al., 2009).

Despite of all these benefits, there are still barriers in turning into cloud. Among all, security of the data is the primary concern that holds back the projects from moving to the cloud. The cloud's security threats can be classified in different ways. Cloud Security Alliance (CSA) presented a list of top threats targeting the cloud computing environment (CSA, 2016; Top Threats Working Group, 2013). In Section 1.2, we went through the security concerns of the cloud and also security aspects that need to be taken into account in cloud computing environment. We discussed that there are three main delivery models for delivering the cloud services (IaaS, PaaS, and SaaS) that each require different levels of security (Rhoton et al., 2013).

In Section 1.3, first we presented the terms and techniques used in our proposed security approach. Obfuscation and diversification are techniques that have been used to secure the software, mainly with the aim of impeding malware. These techniques have been utilized in various domains as well as in cloud computing. In a previous study we conducted a thorough survey to investigate in what way these two techniques have been previously used to enhance the security of cloud computing and protect the privacy of its users (Hosseinzadeh et al., 2015). As the result of this study, we managed to

identify research gaps that motivated us to demonstrate an approach, which fills the gaps to some extent and improves the security in cloud efficiently.

In Section 1.4 we demonstrated an obfuscation (partly including diversification) approach for mainly securing the SaaS model in cloud computing. In this approach we obfuscated the client-side JavaScript components of an application, we did this to demonstrate the feasibility of applying obfuscation in the real-world. We built our solution using existing tools and services to evaluate the experience of integrating obfuscation into an existing application. Implementing the obfuscation only required a relatively small amount of work, mostly because of the use of ready-made libraries. However, the amount of work required is likely to be highly dependant on the complexity of one's target application and the thoroughness of applied obfuscation.

References

Cloud security alliance (CSA): https://cloudsecurityalliance.org/, 2016. Verified 2016-04-08.

The International Information Systems Security Certification Consortium ($(ISC)^2$: https://www.isc2.org/, 2016. Verified 2016-04-06.

Browserify. `http://browserify.org`, 2016.

gulp.js - the streaming build system. `http://gulpjs.com`, 2016a.

gulp-js-obfuscator. `https://www.npmjs.com/package/gulp-js-obfuscator`, 2016b.

Free javascript obfuscator - protect javascript code from stealing and shrink size. `https://javascriptobfuscator.com`, 2016.

js-obfuscator. `https://www.npmjs.com/package/js-obfuscator`, 2016.

Laverna - keep your notes private. `https://laverna.cc`, 2016.

Getting started — less.js. `http://lesscss.org`, 2016.

npm. `https://www.npmjs.com`, 2016.

Source Map Revision 3 Proposal. `https://docs.google.com/document/d/1U1RGAehQwRypUTovF1KRlpiOFze0b-_2gc6fAH0KY0k`, 2016.

B. Agir, T. Papaioannou, R. Narendula, K. Aberer, and J.-P. Hubaux. User-side adaptive protection of location privacy in participatory sensing. *GeoInformatica*, 18(1):165–191, 2014.

L. Arockiam and S. Monikandan. Efficient cloud storage confidentiality to ensure data security. In *Computer Communication and Informatics (ICCCI), 2014 International Conference on*, pages 1–5, Jan 2014.

B. Baudry and M. Monperrus. The multiple facets of software diversity: Recent developments in year 2000 and beyond. *ACM Comput. Surv.*, 48 (1):16:1–16:26, Sept. 2015. ISSN 0360-0300.

B. Bertholon, S. Varrette, and P. Bouvry. Jshadobf: A javascript obfuscator based on multi-objective optimization algorithms. In J. Lopez, X. Huang, and R. Sandhu, editors, *Network and System Security*, volume 7873 of

Lecture Notes in Computer Science, pages 336–349. Springer Berlin Heidelberg, 2013a.

B. Bertholon, S. Varrette, and S. Martinez. Shadobf: A c-source obfuscator based on multi-objective optimisation algorithms. In *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2013 IEEE 27th International*, pages 435–444, May 2013b.

B. Bertholon, S. Varrette, and P. Bouvry. Comparison of multi-objective optimization algorithms for the jshadobf javascript obfuscator. In *Parallel Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International*, pages 489–496, May 2014.

H. Binsalleeh, T. Ormerod, A. Boukhtouta, P. Sinha, A. Youssef, M. Debbabi, and L. Wang. On the analysis of the zeus botnet crimeware toolkit. In *Proceedings of the 8th Annual International Conference on Privacy, Security and Trust (PST)*, pages 31–38. IEEE, 2010.

A. Celesti, M. Fazio, M. Villari, and A. Puliafito. Adding long-term availability, obfuscation, and encryption to multi-cloud storage systems. *Journal of Network and Computer Applications*, 2014.

T. M. Chen and S. Abu-Nimeh. Lessons from stuxnet. *Computer*, 44(4): 91–93, April 2011.

F. B. Cohen. Operating System Protection through Program Evolution. *Comput. Secur.*, 12(6):565–584, Oct. 1993.

C. Collberg, C. Thomborson, and D. Low. A taxonomy of obfuscating transformations. Technical report, Department of Computer Science, The University of Auckland, New Zealand, 1997.

C. Collberg, C. Thomborson, and D. Low. Manufacturing cheap, resilient, and stealthy opaque constructs. In *Proceedings of the 25th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '98, pages 184–196, New York, NY, USA, 1998. ACM.

T. Dierks. The transport layer security (tls) protocol version 1.2. 2008.

S. Drape and A. Majumdar. Design and evaluation of slicing obfuscation. Technical report, Department of Computer Science, The University of Auckland, New Zealand, 2007.

R. Furukawa, T. Takenouchi, and T. Mori. Behavioral tendency obfuscation framework for personalization services. In H. Decker, L. Lhotsk, S. Link, J. Basl, and A. Tjoa, editors, *Database and Expert Systems Applications*, volume 8056 of *Lecture Notes in Computer Science*, pages 289–303. Springer Berlin Heidelberg, 2013.

G. Gao-xiang, Y. Zheng, and F. Xiao. The homomorphic encryption scheme of security obfuscation. In T. Tan, Q. Ruan, X. Chen, H. Ma, and L. Wang, editors, *Advances in Image and Graphics Technologies*, volume 363 of *Communications in Computer and Information Science*, pages 127–135. Springer Berlin Heidelberg, 2013.

K. Govinda and E. Sathiyamoorthy. Agent based security for cloud computing using obfuscation. *Procedia Engineering(38),125-129*, 2012.

P. Gühring. Concepts against Man-in-the-Browser Attacks. `www.cacert.at/svn/sourcerer/CAcert/SecureClient.pdf`, 2006.

M. Guo and P. Bhattacharya. Diverse virtual replicas for improving intrusion tolerance in cloud. In *Proceedings of the 9th Annual Cyber and Information Security Research Conference*, CISR '14, pages 41–44, New York, NY, USA, 2014. ACM.

M. Hataba and A. El-Mahdy. Cloud protection by obfuscation: Techniques and metrics. In *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2012 Seventh International Conference on*, pages 369–372, Nov 2012.

S. Hosseinzadeh, S. Hyrynsalmi, M. Conti, and V. Leppänen. Security and privacy in cloud computing via obfuscation and diversification: A survey. In *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 529–535, Nov 2015.

K. Kansal, M. Mohanty, and P. K. Atrey. Scaling and cropping of wavelet-based compressed images in hidden domain. In X. He, S. Luo, D. Tao, C. Xu, J. Yang, and M. Hasan, editors, *MultiMedia Modeling*, volume 8935 of *Lecture Notes in Computer Science*, pages 430–441. Springer International Publishing, 2015.

K. Karuppanan, K. AparnaMeenaa, K. Radhika, and R. Suchitra. Privacy adaptation for secured associations in a social cloud. In *Advances in Computing and Communications (ICACC), 2012 International Conference on*, pages 194–198, Aug 2012.

M. Kuzu, M. S. Islam, and M. Kantarcioglu. Efficient privacy-aware search over encrypted databases. In *Proceedings of the 4th ACM Conference on Data and Application Security and Privacy*, CODASPY '14, pages 249–256, New York, NY, USA, 2014. ACM.

D. Lamanna, G. Lodi, and R. Baldoni. How not to be seen in the cloud: A progressive privacy solution for desktop-as-a-service. In R. Meersman, H. Panetto, T. Dillon, S. Rinderle-Ma, P. Dadam, X. Zhou, S. Pearson, A. Ferscha, S. Bergamaschi, and I. Cruz, editors, *On the Move to Meaningful Internet Systems: OTM 2012*, volume 7566 of *Lecture Notes in Computer Science*, pages 492–510. Springer Berlin Heidelberg, 2012.

P. Laperdrix, W. Rudametkin, and B. Baudry. Mitigating browser fingerprint tracking: Multi-level reconfiguration and diversification. In *Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2015 IEEE/ACM 10th International Symposium on*, pages 98–108, May 2015.

P. Larsen, A. Homescu, S. Brunthaler, and M. Franz. SoK: Automated software diversity. In *Security and Privacy (SP), 2014 IEEE Symposium on*, pages 276–291, May 2014.

S. Laurén, P. Mäki, S. Rauti, S. Hosseinzadeh, S. Hyrynsalmi, and V. Leppänen. Symbol Diversification of Linux Binaries. In *Proceedings of World Congress on Internet Security (WorldCIS-2014)*, 2014.

L. Li, Q. Li, Y. Shi, and K. Zhang. A new privacy-preserving scheme dospa for saas. In Z. Gong, X. Luo, J. Chen, J. Lei, and F. Wang, editors, *Web In-*

formation Systems and Mining, Lecture Notes in Computer Science(6987), pages 328–335. Springer Berlin Heidelberg, 2011.

C. Linn and S. Debray. Obfuscation of executable code to improve resistance to static disassembly. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, CCS '03, pages 290–299, New York, NY, USA, 2003. ACM.

X. Liu, D. Yuan, G. Zhang, W. Li, D. Cao, Q. He, J. Chen, and Y. Yang. Cloud workflow system quality of service. In *The Design of Cloud Workflow Systems*, SpringerBriefs in Computer Science, pages 27–50. Springer New York, 2012.

T. Mather, S. Kumaraswamy, and S. Latif. *Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance*. Theory in Practice. O'Reilly Media, Inc., Sebastopol, CA, 2009.

P. Mell and T. Grance. The NIST definition of cloud computing. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, 2011.

M. Mowbray and S. Pearson. A client-based privacy manager for cloud computing. In *Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE*, COMSWARE '09, pages 5:1–5:8, New York, NY, USA, 2009. ACM.

M. Mowbray, S. Pearson, and Y. Shen. Enhancing privacy in cloud computing via policy-based obfuscation. *The Journal of Supercomputing*, 61(2):267–291, 2012.

J. Nagra and C. Collberg. *Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection*. Pearson Education, 2009.

R. Omar, A. El-Mahdy, and E. Rohou. Arbitrary control-flow embedding into multiple threads for obfuscation: A preliminary complexity and performance analysis. In *Proceedings of the 2Nd International Workshop on Security in Cloud Computing*, SCC '14, pages 51–58, New York, NY, USA, 2014. ACM.

R. Padilha and F. Pedone. Confidentiality in the cloud. *Security Privacy, IEEE*, 13(1):57–60, Jan 2015.

M. Palanques, R. DiPietro, C. del Ojo, M. Malet, M. Marino, and T. Felguera. Secure cloud browser: Model and architecture to support secure web navigation. In *Reliable Distributed Systems (SRDS), 2012 IEEE 31st Symposium on*, pages 402–403, Oct 2012.

R. Patibandla, S. Kurra, and N. Mundukur. A study on scalability of services and privacy issues in cloud computing. In R. Ramanujam and S. Ramaswamy, editors, *Distributed Computing and Internet Technology*, volume 7154 of *Lecture Notes in Computer Science*, pages 212–230. Springer Berlin Heidelberg, 2012.

S. Pearson, Y. Shen, and M. Mowbray. A privacy manager for cloud computing. In M. Jaatun, G. Zhao, and C. Rong, editors, *Cloud Computing*, volume 5931 of *Lecture Notes in Computer Science*, pages 90–106. Springer Berlin Heidelberg, 2009.

I. V. Popov, S. K. Debray, and G. R. Andrews. Binary obfuscation using signals. In *USENIX Security*, 2007.

P. Prasadreddy, T. Rao, and S. Venkat. A threat free architecture for privacy assurance in cloud computing. In *Services (SERVICES), 2011 IEEE World Congress on*, pages 564–568, July 2011.

Y. Qin, S. Shen, J. Kong, and H. Dai. Cloud-oriented sat solver based on obfuscating cnf formula. In W. Han, Z. Huang, C. Hu, H. Zhang, and L. Guo, editors, *Web Technologies and Applications*, volume 8710 of *Lecture Notes in Computer Science*, pages 188–199. Springer International Publishing, 2014.

S. Rauti, S. Laurén, S. Hosseinzadeh, J.-M. Mäkelä, S. Hyrynsalmi, and V. Leppänen. Diversification of System Calls in Linux Binaries. In *Proceedings of the 6th International Conference on Trustworthy Systems (InTrust 2014)*, 2014.

C. Reiss, J. Wilkes, and J. Hellerstein. Obfuscatory obscanturism: Making workload traces of commercially-sensitive systems safe to release. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 1279–1286, April 2012.

J. Rhoton, J. de Clercq, and D. Graves. *Cloud Computing Protected: Security Assessment Handbook*. Security Assessment Handbook. Recursive, Limited, 2013.

P. Ryan and S. Falvey. Trust in the clouds. *Computer Law & Security Review*, 28(5):513 – 521, 2012.

E. Skoudis. *Malware: Fighting malicious code*. Prentice Hall Professional, 2004.

P. Skvortsov, F. Drr, and K. Rothermel. Map-aware position sharing for location privacy in non-trusted systems. In J. Kay, P. Lukowicz, H. Tokuda, P. Olivier, and A. Krger, editors, *Pervasive Computing*, volume 7319 of *Lecture Notes in Computer Science*, pages 388–405. Springer Berlin Heidelberg, 2012.

S. Subashini and V. Kavitha. A survey on security issues in service delivery models of cloud computing. *Journal of Network and Computer Applications*, 34(1):1 – 11, 2011.

J. Tapiador, J. Hernandez-Castro, and P. Peris-Lopez. Online randomization strategies to obfuscate user behavioral patterns. *Journal of Network and Systems Management*, 20(4):561–578, 2012.

Y. Tian, B. Song, and E.-N. Huh. Towards the development of personal cloud computing for mobile thin-clients. In *International Conference Information Science and Applications (ICISA)*, pages 1–5, April 2011.

Top Threats Working Group. The notorious nine: cloud computing top threats in 2013. *Cloud Security Alliance*, 2013.

C. Tunc, F. Fargo, Y. Al-Nashif, S. Hariri, and J. Hughes. Autonomic resilient cloud management (arcm) design and evaluation. In *Cloud and Autonomic Computing (ICCAC), 2014 International Conference on*, pages 44–49, Sept 2014.

V. Varadharajan and U. Tupakula. Security as a service model for cloud environment. *IEEE Transactions on Network and Service Management*, 11(1):60–75, March 2014.

M. Villari, A. Celesti, F. Tusa, and A. Puliafito. Data reliability in multi-provider cloud storage service with rrns. In C. Canal and M. Villari, editors, *Advances in Service-Oriented and Cloud Computing*, volume 393 of *Communications in Computer and Information Science*, pages 83–93. Springer Berlin Heidelberg, 2013.

M. Vleju. A client-centric asm-based approach to identity management in cloud computing. In S. Castano, P. Vassiliadis, L. Lakshmanan, and M. Lee, editors, *Advances in Conceptual Modeling*, volume 7518 of *Lecture Notes in Computer Science*, pages 34–43. Springer Berlin Heidelberg, 2012.

P. Yang, X. Gui, F. Tian, J. Yao, and J. Lin. A privacy-preserving data obfuscation scheme used in data statistics and data mining. In *High Performance Computing and Communications 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC-EUC), 2013 IEEE 10th International Conference on*, pages 881–887, Nov 2013.

Q. Yang, C. Cheng, and X. Che. A cost-aware method of privacy protection for multiple cloud service requests. In *Computational Science and Engineering (CSE), 2014 IEEE 17th International Conference on*, pages 583–590, Dec 2014.

S. S. Yau and H. G. An. Protection of users' data confidentiality in cloud computing. In *Proceedings of the Second Asia-Pacific Symposium on Internetware*, Internetware '10, pages 11:1–11:6, New York, NY, USA, 2010. ACM.

G. Zhang, Y. Yang, and J. Chen. A historical probability based noise generation strategy for privacy protection in cloud computing. *Journal of Computer and System Sciences*, 78(5):1374 – 1381, 2012a. {JCSS} Special Issue: Cloud Computing 2011.

G. Zhang, Y. Yang, X. Liu, and J. Chen. A time-series pattern based noise generation strategy for privacy protection in cloud computing. In *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, pages 458–465, May 2012b.

G. Zhang, Y. Yang, D. Yuan, and J. Chen. A trust-based noise injection strategy for privacy protection in cloud. *Software: Practice and Experience*, 42(4):431–445, 2012c.

G. Zhang, X. Zhang, Y. Yang, C. Liu, and J. Chen. An association probability based noise generation strategy for privacy protection in cloud computing. In C. Liu, H. Ludwig, F. Toumani, and Q. Yu, editors, *Service-Oriented Computing*, volume 7636 of *Lecture Notes in Computer Science*, pages 639–647. Springer Berlin Heidelberg, 2012d.

G. Zhang, Y. Yang, and J. Chen. A privacy-leakage-tolerance based noise enhancing strategy for privacy protection in cloud computing. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 12th IEEE International Conference on*, pages 1–8, July 2013.

G. Zhang, X. Liu, and Y. Yang. Time-series pattern based effective noise generation for privacy protection on cloud. *Computers 64(5), IEEE Transactions on*, pages 1456–1469, May 2015.