University of Bonn
Faculty of Mathematics and Natural Sciences
Institute of Computer Science IV

Bachelor Thesis

A protocol for preserving the confidentiality of pailler-encrypted secrets while uncovering their homomorphic additions using secret sharing techniques

Matthias Ulbrich
2743974

March 11, 2017

First examiner:	Dipl.-Inf. Saffija Kasem-Madani
Second examiner:	Prof. Dr. Michael Meier

Statutory Declaration

I hereby declare that I have authored this thesis independently, that I have not used other than the declared sources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Date: .... Signature: ....

Note of thanks and dedication

I want to thank my supervisor Dipl.-Inf. Saffija Kasem-Madani for the professional support and Prof. Dr. Michael Meier for the possibility to write this bachelor thesis in his department.
Many thanks to my girlfriend for the emotional support, to my sister, parents and my great aunt for the financial support during the course of my study. I dedicate this work to you.

Abbreviations

During their first use these terms will be spelled out completely. Any further mentions will use the corresponding abbreviation.

HBC	Honest-but-curious
SMC	Secure multi-party computation
SS	Secret sharing
SSS	Shamir's secret sharing
TTP	Trusted third party

Contents

List of Figures **v**

1 Introduction **1**

2 Perequisites **2**

2.1 Secure multi-party computation 2
2.2 Security models 4
2.2.1 Honest-but-curious/passive adversary 4
2.2.2 Malicious adversary 4
2.2.3 Static and dynamic adversary 4
2.2.4 Adversary structure 5
2.2.5 Fair vs unfair 5
2.2.6 Ideal-real-world model 5
2.3 Secret sharing 5
2.3.1 Shamir's secret sharing 6
2.3.1.1 Sharing the secret 6
2.3.1.2 Reconstructing the secret from k shares 6
2.3.2 Homomorphic Secret Sharing 7
2.3.3 Verifiable Secret Sharing 7
2.4 The Pailler Cryptosystem 9
2.4.1 Encryption 9
2.4.2 Decryption 9
2.4.3 Homomorphic properties 9
2.4.4 Malleability 9

3 A protocol for SMC with SSS **9**

3.1 Related work 9
3.2 Design 9
3.2.1 Dealer 9
3.2.2 Analyzer 9
3.2.3 Shareholder 9
3.3 Implementation of the Protocol 9
3.3.1 Dealer 9
3.3.2 Analyzer 9
3.3.3 Shareholder 9
3.4 Evaluation of the Protocol 9
3.4.1 Input Privacy and Correctness 9
3.4.2 Ideal-real-world comparison 9
3.4.3 A differential attack 9
3.5 Final words 9

References **10**

List of Figures

1 Ein Testbild 8

1 Introduction

Tracking of has become an omnipresent phenomenon in today's web. Common websites make use of multiple trackers and these trackers communicate to third party services such as facebook [How09]. With constant tracking of users the accumulated data has to be stored and managed in a privacy-preserving manner. The german lawmaker calls to limit the use of personalized tracking data and encourages the use of methods of anonymization and pseudonymization in the Bundesdatenschutzgesetz [BDS]:

> **§3a Data reduction and data economy**
> The collection, processing and use of personal data and the selection and design of data processing systems must be geared to the goal of collecting, processing or using as little personal data as possible. In particular, personal data shall be anonymised or pseudonymized, insofar as this is possible according to the intended use and does not require disproportionate expenditure in relation to the intended protection purpose. (Translation by Google Translate[1])

Many companies have build their business around tracking users around the web and evaluating their activities and preferences. Most well known are Google and Facebook which caused most of the third party requests in a 1-million-site census [EN16]. Both companies which have made news when changing their terms of use for their services to be more privacy evasive or, in the case or Facebook when they aquire another company such as Instagram or WhatsApp. People fear for their privacy which seems at times out of control while at the same time law makers try to give indiviuals back control of their data [Neu].

Laws makers call upon companies to anonymize data, which could very well also be in a companies interest for fear of public haze following hacking attempts of their publicized databases. Moreover, the accumulated data represents a value in itself upon which extended data-mining is performed to learn about users. Thus, it is evident that it is in a companies own interest to keep its data confidential even when sharing it with other parties.

In this thesis a design is proposed which allows a Data Holder - the party which originally accomulated tracking data - to share its information in such a way that analysis can be peformed without revealing the indivitual pieces of information in the database. Specificly, the database consists of positive integers (e.g. net income). The client (which will be referred to as the Analyzer from here on) is able to determine the total income of several individuals without gaining knowledge about the specific income of an individual in the process.

The design employs **s**hamir's **s**ecret **s**haring, an application of **s**ecure **m**ulti-party **c**omputation to achieve this. While determing the total of a subset of individuals net income does not directly reveal an individuals income, it is shown that by design allowing the computation of a very small subset can enable an attacker to end up gaining insight about individual data.

[1] `translate.google.de`, for the official translation: `www.gesetze-im-internet.de/englisch_bdsg`

2 Perequisites

2.1 Secure multi-party computation

In an everyday context there are scenarios where competing parties may need to collude in computing. Take two millionares for example which want to know which one is richer without revealing their wealth to each other. This is known as the Millionaire's Problem and Andrew C. Yao has introduced a privacy-preserving SMC protocol for this problem in [Yao82]. Another example explained by Anna Lysyanskaya is how three parties can compute their net wight without having to admit their own weight to each other [Lys08].

The most basic example is the use of a **t**rusted **t**hird **p**arty. All parties send their inputs to that party which computes and then returns the outcome to all parties. However, in the case of rivals privacy of the inputs is a main concern. A TTP is to be avoided and removed entirely from the protocol. This is where SMC comes in. SMC, also referred to as secure function evaluation is a term used in cryptography for designs which involve the participation of several parties to compute in conjunction over a distributed network - each party using their input - while keeping said input private towards the other parties. Thus, SMC enables parties with conflicting interest to cooperate with each other.

Any design of a protocol will need to satisfy different constraints, one the one hand security constraints to make it secure against a to be defined attacker model and privacy constraints on the other hand, i.e. privacy guarantees which we want to supply with the protocol. Depending on the model of the attacker the design will differ to account for powers of the attacker. Thus any design needs to describe what security model it has taken into consideration for an outsider to know what it means when one is speaking of a secure design. We will discuss the types of various security models in 2.2.

Formally, an instance of a SMC problem can be defined as follows [CDN15, p.6]:

Definition 1. ***(Secure multi-party computation)***
Given n participating parties $P_1, P_2, ..., P_n$ which each have a secret input x_i, these parties want to compute a function

$$f(x_1, x_2, ..., x_n) = y$$

jointly while satisfing the two conditions:

1. *Input privacy: Party P_i does not learn anything about x_j for all $i \neq j$ during the execution of the protocol, i.e. there is no leakage of private data.*

2. *Correctness: The function f is correctly evaluated.*

When computing f with regards to these two conditions we say that f is computed securely. Hence the name alternative name secure function evaluation.

Now an example of SMC: Computing the net weight of three individuals as mentioned by Lysyanskaya in [Lys08]. Alice has the input $x_1 = 45$, Bob the input $x_2 = 60$, and Carol the input $x_2 = 53$. These numbers refer to their weight in kilogram. How do they collude to determine their net weight? We want to evaluate the function $f(x_1, x_2, x_3) = \sum_i x_i$. In the proposed protocol each party firstly creates three random shares such that their sum equals their own weight mod 1000. These shares are then distributed to the other parties. From then on Alice, Bob and Carol need to compute in conjunction to evalute the function f.

1. Each party creates random shares depending on their input
   Alice: $(140, 620, 285)$
   Bob: $(500, 440, 120)$
   Carol: $(333, 621, 99)$

2. Each party distributes two of their shares to the other parties.

3. Each party adds these shares:
   Alice: $140 + 440 + 621 =_{1000} 201$
   Bob: $500 + 620 + 99 =_{1000} 219$
   Carol: $333 + 285 + 120 =_{1000} 738$

4. Each party broadcasts their result.

5. The sum mod 1000 of these values is the net weight of Alice, Bob and Carol.
   Everyone computes: $201 + 219 + 738 =_{1000} 158$ $(= 45 + 60 + 53)$!

As we can see, this SMC protocoll enables Alice, Bob and Carol to compute their net weight correctly without giving up privacy. So both conditions mentioned in 2.1 have been met. This functionality does come at a cost however, which is that the obfuscation by the computation is usually complex and requires more communication between the parties as can be seen in the example above. Computing the total weight of the tree parties requires five steps in total of which two steps require each party to communicate with all other parties. Thus, during the evaluation of the proposed design sheme of this thesis we will also take a look at common performace characterisitcs.

Is it also important to notice that in the example Alice, Bob and Coral were following the protocol and publishing the correct interim results in step four. Here it would be possible for an adversary to publish an incorrect interim result. Imagine for Coral to publish the value 748 instead of 738. Then Coral would still be able to perform the final calculation in private. But Alice and Bob are left in the dark about their true net weight. Both will assume that 168 is their true net weight. However, while they have published their shares in step 2, they have not published their personal weights. We introduce the different types of adversaries in 2.2.

While the example is simple, the complexity of the communication may suggest that MPC is not entirely possible for any desired function. It has been shown that *any* function can be securely evaluated. However, we need to assume that the majority of players is honest [GMW87]. To better describe how secure a SMC design is we will to introduce security models.

2.2 Security models

It is necessary to consider a security model which will underlie the design because the security model determines goals which are to be realized. A model is then called secure, if those security goals are met.

2.2.1 Honest-but-curious/passive adversary

A honest-but-curious adversary, also referred to as a passive adversary follows the SMC protocol but is interested in breaking the privacy of other parties [CD09]. An HBC adversay will try to extract as much information as possible from the given data, but cannot collude with other parties as this would deviate from the protocol. In that sense this can be considered a weak security model, if the parties in real world have an active interest in cooperating outside the protocol specifications to extract more information.

In Samim's thesis [Fai] an HBC adversary can leak the private data of a Pailler ciphertext by adding a zero which merely deforms the ciphertext. The Decrypter is not able to expose such an attack:

$$Pailler_{k_p}(m) * Pailler_{k_p}(0) = Pailler_{k_0}(m+0) = Pailler'_{k_p}(m)$$

2.2.2 Malicious adversary

Malicious adversaries don't follow the protocol specification and thus can collude with each other in order to subvert the privacy of another party.[CD09] The SMC design will have to take into consideration any subset of parties that may act malicious (see 2.2.4). Since a malicious adversary can deviate from the protocol he can send corrupt data or no data in particular to hinder other parties from computing the desired function as shown in the computation example earlier.

If n denotes the number of parties participating in the protocol, it is possible to securely evalute any function for up to $t < n/2$ HBC adversaries (i.e. honest majority). In the case of malicious adversaries we can also tolerate up $t < n/2$ malicious adversaries, but the security is based on a computational assumption. [Sma03][GMW87, p.220].

2.2.3 Static and dynamic adversary

Both the HBC and Malicious adversaries may be static, meaning that from the beginning of the execution of the protocol up until the end no new parties are corrupted. However, to model a more agressive scenario it should be possible for an attacker that controls a participating party to infiltrate others during the execution. The dynamic adversary may, at any time, corrupt new players. This model is also referred to as adaptive security. [CDN15, p.79].

The static malicious adversary is a fit model in cases where a party has been corrupted throught the whole process while a dynamic adversary is the most open model which takes into account any real word possibility. [Sma03]

2.2.4 Adversary structure

When considering a dynamic adversary it does not make sense to consider that *any* amount of players can be corrupted (extreme case: all players are corrupted). There needs to be an honest majority to be able to design a secure SMC scheme as mentioned earlier. To limit the amount of parties that can be corrupted the term adversary structure is introduced [CD09]:

The adversary structure $\mathcal{A}$ is a family of subsets of the players limited in size:

1. If $(A \in \mathcal{A}) \wedge (B \subset A) \implies B \in \mathcal{A}$ (monotone)
2. $\forall A \in \mathcal{A} : |A| < t$ (threshold)

2.2.5 Fair vs unfair

With asynchronous communication, one party may go last in order to subvert the protocol. This could be the case with Carol in step four of the aforementioned protocol example. Now Carol can subvert the protocol by not sending her interim result.

If a protocol ensures that *all* players can still determine the correct evaluation or *none* at all, we is called fair. [Sma03, p.386]

2.2.6 Ideal-real-world model

SMC is hard to formalize which is a problem, because the depicted problem is very general. Thus SMC was introduced in with two requirements in mind: Privacy of inputs and a correct result. However, this lax definition is not very formal and does not immediately lead to a definition of when a function has been securely evaluated. To measure security the ideal-real-word model is used [CD09, Zie15]:

A protocol is called secure if the adversary $\mathcal{A}$ does not *learn more* in the real world than it would in the ideal world.

Given a function f we imagine an ideal, incorruptible implementation of f in the functional $\mathcal{F}_{SFE}^{f}$ which leaks no private information. Then we achieve security by establishing a secure channel between the parties P_i and $\mathcal{F}_{SFE}^{f}$. However, in the real word f is implemented by some functional π_{SFE}^{f}. This functional needs to provide a protocol without the help of a TTP and with adversaries in mind.

If we denote what the adversary $\mathcal{A}$ learns in the real word with $\text{IDEAL}_{\mathcal{A}}(\mathcal{F}_{SFE}^{f})$ and analogous $\text{REAL}_{\mathcal{A}}(\pi_{SFE}^{f})$ for what the adversary learns in the real world, we call the protocol $\mathcal{F}_{SFE}^{f}$ perfectly secure if

$$\text{IDEAL}_{\mathcal{A}}(\mathcal{F}_{SFE}^{f}) = \text{REAL}_{\mathcal{A}}(\pi_{SFE}^{f})$$

2.3 Secret sharing

In the following section we will introduce the secret sharing paradigm using a method first described by Shamir [Sha79]. Generall there secret sharing schemes fall into two categories: Yao's garbled circuit approach enables two-party SMC for his millionaire problem [Yao82]. **S**hamir's **s**ecret **s**haring enables SMC in a distributed network of two or more parties.

2.3.1 Shamir's secret sharing

Shamir's secret sharing is based on polynomials over a finite field $\mathbb{F}$, say $\mathbb{F} = \mathbb{Z}_p$ for some prime p. Shamir's secrets sharing devides a numerical secret $s \in \mathbb{Z}_p$ using a polynomial $f(x)$ into n *unique* pieces of information which are called shares. When diving up a polynomial, these shares are evaluated function values $s_i := f(i)$, i.e. $(i, s_i))$ are points on the graph of the polynomial function. These s_i are divided among a set of n shareholders identified by the index i.

Any subset of k shareholders out of n can reconstruct the polynomial as a polynomial of degree $n - 1$ is determined by n unique points. This is called a $(k, n) - threshold$ sharing sheme.

Formally we define in accordance with Shamir[Sha79]:

Definition 1. *(k,n) threshold sheme*
A (k,n) threshold sheme is a division of a piece of information D (e.g. a secret key) into n pieces $D_1, D_2, \ldots, D_n$ such that:

1. *knowing any subset of k pieces D_i makes D easily computable*
2. *knowing any subset of less than k pieces D_i leaves D undetermind in such a way that all possible values are equally likely*

2.3.1.1 Sharing the secret

Now imagine we have a secret $s \in \mathbb{Z}_p$. This numerical value is information which some data holder has collected. The data holder wants to sell this information for analysis while hiding the secret value. The data holder decides to employ a (k, n) threshold scheme. In order to do so the data holder generates random coefficients $f_i \in \mathbb{F}_q$ with $i = 1, \ldots, k$ and $k < n$ to form a random polynomial $f(x)$. Then k shares determine the polynomial. Note that the secret s is selected to be the value of the polynomial when evaluated at zero $s = f(0)$:

$$f(x) = s + f_1 x + f_2 x^2 + \ldots + f_{k-1} x^{k-1}$$

The data holder has the role of a honest dealer when generating the shares from the polynomial. Set $X \subset \mathbb{F}_q\{0\}$, then the dataholder generates a random share by drawing a point from the polynomial for values $i \in X$. The zero value is excluded as the evaluation of the polynomial at zero is reserved to store the secret s. Finally, we end up with the vector:

$$\text{Shares of s: } (s_1, \ldots, s_n)$$

Threshold sharing is enabled by generating equal or more shares than the degree of the polynomial, such that the points (i, s_i) (over-)determine the polynomial.

2.3.1.2 Reconstructing the secret from k shares

Now that the data holder has given out the shares, k shareholders can collude in reconstructing the polynomial and thus determine the secret. Given k shares

$(s_1, \ldots, s_k)$ we want to reconstruct the secret s. This is accomplished by determining the Lagrange interpolation of $f(x)$:

$$f(x) \coloneqq \sum_{i=0}^{k} \delta_{x_i}(x) f(x_i)$$

Where $\delta_{x_i}(x)$ is the Langrage basis polynomial defined as:

$$\delta_{x_i}(x) \coloneqq \prod_{i \neq j} (x - x_j)(x_i - x_j)^{-1}$$

The left factor guarantees that the langrange polynomial evaluates to 1
[Sma03]

2.3.2 Homomorphic Secret Sharing

Homomorphic Secret Sharing refers to a secret sharing sheme which employs an operation $\otimes$ on the shares [Sch99]. The combination of secret shares then enables to reconstruct a polynomial for the under $\otimes$ *combined* secrets. Formally, we define the homophoric property of a sharing sheme as introduced by [Ben86].

Definition 1. $(\otimes, \oplus)$*-homomorphic*
Given the operators $\otimes$ on the secret domain T and $\oplus$ on share domain S, a (k,n) threshold sheme is called $(\otimes, \oplus)$-homomorphic if for all functions $F_I : T^k \to S$ which compute the secret value of the secret sharing mechanism the follow implication is valid:

$$\begin{aligned} D &= F_I(D_{i_1}, D_{i_2}, \ldots, D_{i_k}) \quad \wedge \\ D' &= F_I(D'_{i_1}, D'_{i_2}, \ldots, D'_{i_k}) \\ \Rightarrow D \otimes D' &= F_I(D_{i_1} \oplus D'_{i_1}, Di_2 \oplus D'_{i_2}, \ldots, D_{i_k} \oplus D'_{i_k}) \end{aligned}$$

Shamir's Secret Sharing as described in 2.3.1 is $(+, +)$-homomorphic. This property will be used in our design proposed in 3.

2.3.3 Verifiable Secret Sharing

A key tool for secure MPC, interesting in its own right, is verifiable secret sharing (VSS): a dealer distributes a secret value s among the players, where the dealer and/or some of the players may be cheating. It is guaranteed that if the dealer is honest, then the cheaters obtain no information about s, and all honest players are later able to reconstruct s, even against the actions of cheating players. Even if the dealer cheats, a unique such value s will be determined already at distribution time, and again this value is reconstructable even against the actions of the cheaters.[CD09]

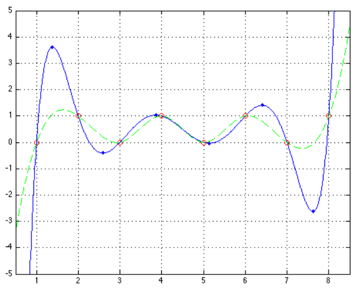

Figure 1: Ein Testbild

2.4 The Pailler Cryptosystem

2.4.1 Encryption

2.4.2 Decryption

2.4.3 Homomorphic properties

2.4.4 Malleability

3 A protocol for SMC with SSS

3.1 Related work

3.2 Design

3.2.1 Dealer

3.2.2 Analyzer

3.2.3 Shareholder

3.3 Implementation of the Protocol

3.3.1 Dealer

3.3.2 Analyzer

3.3.3 Shareholder

3.4 Evaluation of the Protocol

3.4.1 Input Privacy and Correctness

3.4.2 Ideal-real-world comparison

3.4.3 A differential attack

3.5 Final words

In this chapter a design is proposed which allow a user to ga... Assumtion made: Secure channels, and identification of other users: TLS with signatures. The shares shares distributed by the dataholder are consistent. VSS

Sonstige: Stichwörter provably secure semi-honest privacy-preserving perfect security

References

[BDS] Bdsg - einzelnorm. https://www.gesetze-im-internet.de/bdsg_1990/__3a.html. (Accessed on 02/08/2017).

[Ben86] Josh Cohen Benaloh. Secret sharing homomorphisms: Keeping shares of a secret secret. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 251–260. Springer, 1986.

[CD09] Ronald Cramer and Ivan Damgård. Multiparty computation, an introduction. 2009.

[CDN15] Ronald Cramer, Ivan Bjerre Damgrd, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, New York, NY, USA, 1st edition, 2015.

[EN16] Steven Englehardt and Arvind Narayanan. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1388–1401. ACM, 2016.

[Fai] Mohammad Harun Samim Faiz. Untersuchung und Implementierung ausgewählter homomorpher Hashing-Verfahren für die Integritätsprüfung homomorpher Chiffrate.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 218–229. ACM, 1987.

[How09] How online tracking companies know most of what you do online (and what social networks are doing to help them) — electronic frontier foundation. https://www.eff.org/deeplinks/2009/09/online-trackers-and-social-networks, 09 2009. (Accessed on 02/08/2017).

[Lys08] Anna Lysyanskaya. How to keep secrets safe. *Scientific American*, 299(3):88–95, 2008.

[Neu] Neue eu-datenschutzregeln: Facebook erst ab 16 jahren — heise online. https://www.heise.de/newsticker/meldung/Neue-EU-Datenschutzregeln-Facebook-erst-ab-16-Jahren-3044585.html. (Accessed on 02/08/2017).

[Sch99] Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In *Annual International Cryptology Conference*, pages 148–164. Springer, 1999.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[Sma03] Nigel Paul Smart. *Cryptography: An Introduction*, volume 5. McGraw-Hill New York, 2003.

[Yao82] Andrew C Yao. Protocols for secure computations. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 160–164. IEEE, 1982.

[Zie15] Christian Zielinski. Introduction to secure multi-party computations. `https://github.com/czielinski/secmultipartycomp/blob/master/slides/sec_multi_party_comp.pdf`, 2015. (Accessed on 02/08/2017).