How to Use Garbling for Privacy Preserving Electronic Surveillance Services

Tommi Meskanen[1], Valtteri Niemi[1] and Noora Nieminen[1,2]

[1]*Department of Mathematics and Statistics, University of Turku, 20014 Turun yliopisto, FINLAND*
[2]*Turku Centre for Computer Science (TUCS), FINLAND*
Corresponding Authors: {tommes; pevani; nmniem}@utu.fi

Received 15 September 2014; Accepted 17 April 2015;
Publication 22 May 2015

Abstract

Various applications following the Internet of Things (IoT) paradigm have become a part of our everyday lives. Therefore, designing mechanisms for security, trust and privacy for this context is important. As one example, applications related to electronic surveillance and monitoring have serious issues related to privacy. Research is needed on how to design privacy preserving surveillance system consisting of networked devices. One way to implement privacy preserving electronic surveillance is to use tools for multiparty computations. In this paper, we present an innovative way of using garbling, a powerful cryptographic primitive for secure multiparty computation, to achieve privacy preserving electronic surveillance. We illustrate the power of garbling in a context of a typical surveillance scenario. We discuss the different security measures related to garbling as well as efficiency of garbling schemes. Furthermore, we suggest further scenarios in which garbling can be used to achieve privacy preservation.

Keywords: Internet of Things, privacy, electronic surveillance, garbling schemes.

Journal of Cyber Security, Vol. 4, 41–64.
doi: 10.13052/jcsm2245-1439.413

1 Introduction

Nowadays, we are surrounded by an increasing variety of *things* or *objects* that are connected with each other and accessible through the Internet. This trend is a consequence of a novel paradigm, Internet of Things (IoT), in which the devices form a network configured to reach goals common to all devices. The paradigm itself has gained increasing interest after the introduction of technologies that enable computing-like devices to share their states through the common network. These technologies include *Radio-frequency identification tags* (RFID) [10], *Near-field communication* (NFC) techniques and *Wireless sensor and actuator network* (WSAN) [27]. As an example of a network of devices trying to reach a common goal, consider an anti-theft system with motion detecting sensors. The sensors located differently interact with each other in order to detect unauthorized motion and prevent intruders. Many other applications of IoT can be found in [3, 27]. Some of the applications mentioned in [3] have been collected into Figure 1.

1.1 Related Research on Security

The technological advances alone are not sufficient to guarantee success for IoT-based solutions – the security of the technology is an important aspect as well. There are a variety of security threats related to IoT, as Roman et al. show in [26]: The threats are targeted at infrastructure, protocol and network security, data and privacy, identity management, trust and governance as well as at fault tolerance. For example, current Internet protocols may not meet the

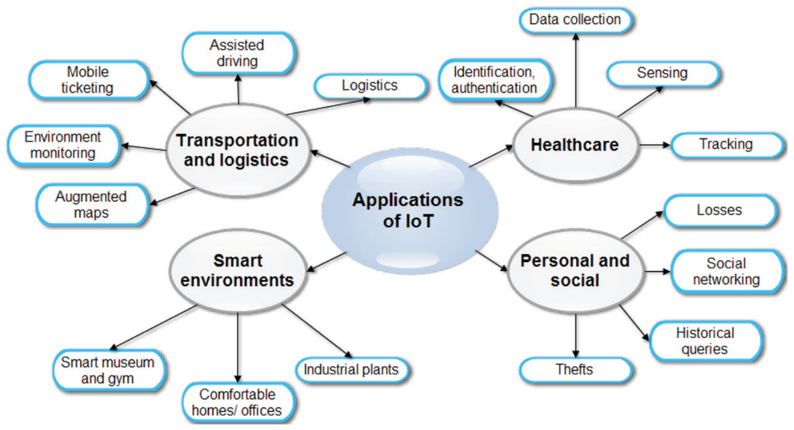

Figure 1 Applications of the Internet of Things (adapted from [3]).

security requirements of IoT, especially in the IP-based IoT as Heer et al. show in [18]. The physical security of IoT devices, e.g. tamper-resistance, is also an important aspect. An overview of different threats and possible solutions can be found in [27], whereas a more detailed threat analysis of RFID can be found in [10] and analyses of NFC from [17].

Several security threats are also identified by Kozlov et al. in [20]: there are numerous scenarios which endanger the security, trust or privacy of the IoT and these issues must be taken into account when considering legislation related to the Internet of Things. According to Weber [28], the IoT technology used by private enterprises must have resilience to attacks, authenticate the retrieved address and object information, have an access control and ensure client privacy. The privacy concerns are notable in situations in which the actions of individuals are monitored in a privacy-sensitive context. For example, a failed implementation of IoT related technology in a supermarket may violate the client privacy by enabling “the mining of medical data, invasive targeted advertising, and loss of autonomy through marketing profiles or personal affect monitoring” [29]. However, innovative ways of deploying privacy preserving IoT in privacy sensitive environments successfully are also possible: Abie et al. consider risk-based adaptive security framework for IoT in eHealth in [2]. More generally, techniques to achieve privacy preserving IoT applications have been considered widely. For example, privacy preserving electronic surveillance [11, 24] and even privacy preserving data mining [8] are possible by using a set of powerful cryptographic methods, called *secure multiparty computation* (SMPC).

1.2 Our Contributions

The solutions to achieve SMPC include a variety of protocols, e.g. *oblivious transfer* [25], *secure sum protocols* [8] and *garbled circuits* [30]. In this paper, we consider a way of achieving privacy preserving IoT applications by applying SMPC protocols. More specifically, we introduce a new way to realize privacy preserving electronic surveillance.We present a new tool in this context, *garbling*, which enables private computation on encrypted data.

The paper is organized as follows. We demonstrate the power of garbling in an example scenario presented in Section 2. In Section 3, we describe the realization of the privacy preserving electronic surveillance. In Section 4, we analyze the novel application of garbling in more details by considering its efficiency and what kinds of security goals are achieved by this technique. Section 5 concludes the paper and proposes directions for future research related to privacy preserving electronic surveillance.

2 Problem Setting: Privacy Preserving Electronic Surveillance System

Electronic surveillance is an application where privacy is a central concern. Many cryptographic tools have been proposed to ensure at least some level of privacy. In this paper, we present an innovative way of achieving privacy by using garbling in the context of electronic surveillance. Let us consider the following scenario as an example.

The client in this scenario is an elderly person living alone who wants to use the security service provided by a security company. The security company bases its service on an electronic surveillance system consisting of *Closed-Circuit Televisions* (CCTV) and various sensors (for example, motion detectors and/or sensors measuring the activity of the client). The security company collects data obtained by the system for further analysis. The analysis process contains tools for *data mining*, *pattern recognition* and *machine learning* – the intelligent surveillance system is supervised to react correctly on different situations. In certain situations (for instance, when the ongoing event seems to differ significantly from the usual course of events), the system evokes an alarm. The alarm together with an assessment of the situation enables the security company to react appropriately to the situation (e.g. call police/ambulance, send a guard from the company or just notify the client). The security company has outsourced its data center services into *a cloud* managed by a third-party company. The data from the surveillance system is stored and analyzed entirely in the cloud environment.

The main concern in this scenario is how the privacy for the client is managed. First obvious requirement is that anyone beyond the client, the security company and the cloud should not learn the contents of the data collected by the surveillance system. This requirement can be reached by simply encrypting the data on the client side and decrypting the data on the security company side. As a consequence, the security company and the cloud provider can follow everything that is going on at the client's home. A serious concern is that the third-party company managing the cloud can learn something about the client that could be used for unwanted or even malicious purposes. Thus it is highly justified to hide the raw data also from the cloud whereas the security company needs the raw data to be able to react correctly in the alarming situations. A solution to this is to use two-party computation between the security company and the cloud. This would allow the cloud to analyze the surveillance data without allowing the cloud to learn the raw data or the analytics tools.

However, this solution is still problematic. The security company should monitor the surveillance data of numerous customers in real time while the analysis on cloud is ongoing. This is not desirable because of several reasons. From the company's perspective, real-time monitoring is inefficient – several employees are tied to follow the monitors and are demanded to be in alert readiness all the time, even though nothing alarming is happening. From the client's perspective, the all-time surveillance is distracting and feels privacy violating – the security company should be able to study the raw data only in alarming situations and not otherwise.

To summarize the above analysis of the scenario, the implementation of the privacy preserving electronic surveillance system should have the following properties.

Confidentiality: All the information related to the electronic surveillance is kept secret from parties excluding the client, the security company and the third-party cloud. The third-party cloud performs the analysis on encrypted data. The cloud retrieves the encrypted surveillance data from the client and the encrypted surveillance data from the security company. The cloud is not allowed to find out the unencrypted surveillance data (the data is privacy-sensitive) or the analytics tool (the tool may be intellectual property of the security company). Depending on the contract between the security company, the client and the cloud, the final analysis result can either be concealed from the cloud or can be revealed to the cloud. These alternatives are discussed in more detail in Section 4. The security company is not allowed to retrieve the unencrypted surveillance data unless the analysis result yields an alarm. The client is not allowed to learn the implementation details of the analytics tool (the tool may be intellectual property of the security company).

Integrity: We may assume that the client is honest and therefore the surveillance data is authentic. Cloud can be honest, semi-honest or even malicious – a garbling scheme achieving certain level of security (explained in Section 4) guarantees that the analysis result is also authentic. Additionally, integrity of data in transit is protected, e.g. by using message authentication codes.

Entity authentication: The cloud does not need to authenticate itself, since all the data it processes is encrypted (in the case the cloud is not allowed to find out the analysis result). The security company and the client authenticate themselves when the system is first configured. After the authentication, we assume that the channel between the client and the security company is confidential and authentic.

Access control: The security company is able to retrieve the unencrypted surveillance data only in the case in which the final analysis result yields an alarm. This requires that the analytics tools must not reveal the surveillance data. To ascertain this, the client and the security company use a trusted auditor that verifies appropriateness of the analytics tool (the analytics tool does not leak surveillance data in the final analysis report). We have described the different solutions for accessing the unencrypted surveillance data in Section 3.1.

Authorization: Access control, entity authentication and other security measures naturally require that access to various resources is properly authorized. For example, the client has to authorize the security company to have access to raw data in case of alarm and the security company has to provide authorization for the cloud provider in order to receive garbled data from the client.

Non-repudiation: We assume that the garbling protocol will achieve authenticity. This guarantees that the cloud cannot forge the garbled evaluation, and that the encrypted analysis result is authentic. We assume that the surveillance data is authentic. We also assume that the channel between the client and the security company is confidential and authentic. We also assume that the cloud service provider and the security company are not in the conspiracy against the client. Then log data collected by all parties can be used for non-repudiation purposes, see also discussion about logs in Section 3.2.

Availability: The system is naturally based on the assumption that raw data will be available for the security company in alarming situations. Related to this, there are threats purely concerning implementation. For example, burglars may cut off sending of data to the cloud. Also, the surveillance data stream may be interrupted on client side. As an example, robbers may break the CCTV equipment and sensors or the client may throw a towel on top of the surveillance camera etc.

To achieve these properties, we need an additional tool that enables the cloud to evaluate the analytics algorithms on the surveillance data without learning anything about the algorithms or data. A tool that fulfills this requirement is garbling. The formal definition of garbling and different security aspects related to garbling can be found in Section 4. In the following section, we concentrate on how the surveillance system using garbling should be implemented.

3 Operating Model: How to Build Privacy Preservation in the Surveillance System

In this section, we describe the operating model that aims at a solution for the problem presented in the previous section: How can the security company provide privacy preserving electronic surveillance to an elderly person even when all the data services of the company have been outsourced to a third-party cloud provider.

Our solution is based on a cryptographic tool for secure multiparty computation, *garbling*. Garbling enables *secure and private function evaluation*. A user who does not have enough computing resources utilizes a possibly untrustworthy evaluator, such as cloud, to accomplish the evaluation of some function f on argument x. However, the user wants to keep both the function f and the argument x secret from the evaluator. The user and the cloud agree on using a garbling scheme that works as follows. First, the user garbles function f and its argument x and obtains garbled function F and garbled argument X. The user gives F and X to the evaluator who runs the garbled evaluation to get the garbled value $Y = \mathrm{Ev}(F, X)$. Now, either the evaluator or the user ungarbles Y to get the final value y, which is equal to the result of the original evaluation $y = \mathrm{ev}(f, x)$.

In the scenario presented in the previous section, the electronic surveillance system is designed under the IoT paradigm, making the computational resources of the system limited. This means that the surveillance data analysis must take place outside the surveillance system, for example at the data center of the security company. Since the data center services are outsourced, the analysis takes place in the cloud managed by a third-party company. The surveillance data from the client's home is privacy sensitive as are the analytics tools of the security company, so the three parties agree on using a garbling scheme. Figure 2 illustrates the scenario showing also how the garbling scheme is used by the different parties.

3.1 Responsibilities of the Different Parties

The surveillance data is garbled on the client side. In this way, neither the cloud nor the security company is able to access the raw data directly. The garbled surveillance data is sent to the cloud for analysis. The security company has garbled its analytics tools that act as the function to be evaluated on cloud. After receiving both the garbled data and the garbled analytics tools, the cloud runs the garbled evaluation getting the garbled final value. If the cloud is allowed to decrypt the garbled analysis result, then it decrypts the

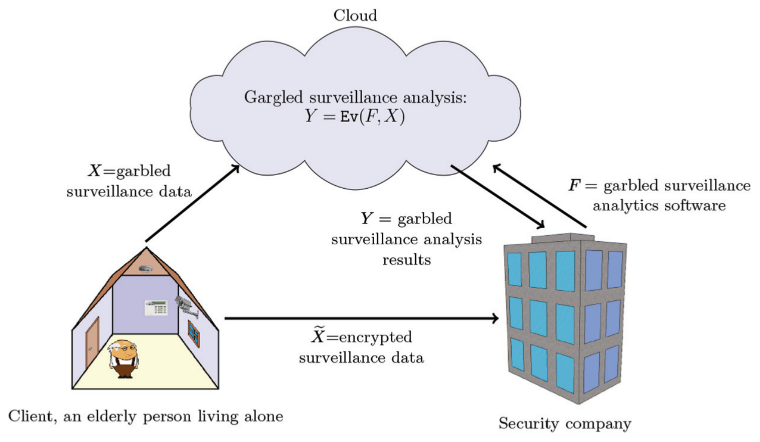
Cloud
Gargled surveillance analysis:
$Y = \mathtt{Ev}(F, X)$
X=garbled surveillance data
Y = garbled surveillance analysis results
F = garbled surveillance analytics software
$\widetilde{X}$=encrypted surveillance data
Client, an elderly person living alone
Security company

Figure 2 Scenario about electronic surveillance.

garbled value getting the final outcome of the analysis. This final outcome is sent to the security company for further investigation. However, letting the cloud learn the analysis results might not be convenient – it can violate the privacy in similar manner as the actual surveillance data. Thus, a more convenient way of implementation is that the cloud sends the garbled analysis outcome to the security company for further investigation. Now, the security company ungarbles the data received from the cloud. Based on the analysis outcome, the security company takes corresponding actions (e.g. by visiting the home or calling the police, an ambulance, a social worker, the person's relatives etc.).

A straight-forward way of reacting to the alarm situation for the security company is that a guard from the company visits the client for further inspection in spite of what has caused the alarm. Then, the security company does not have or even need an access to the raw data (in Figure 2 this means, that no encrypted surveillance data $\widetilde{X}$ is provided to the security company). However, this is not a practical approach. The company should adaptively react to different alarms – for example a robbery should cause different reactions than the client staying suspiciously long in the shower.

To adaptively react to the various situations, the security company needs an access to the raw data. Granting the security company access to the raw data with no restrictions is not a satisfactory solution since it would violate the requirements set to the system: the raw data should be accessible for the security company only in alarming situations (see Confidentiality requirement in Section 2). One possible way of realizing the access control would be to encrypt the raw data twice, independently for the cloud and for the security company. The raw data is protected against the cloud by garbling the argument x. Garbling x is modeled by encrypting x using the encryption algorithm En together with encryption key *e*. Respectively, the algorithm De with the decryption key d is used to ungarble *Y* to final value y. The encryption against the security company utilizes an independent encryption algorithm $\widetilde{\mathrm{En}}$ with key $\widetilde{e}$. The decryption algorithm $\widetilde{\mathrm{De}}$ with the decryption key $\widetilde{d}$ is used to recover the raw data from the encrypted data $\widetilde{X}$ – this key is called *recovery key* to avoid confusion between the two keys d (which is needed for ungarbling) and $\widetilde{d}$ (which is needed for recovering x from $\widetilde{X}$).

The surveillance data is collected in pieces and these pieces are then encrypted and sent to the cloud (X) and to the security company $(\widetilde{X})$ by the client. Data pieces may contain overlaps so that successful reconstruction of the course of events without gaps is possible. Since the cloud does not learn the keys (e, d), and hence learns nothing privacy – violating about the surveillance data x or the analysis result y, the same keys (e, d) may be used for many evaluations by the analytics tool.

The same does not hold for the keys ($\widetilde{e}$, $\widetilde{d}$) related to the encryption of surveillance data against the security company. If the same keys ($\widetilde{e}$, $\widetilde{d}$) were used, then the security company would-be able to follow all the surveillance data after accessing the keys ($\widetilde{e}$, $\widetilde{d}$) for the first time – even in the non – alarming situations. This clearly violates the privacy policy we have set to the system. Thus, a more sophisticated access control method is needed. We have identified the following two approaches to implement access to the recovery key $\widetilde{d}$.

3.2 The First Approach

In this approach, the recovery key $\widetilde{d}$ for recovering the encrypted surveillance data $\widetilde{X}$ is possessed by the security company. However, the key $\widetilde{d}$ must be protected by an electronic seal because otherwise the company could decrypt all the surveillance data and not only the data related to alarms. The company is allowed to break the seal whenever the analysis yields an alarm. After breaking the seal, the company uses the recovery key to obtain the actual surveillance data consisting of the moments some time before and after the alarm.

The above approach requires a countermeasure to detect unauthorized access to the surveillance data. One possible way is to utilize event logging. Each of the three parties related to the surveillance are maintaining their own independent event logs. The independent logs contain information that can be derived from the activities of the different parties (for example, the company logs access to the raw data together with a synopsis of the analysis results). These three independent logs can in principle be compared to detect unauthorized or illegitimate access to the backup data. Of course, the different logs can be forged and the comparison does not work in the desired way in case there are conspiracies between the parties but solving conspiracy issues is not in the scope of this paper.

In this approach, the efficiency of the implementation depends on the efficiency of the used garbling scheme as well as the efficiency of the used independent encryption scheme $\mathcal{E} = (\widetilde{\mathrm{KeyGen}}, \widetilde{\mathrm{En}}, \widetilde{\mathrm{De}})$. The efficiency of garbling schemes is discussed in more detail in Section 4.3. The efficiency of the encryption scheme $\mathcal{E}$ is due to the choice of the security company. For example, $\mathcal{E}$ may be AES-128.

3.3 The Second Approach

In this approach, the recovery key $\widetilde{d}$ is in client's possession. Since the security company does not possess the decryption key $\widetilde{d}$ of $\widetilde{X}$, the company cannot monitor the data unless it is handed the decryption key. The company should be able to get the decryption key only in alarming situations. A straight-forward way of implementing the access control into the recovery key $\widetilde{d}$ is to use timestamped key management (see [19] for further information). The security company can access the keys $\widetilde{d}$ related to the raw data having certain timestamps that correspond to the time of the alarm detection as well as the data from some moments before and after the alarm detection. The client can later check which keys have been sent to the security company and, if needed, check the corresponding raw data.

There is also a more innovative way of implementing the access control into the recovery key $\widetilde{d}$. Informally, our idea is to send the recovery key $\widetilde{d}$ to the security company via the cloud in such a way that the cloud does not learn the recovery key. Moreover, the security company will receive the key only in the case that the final analysis results yield an alarm. Next, we explain in more details, how this functionality can be implemented.

For simplicity, let us assume that the final surveillance analysis result is either alarm or no alarm, i.e. $y \in \{\text{alarm, no alarm}\}$. Now, we want that the security company gets $\widetilde{d}$ whenever y = alarm. This can be reached by attaching first the recovery key $\widetilde{d}$ to the surveillance data, i.e. $x_m = (x, \widetilde{d})$. This argument is then garbled and sent to the cloud, thus the cloud is not able to learn $\widetilde{d}$. The function f needs to be modified in order to be able to handle the new argument type. We define the modified function as follows

$$f_m(x_m) = \begin{cases} (y, \widetilde{d}) & \text{if } y = \text{alarm} \\ (y, \varepsilon) & \text{otherwise (where } \varepsilon \text{ is the empty string)} \end{cases}$$

The garbling scheme works in a similar manner as before. The cloud gets the garbled argument X_m from the client and the garbled function F_m from the security company. The cloud computes the garbled value Y_m and sends it to the security company. The security company ungarbles Y_m and gets y_m= (y, β). Here, $\beta \in \left\{\widetilde{d},\ \varepsilon\right\}$ depends on whether y = alarm or not.

The modifications in x and f now give the required functionalities. First of all, the security company gets the recovery key $\widetilde{d}$ only in the case y yields an alarm. Secondly, sending the key via the cloud is not insecure – the key remains garbled during the whole garbled evaluation in similar manner as the argument and the function.

The efficiency of implementation using this approach depends on the efficiency of the used garbling scheme and the efficiency of the used encryption scheme $\mathcal{E} = \left(\widetilde{\mathrm{KeyGen}}, \widetilde{\mathrm{En}}, \widetilde{\mathrm{De}}\right)$. However, the function f_m and the argument x_m are more complex than in the first approach, since the argument x_m contains the recovery key $\widetilde{d}$ and the function f_m needs to process $\widetilde{d}$ somehow. This means, that the second approach is not as efficient as the first approach. On the other hand, the second approach provides better control over the use of the recovery key $\widetilde{d}$.

4 Implementation of the Surveillance Service

In this section, we describe garbling schemes in more details. We start by defining the concept after which we discuss the different security measures for garbling schemes. We also discuss which of the security concepts are ideal for the use in the context of privacy preserving electronic surveillance.

4.1 Building Blocks

As mentioned earlier, our main building block to construct a privacy preserving and cloud-assisted surveillance system is garbling. Garbling enables surveillance data to be analyzed on cloud environment without compromising the privacy of the client or revealing business secrets in the form of the analytics tool.

The surveillance analytics tool may contain algorithms e.g. for anomaly detection [7, 9] (to detect the abnormal situations among the normal situations), and for machine learning. Recently, a method for running machine learning algorithms on encrypted data has been proposed [16].

One possible way of teaching the analytics tool is the following. Before the surveillance starts, the company and the client may have collected data from normal situations. These labeled situations together with the data from the surveillance system act as the training data for the semi-supervised learning (see [31] for more information) algorithm that now helps in doing the final analysis together with the other algorithms. We do not concentrate on the exact implementation of the analytics tool as our focus is on the tools enabling the privacy preserving surveillance.

4.2 Formal Definitions for Garbling

Formally, a garbling scheme is a 6-tuple of algorithms, (KeyGen, Ga, En, De, Ev, ev). The last component of the tuple is the evaluation algorithm ev: an algorithm that computes the value of function f on argument, i.e. $y = f(x)$. In our scenario, the function f is the surveillance analytics tool and the argument x is the surveillance data. To hide the analytics tool and the surveillance data, both f and x are garbled. To do this, first key generation algorithm KeyGen is called to generate three keys $(g,\ e,\ d)$. The garbling algorithm Ga computes the garbled function F = Ga(g, f) based on the function f and garbling key g. The encryption algorithm En computes the garbling X = En(e, x) based on argument x and encryption key e. The garbled evaluation function (the garbled analytics tool) computes the garbled value Y = Ev(F, X) (garbled analysis). Finally, the decryption algorithm De ungarbles Y and returns the final analysis result y = De(d, Y) by using the decryption key d issued by the KeyGen algorithm. Note that the final analysis result must be the same despite of the method used for evaluation: the garbled evaluation must yield the same final analysis result as the actual evaluation, i.e. ev(f, x) = y = De(d, Y) = De(d, Ev(F, X)). The garbled evaluation process is illustrated in Figure 3. For further details, consult e.g. [22].

In the example scenario presented in this paper, a garbling scheme G = (KeyGen, Ga, En, De, Ev, ev) is used as follows. The client in the scenario uses the algorithms KeyGen and En. The security company uses algorithm Ga. The cloud uses algorithm Ev. Depending on the case, either the cloud or the security company uses algorithm De. Figure 4 illustrates how the different algorithms are run by different parties in the example scenario.

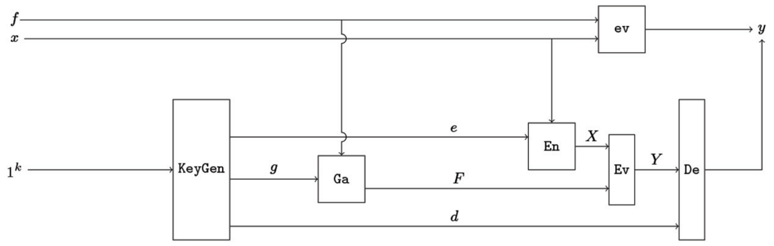

Figure 3 The components and the workings of a garbling scheme. The diagram is adapted from [22].

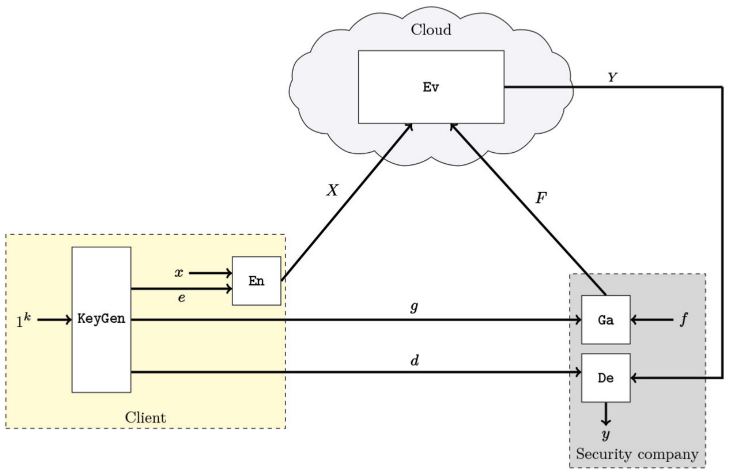

Figure 4 Garbling scheme in the surveillance scenario.

In the illustration, we assume that the channel between the client and the security company assures data integrity, authenticity and confidentiality. This is not assumed for the channel between the client/the security company and the cloud. We also present the situation in which the security company is the party ungarbling Y.

4.3 Security Considerations

In this section, we first introduce different security concepts for garbling schemes. Exact definitions for each concept can be found in literature [6, 5, 21–23]. Then, we analyze which of the security concepts meet the requirements for the privacy preserving electronic surveillance system proposed in the previous section.

Every security concept can be characterized by *security notion* and *level of reusability*. The security notion tells what kind of information about the function f and the argument x is allowed to be leaked. The notion *function and argument hiding* means that the garbling scheme is allowed to leak $f(x)$, but neither f nor x. The notion *function, argument and final value hiding* does not allow the garbling scheme to leak any of $f,\ x$ or $f(x)$. The notion *matchability-only* does not allow the garbling scheme to leak f nor x, but when evaluating f on two different arguments x_1 and x_2, the garbling scheme is allowed to leak whether $f(x_1)= y_1 = y_2 = f(x_2)$. Note that the names of the notions differ from the ones used in literature. We use non-standard names to distinguish what we mean by privacy in the example scenario and privacy related to garbling schemes. The notion *function and argument hiding* corresponds the notion *privacy* in [5, 23]. The notion *function, argument and final value hiding* corresponds the notion *obliviousness* in [5, 23].

The security notions described above deal with secrecy. The *authenticity* property can also be formalized for garbling schemes. Authenticity guarantees that an adversary is unable to create a garbled value Y from a garbled function F and its garbled argument X such that $Y \neq F(X)$ but which will be considered authentic. For a formal definition of authenticity for garbling schemes, consult [6]. The authenticity of garbling schemes is needed to fulfill requirement 2, demanding that the final analysis result must be authentic. Thus, the garbling scheme used in the example scenario must achieve authenticity in the sense explained in [6].

Another characteristic of a garbling scheme is *the level of reusability*. The level of reusability tells how many times the same garbled function can be securely used for different arguments. The first reusability level enables only one-time use of the same garbled function [5, 6, 21] whereas higher levels of reusability enable several or even arbitrary reuse of the garbled function [15, 23].

Let us first recall Confidentiality, Integrity, Entity authentication, Access control, Authorization, Non-repudiation and Availability requirements presented in Section 2. The Confidentiality requirement says that the unencrypted surveillance data must be kept secret from third parties, including the cloud. Moreover, the analytics tool must be hidden from third-party cloud. From the garbling point-of-view, this means that the garbling scheme should be at least function and argument hiding. The Confidentiality requirement also tells that hiding the final analysis result depends on the contract between the client, security company and the cloud.

We have identified three possible configurations of the surveillance system all of which set different security requirements to the garbling scheme in use. In all three cases, the surveillance data as well as the surveillance data analytics tools are kept secret from the cloud. The differences in the configurations are related to the final analysis results: are the analysis results kept totally, partially or not at all secret from the cloud. Let us next provide more details of these three different configurations.

Case 1: The cloud is allowed to learn nothing about the resulting analysis. This means that the surveillance data, the analytics tool and the final analysis result are all hidden from the cloud. From the garbling point-of-view, this is the same as hiding the function, the argument and the final value. This is desirable, because the third-party company may use the information about the analysis for its own purposes that might be unwanted by the client. Thus, the garbling scheme must leak none of f (the analytics tool), x (the surveillance data) or $y = \mathrm{ev}(f, x)$ (the analysis result) to the cloud. A garbling scheme that is function, argument and final value hiding meets these requirements.

Case 2: The cloud is allowed to learn indirect information about the analysis result but possibly not the actual content of the final analysis. From the garbling point-of-view, this is the same as hiding the function, the argument and the final value but leaking some information about the final value. One justification for this weaker privacy requirement is the following: the cloud service provider may anyway be able to find out the actions of the security company related to certain garbled analysis results. For example, the cloud has found that garbled analysis result Y yields a call to the police. When the same garbled analysis result Y is found later again on the cloud, the cloud service provider is able to predict the reaction of the security company – the security company will probably call to the police.

Thus, we could require that the cloud service provider cannot find out f, x or y but it is able to find out whether the certain Y yields similar actions as before. For the garbling scheme this means that the scheme should not leak $f,\ x$ or y but it may leak whether $f(x_1) = y_1 = y_2 = f(x_2)$ when computing $\mathrm{ev}(f_1, x_1)$ and $\mathrm{ev}(f_2, x_2)$. This is exactly what a garbling scheme achieving matchability-only security provides.

Case 3: The cloud service provider is allowed to learn the final analysis result. From the garbling point-of-view, this means that the garbling scheme is allowed to leak the final value y whereas it must hide the function and the argument. However, the Confidentiality requirement tells that the cloud is allowed to learn nothing about the ungarbled surveillance data, meaning that the final analysis cannot contain parts of surveillance data. To assure this, the final value could be something else than a review of the surveillance data. It may also be *the type of alarm*, like no alarm, low urgency, medium urgency, high urgency etc. or simply alarm/no alarm. Now it may under some circumstances be acceptable to let the cloud provider to know the type of the alarm. This means the garbling scheme should hide f and x but it may leak y. This is exactly what a function and argument hiding garbling scheme provides. However, it is questionable whether the cloud should generally learn that there is an alarming situation at the client. This violates the requirement that the third – party cloud should learn nothing about the surveillance, not even the fact that the security company is being alarmed.

The above reasoning suggests that the garbling scheme should either be function, argument and final value hiding or achieve matchability-only. From the practical point of view, matchability-only is preferable as it has been shown in [21–23] that it is at least as easy to achieve matchability-only as to be function, argument and final value hiding. Moreover, for practical reasons one should be able to use the same garbled analytics tool for several garbled surveillance data entries. For the garbling scheme this converts to reusability. Thus we suggest that the applied garbling scheme should be reusable as well as achieve matchability-only and authenticity. This guarantees that the surveillance is privacy-preserving since the third parties do not learn the ungarbled surveillance data or the ungarbled analysis result.

4.4 Efficiency Considerations

The above concepts do not restrict the computation method for evaluating function f on argument x – the function f may represent models such as a circuit, a Turing machine or a random-access machine. Methods for garbling various computational models have been constructed. For example, there exist garbling schemes for circuits [30], Turing machines [14] and random-access machines [13].

The choice of the computation method affects the efficiency of the garbling scheme. Choosing circuits over Turing machines has at least two unfortunate consequences. The first consequence is related to the running time of circuits. The running time of a circuit is constant, implying that evaluating a circuit with any input takes the worst-case running time. This is not the case for Turing machines. Another unfortunate consequence is related to the size of the garbled function F. Turing machines outperform circuits also in this aspect: the size of garbled circuit is as large as the running time of the algorithm where as the size of the garbled Turing machine depends only on the description of the algorithm and not on the input value x. [14]

On the other hand, using circuits as the computation method has benefits over Turing machines when considering the costs of constructing the garbling scheme. Garbled circuits are known to have efficient constructions [4] where as such are not known for garbled Turing machines. Garbled Turing machines typically use fully-homomorphic encryption [12] as a building block which causes inefficiency in the construction.

Next we provide some numbers on the efficiency of garbling. The values have been collected from [4], in which three different garbling schemes have been experimented by using JustGarble (the source code is open-source and is available in [1]) system on an x86-64 Intel Core i7-970 processor clocked at 3.201 GHz with a 12MB L3 cache. The three garbling schemes are based on a function and value hiding garbling scheme Garble1 presented in [6]. All three presented garbling schemes are based on *dual-key cipher*. The difference in the three schemes is the different optimization techniques used to reduce the evaluation time. For more details, consult [4].

On a circuit having 15.5 million gates, of which 9.11 million gates are XOR gates, the most efficient garbling scheme *GaX* uses approximately 0.49 seconds to garble the circuit and 0.23 seconds to evaluate the garbled circuit [4]. This shows that the time for evaluating and garbling using *GaX* is efficient even on quite large circuits, meaning that even complex algorithms represented as circuits can be efficiently garbled and evaluated with *GaX*.

Using this measurement data presented in [4], we can estimate the efficiency of our solution for the example scenario as follows. Let us assume that the garbling scheme *GaX* is used. Let us further assume that the analytics tool is presented as a circuit having approximately 15.5 million gates. If the client sends surveillance data at rate of 0.5 kilobits/second then an analysis result is received by the security company approximately once per second. Achieving the low sending rate of 0.5 kilobits per second requires some pre-processing of the surveillance data on client side, e.g concerning the captured video stream where raw data is accumulated in much higher data rate. As a summary, these figures seem to be acceptable from practical point-of-view.

There are some known issues related to the use of scheme *GaX*. The garbled argument *F* is constructed at the same time as the keys (e, d), meaning that one party needs to possess both the function and the argument. We can solve this problem in two ways. First one is to let a trusted authority to run the garbling algorithm Ga with the garbling key g obtained from the client and the function f obtained from the security company. Another solution would be that the client and the security company use a secure multiparty computation protocol for computing $\mathrm{Ga}(g, f)$ together in such a way that the security company does not learn g and the client does not learn f. Unfortunately, both solutions add to the complexity of the system and increase the time to garble.

Another problem in using any of the garbling schemes from [4] is that these garbling schemes are not reusable. This means that every time a new surveillance data entry is ready to be processed, both the surveillance data entry and the analytics tool need to be garbled. This implies that big amount of the total computation happens in the client side, and therefore the benefit of using cloud is questionable. In garbling schemes supporting reusable garbled circuits, the analytics tool represented as a circuit is garbled only once which would increase the overall efficiency of the garbling scheme. In our scenario, this would correspond to a situation where most of the computation load can be moved to the cloud. Unfortunately, no efficient constructions for reusable garbled circuits are known.

5 Discussion

Applications following the Internet of Things paradigm have increased rapidly, even to the extent that the security, trust and privacy related to the applications have not been able to keep up with the progress. Especially, privacy preservation seems to be one of the hottest topics related to the Internet of Things. One application that is regarded as privacy violating is electronic surveillance, at least in the private premises such as homes. On the other hand, there is a need to monitor private homes in order to track emergencies and protect the customers from various threats. We are confronting the challenging task of creating a privacy preserving electronic surveillance system.

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

The example scenario is not the only possible application for garbling. As another related example, a monitoring system can be installed in the homes of people using the services for *assisted living*. The party monitoring the data should not learn the habits of the person using the system beyond the situations in which the person needs help. In this scenario, the security company may provide the monitoring services to the company providing the services for assisted living. This makes the privacy preservation even more complex task.

A variant of our example scenario presented in this paper is that the security company does not use third-party cloud services and instead does all the analysis itself. The operation of the parties present in this variant scenario resembles the operation of the same parties in the example scenario. However, there is one fundamental difference: the party possessing the analytics algorithms is the same as the party evaluating the analytics algorithms. This means that the security company first garbles the analytics algorithm as before, but then evaluates the garbled analytics algorithms on the garbled data received from the client.

In this case, the garbled evaluation might directly give the final analysis result y as hiding the result from the security company itself is useless. Moreover, the garbling of the analytics tool is not essential since the same party (the security company) both possesses the analytics tools and is responsible for the evaluation. Hence, the variant scenario is more efficient than the original scenario. However, moving the computation load from the cloud to the security company requires that the computational resources on the security company side should increase.

To conclude, we have found a novel solution to provide privacy preservation in an electronic surveillance system utilizing the Internet of Things paradigm. The biggest advantage in our solution is that garbling provides flexibility in the system. The surveillance analytics tool can be almost anything, from comparisons to complex machine learning algorithms. Moreover, the function f can be changed without need to reconfigure the whole system, easing the system maintenance.

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

Future research may solve the problem of efficient reusable garbled circuits. Moreover, exploring further targets for innovative use of garbling in the context of IoT is important. An interesting target would be larger and more complex systems having more parties, for example a scenario where a person uses services for assisted living. In this scenario, we would have four parties – the client, the assisted living service provider, the security service provider (providing the devices for monitoring the client) and the third-party cloud service provider.

Acknowledgments

Authors would like to thank anonymous referees for valuable suggestions that have improved the paper quality significantly. This work was supported by the Academy of Finland project “Cloud Security Services” which is greatly appreciated.

References

[1] JustGarble. http://cseweb.ucsd.edu/groups/justgarble/. Accessed: 2014-10-13.

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

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

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

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

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

[7] V. Chandola, A. Banerjee, and V. Kumar. Anomaly Detection: A Survey. *ACM Comput. Surv.,* 41(3): 15: 1–15: 58, July 2009.

[8] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for Privacy Preserving Distributed Data Mining. *SIGKDD Explor. Newsl.,* 4(2): 28–34, Dec. 2002.

[9] T. Dunning and E. Friedman. *Practical Machine Learning: A New Look at Anomaly Detection.* O'Reilly Media, 2014.

[10] S. Evdokimov, B. Fabian, O. Günther, L. Ivantysynova, and H. Ziekow. RFID and the Internet of Things: Technology, Applications, and Security Challenges. *Foundations and Trends@in Technology, Information and Operations Management,* 4(2):105–185, 2011.

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

[12] C. Gentry. *A Fully Homomorphic Encryption Scheme.* PhD thesis, Stanford University, 2009. crypto. stanford. edu/craig.

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

[14] S. Goldwasser, Y. Kalai, R. Popa, V. Vaikuntanathan, and N. Zeldovich. How to Run Turing Machines on Encrypted Data. In *Proc. of* 33^{rd} *CRYPTO,* volume 8043 of LNCS, pages 536–553, 2013.

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

[16] T. Graepel, K. Lauter, and M. Naehrig. ML Confidential: Machine Learning on Encrypted Data. In *International Conference on Information Security and Cryptology – ICISC 2012, Lecture Notes in Computer Science, to appear*. Springer Verlag, December 2012.

[17] E. Haselsteiner and K. Breitfuß. Security in near field communication (NFC). In *Workshop on RFID security*, pages 12–14, 2006.

[18] T. Heer, O. Garcia-Morchon, R. Hummen, S. L. Keoh, S. S. Kumar, and K. Wehrle. Security Challenges in the IP-based Internet of Things. *Wirel. Pers. Commun.*, 61(3): 527–542, 2011.

[19] A. V. D. M. Kayem, S. G. Akl, and P. Martin. Timestamped Key Management. In *Adaptive Cryptographic Access Control*, volume 48 of *Advances in Information Security*, pages 61–74. Springer US, 2010.

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

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

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

[23] T. Meskanen, V. Niemi, and N. Nieminen. Hierarchy for Classes of Garbling Schemes. In *Proc. of Central European Conference on Cryptology (CECC'14)*, 2014.

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

[25] M. O. Rabin. How to Exchange Secrets with Oblivious Transfer. Technical report tr-81, Aiken Computation Lab, Harvard University, 1981.

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

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

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

[29] J. S. Winter. Surveillance in Ubiquitous Network Societies: Normative Conflicts Related to the Consumer In-store Supermarket Experience in the Context of the Internet of Things. *Ethics and Inf. Technol.*, 161: 27–41, 2014.

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

[31] X. Zhu. Semi-Supervised Learning Literature Survey. http://pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf, July 2008.

Biographies

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

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

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