ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2015-ENST-0062

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Computer Science and Networking

présentée et soutenue publiquement par

Iraklis LEONTIADIS

2 Octobre 2015

Protection de la vie privée dans la collecte et l'analyse des données

Directeur de thèse : **Refik MOLVA**
Co-encadrement de la thèse : **Kaoutar ELKHIYAOUI**

Jury

M. Sébastien GAMBS	Rapporteurs
M. Loukas LAZOS	
M. Emiliano DE CRISTOFARO	Examinateurs
Mme. Maryline LAURENT	
M. Refik MOLVA	Directeurs de thèse
Mme. Kaoutar ELKHIYAOUI	

TELECOM ParisTech
école de l'Institut Télécom - membre de ParisTech

T H È S E

2015-ENST-0062

Doctorat ParisTech

Thesis Dissertation

for the degree of Doctor of Philosophy by

TELECOM ParisTech

Computer Science and Networking

presented by

Iraklis LEONTIADIS

2 October 2015

Privacy Preserving Data Collection and Analysis

Thesis advisors : **Refik MOLVA**, **Kaoutar ELKHIYAOUI**

Jury	
M. Sébastien GAMBS	Reviewers
M. Loukas LAZOS	
M. Emiliano DE CRISTOFARO	Examiners
Mme. Maryline LAURENT	
M. Refik MOLVA	Thesis advisors
Mme. Kaoutar ELKHIYAOUI	

TELECOM ParisTech
école de l'Institut Télécom - membre de ParisTech

Acknowledgements

First and foremost I would like to express my gratitude to my supervisor, Prof. Refik Molva, who opened the door to my research path. He taught me how to think *out-of-the-box* with discipline, creativity and meaningful criticism. The unprecedented trust he showed on my proposals and ideas, rendered me tackling daunting research problems with a stimulating attitude. I am also indebted to Dr. Kaoutar Elkhyaoui who shared with me fruitful discussions and various suggestions, as a second supervisor. Dr. Melek Önen showed an extreme amount of patience on my stubbornness in various aspects during my Ph.D. studies and i am also beholden for her comments. I would also like to thank the jury members for accepting being part of the committee of this dissertation and all great people from Eurécom, whom i met during these four years.

Special thanks go also to Giorgos Arvanitakis for sharing moments, enjoying research successes any time during the day, but most importantly for analyzing the “failures”. During our meetings we have been brainstorming towards the maximization of quality, price and healthy factors of nutrition along with short fitness workout proposals. The result of this brainstorming is the semi-marathon than i run with Vaggelis. My dissertation would not be interesting without the nights in Tzarewitch with Arvanitakis, Tsimiklis, Vaggelis and Nikos, which motivated me to make a pause from my work. I am also grateful to Isidora for her absurd tolerance to the fluctuation of my personality during my dissertation. I am also appreciative to Mairh and Leon who sacrificed a lot of their personal time and financial budget (unfortunately) for my education.

Abstract

The progress in hardware and communication technology enables data analyzers to compute with accuracy aggregate information, owing to the wide availability of data. The personal sensitive information that is binded with individual data, renders users reluctant in publishing it. The seemingly paradoxical requirement of preserving individual confidentiality while at the same time granting partial access to an aggregate value over the data, has been addressed with Privacy Preserving Data Collection and Analysis protocols. However, in order to achieve individual privacy and efficiency, current cryptographic solutions assume *honest-but-curious* third parties or fully trusted key-dealers to distribute keys, thus restricting the security model and hindering its deployment in a dynamic environment.

In this dissertation, we design and analyze new Privacy Preserving Data Collection and Analysis protocols to strengthen the existing security model and to propose new features. We first propose a solution to the problem of privacy preserving clustering by exploiting the inherent properties of a specific similarity detection algorithm. Then, we design a solution that allows an energy supplier to learn more sophisticated statistics, such as the time interval of maximum energy consumption, without violating individuals' privacy. Afterwards, we address the problem of data aggregation in a dynamic environment by relaxing existing trust assumptions. Finally, we strengthen the security requirements of existing protocols by considering the case of a malicious Aggregator who will try to provide bogus or biased results. Our protocols are analyzed under the provably secure framework and their practicality is shown with prototype implementations.

Résumé

Les progrès matériels et technologiques en terme de communication permettent aux analystes de données de calculer avec précision des données agrégées en raison du grand volume de données disponibles. Les informations personnelles sensibles, liées aux données individuelles, rendent les utilisateurs réticents à publier ces données. La contradiction entre le besoin de respect de la vie privée individuelle d'une part, et celui de donner un accès partiel à une description synthétique des données d'autre part à été résolue avec des protocoles de collecte et d'analyse des données respectueux de la vie privée. Cependant afin de garantir à la fois efficacité et respect de la vie privée, les solutions cryptographiques actuelles supposent l'existence d'un tiers parti honnête mais curieux ou d'une entité distribuant les clés en qui l'on puisse avoir une confiance totale, limitant ainsi le modèle de sécurité de ces solutions et entravant leur déploiement dans un environnement dynamique.

Dans cette thèse, nous concevons et analysons plusieurs nouveaux protocoles de collecte et d?analyse des données respectueux de la vie privée en vue de renforcer le modèle de sécurité existant et de proposer de nouvelles fonctionnalités. Nous proposons une première solution au problème de la catégorisation respectueuse de la vie privée en exploitant les propriétés inhérentes d'un algorithme spécifique de détection de similarité. Puis, nous concevons une solution qui permet à un fournisseur d'énergie d'apprendre des statistiques plus sophistiquées, comme l'intervalle de temps de la consommation d'énergie maximale sans violer la vie privée individuelle. Ensuite, nous abordons le problème de l'agrégation de données dans un environnement dynamique en relaxant les hypothèses de confiance existantes. Enfin, nous renforçons les exigences de sécurité des protocoles existants avec un agrégateur malveillants qui tentera de fournir des résultats faux. Nous montrons la faisabilité de nos solutions avec des implémentations de prototype. La sécurité de chacun de protocoles est analysée dans le cadre de la sécurité prouvée.

Contents

List of Figures

List of Tables

List of Publications

The author of this dissertation has contributed with the following scientific publications:

1. Iraklis Leontiadis, Kaoutar Elkhiyaoui, Melek Önen, Refik Molva. *PUDA–Privacy and Unforgeability for Data Aggregation.* In Proceedings of the 14th International Conference on Cryptology and Network Security **CANS 2015**, Marrakesh, Morocco, December 2015.

2. Iraklis Leontiadis, Kaoutar Elkhiyaoui, Refik Molva. *Private and Dynamic Time-Series Data Aggregation with Trust Relaxation.* In Proceedings of the 13th International Conference on Cryptology and Network Security **CANS 2014**, Heraklion, Crete, Greece, October 2014.

3. Iraklis Leontiadis, Refik Molva, Melek Önen. *Privacy Preserving Statistics in the Smart Grid.* In Proceedings of the 13th International Workshop on Big Analytics for Security, in conjunction with **ICDCS** Madrid, Spain, June 2014

4. Iraklis Leontiadis, Refik Molva, Melek Önen. *A P2P Based Usage Control Enforcement Scheme Resilient to Re-injection Attacks.* In Proceedings of the Fifteenth International Symposium on a World of Wireless, Mobile and Multimedia Networks (**WoWMoM**), Sydney, Australia, June 2014

5. Iraklis Leontiadis, Melek Önen, Refik Molva, M.J. Chorley, G.B. Colombo. *Privacy preserving similarity detection for data analysis.* In Proceedings of the Collective Social

Awareness and Relevance Workshop 2013, co-located with the Third International Conference on Cloud and Green Computing. Karlsruhe, Germany, September 2013

6. Iraklis Leontiadis, Constantinos Delakouridis, Leonidas Kazatzopoulos, Giannis F. Marias. *ANOSIP: Anonymizing the SIP protocol.* In Proceedings of the Measurement, Privacy and Mobility Workshop, co-located with **EuroSys**. Bern, Switzerland, April 2012

Chapter 1

Introduction

Contents

1.1 Introduction

The motivation of this dissertation stems from our engagement with a project on Usage Control [3], which started as a side problem and progressively took us to the core topic of this dissertation. Namely, protocols for Usage Control aim to control how data is used during its entire lifetime, since regular access control systems cannot assure privacy for usage of data. An adversary in an access control scheme may copy and store deleted data, duplicate it, or malevolently use it in an unauthorized way. An essential component of a Usage Control enforcement scheme turns out to be a similarity detection function, which is employed to detect malicious data usage by untrusted parties. Our investigation on a specific category of data analysis operations such as similarity detection for security purposes, spurred our interest in a broader category of protocols in which functions other than similarity are evaluated by untrusted parties for different purposes.

Similarly, as with the Usage Control problem, there is a conflicting requirement between security and utility, in this type of protocols. An untrusted third party aims to learn some useful statistical information over a census of data, that represents a population of users. Untrusted parties collect data from individuals during the collection phase. After collecting all data, Aggregators, which are not authorized to have access to individual data, will try to analyze it to derive an aggregate value. Individual data contains personal sensitive information and users providing it seek to protect their privacy. Through the analysis of the data collected from users, useful statistical information can be computed in cleartext that will help Aggregators in decision making. As such, the problem becomes challenging when individual inputs to the function are obfuscated, so as to assure confidentiality. Different solutions address the problem hereafter called Privacy Preserving Data Collection and Analysis (**PPDCA**), using two classes of techniques. The first class relies on adding noise to data samples to assure privacy. The added noise allows the Aggregator to compute a statistical function over data with some error. On another variant of solutions that is based on cryptography, through non-conventional encryption and key-management techniques, the untrusted Aggregator learns the result of a statistical function without any noise. In spite of the advances made by such cryptographic solutions towards privacy and efficiency, the underlying security model does not involve a fully malicious Aggregator, or assumes the existence of a fully trusted key dealer who distributes keys to each party of the protocol. This dissertation focuses on cryptographic techniques for **PPDCA** protocols with a stronger threat model, while relaxing the existing trust assumptions, in order to better suit real world deployment. Before presenting the challenges and the goals that we tackle, we present use-case scenarios that motivate us.

1.2 Scenarios

In this section we provide some evidence through real world scenarios for the broad range of applications of Privacy Preserving Data Collection and Analysis protocols.

The confluence of powerful servers, ubiquitous computing devices and smart computing, allows the collection of massive amount of data from end users. The availability of large volume of information on the other hand, allows for aggregate operations as a powerful tool to infer statistical information about the underlying population, that improves the social welfare: consider a healthcare scenario whereby patients in a hospital receive personal information about their health status in an electronic form. This information constitutes their health history and it is considered personal information. Medical scientists on the other hand, seek to operate on data in order to derive statistical information such as sophisticated predictive models for predisposition to diseases (cancer, heart attack, genetic anomalies) or for offsprings' likelihood to diseases through genomic data. The medical data that are produced by a population of patients are collected by the medical scientists who behave as Aggregators. A possible misuse of patients' private data may have negative results in patients' life: the elicitation of medical data to an insurance company will negatively dominate the decision of the latter as to whether or not a patient will become a potential client. The core problem of the healthcare scenario between the medical data producers (users) and the medical scientists, who may act maliciously, is to assure the confidentiality of individual data, while allowing medical scientists to perform some operations on them.

In another context, thanks to the plummeting cost of hardware devices, smart meters are widely deployed in homes in order to report energy consumption in a smart grid environment. As the energy consumption monitored by smart meters may reveal sensitive information about the home, such as the number of people, the appliances and personal activities, users are not eager to disclose their energy consumption patterns. On the other end of the smart grid system, energy suppliers viewed as data consumers, collect and analyze energy consumption samples from smart meters in order to achieve various types of optimization. From the analysis of these samples, they are able to precisely forecast the electricity demand in order to allocate energy in advance according to the needs of an entire population. A typical privacy versus utility trade-off thus arises between the two ends of the smart grid system. The challenge in the smart metering scenario is to preserve individual data privacy while allowing untrusted parties access to some aggregate information over the meterings.

1.3 Privacy Preserving Data Collection and Analysis

In the aforementioned scenarios an untrusted party collects data from multiple users. Users want to protect their data confidentiality and they are reluctant to reveal their personal information. On the other hand, the untrusted party seeks to derive in cleartext a function over the entire data, without learning individual inputs. During a collection phase, an Aggregator collects obfuscated data. Later, during the analysis phase, the Aggregator performs some operations on data that allow it to reveal in cleartext some useful statistical information over the collected information. The conflicting requirement of preserving individual privacy on data and allowing access on aggregate information, renders the design of **PPDCA** protocols challenging. Let us now see some possible solutions to the problem. Homomorphic encryption allows operations on encrypted data but does not solve the problem of deriving a cleartext aggregate value. In a standard setting based on homomorphic encryption, the untrusted Aggregator would need the secret decryption key in order to decrypt the encrypted aggregate result, which would compromise individual privacy of users. Following a different direction, the problem could be mitigated with Multi-Party Computation (MPC) protocols. However, MPC implies a large communication overhead because users must exchange many secret messages for the computation of a function over the data. The functional encryption paradigm [31] can be used to design **PPDCA** protocols, but in case of multiple inputs–modeling multiple users owing personal data–the proposed functional encryption schemes would be prohibitively complex and costly [76].

We turn to more customized approaches that specifically deal with the **PPDCA** problem. Noise-based solutions consist of adding some noise to each data value before sending it to the Aggregator. The noise prevents the Aggregator from compromising individual privacy, but it is appropriately designed such that some statistics over all data inputs can be inferred. The second approach uses cryptographic protocols with a restricted set of operations an Aggregator can perform over the entire data. The customized **PPDCA** protocols proposed in the literature can be classified as follows:

- **Noise-based protocols**

  - **Ad-Hoc** techniques [5, 7, 19, 20, 95, 108, 120, 139]. In the ad-hoc approach, data is obfuscated by users with noise, such that a function over the cleartext data can be estimated by an untrusted party, while the adversary is confused by the noise data samples, thus achieving user privacy. Privacy is expressed with different means such as the distance between the noisy data distribution and the cleartext data distribution, or it is modeled in terms of the entropy of the noisy data distribution.

  Ad-hoc techniques lack a comprehensive notion of privacy. Furthermore, a comparison of ad-doc techniques in terms of the privacy they provide does not seem feasible, since each one uses a different privacy model. Solutions that adhere to the differential privacy framework address this lack of a formal privacy definition.

  - **Differential privacy** [4,15,22,42–44,55,56,59,60,62,64,71,116,118,124,132]. Within the differential privacy approach, there is a rigorous privacy definition that is fulfilled by adding appropriate noise either by each user separately or by a trusted party, in a centralized fashion. Differential privacy assures that an adversary who learns a function over two sets of data values that differ at most by one value cannot tell with which data set it communicates. Alternatively, differential privacy prevents the adversary from determining the existence or absence of an individual data value through a statistical function.

  Noise-based techniques are inappropriate in scenarios where precision of the final data analysis operation is of crucial importance: charging users based on noisy measurements would never be acceptable by end users be it for the benefit of individual privacy. Cryptographic protocols cope with this limitation by achieving precision in the computation of the statistical information by the Aggregator.

- **Cryptographic Protocols** [4, 15, 21, 42, 45, 52, 63, 82, 87, 92, 104, 124, 132]. Privacy Preserving Data Collection and Analysis protocols have been modeled within a rigorous cryptographic framework. Conventional privacy and security definitions have been changed in order to follow the new privacy and security requirements for Privacy Preserving Data Collection and Analysis protocols. Namely, security definitions based on indistinguishability aim to capture an adversarial behavior, which an adversary seeks to retrieve individual data inputs from the ciphertexts and the result of a function over the plaintext data. In order to assure confidentiality of inputs, cryptographic protocols often use a fully trusted key dealer, which distributes secret keys to the users and to the Aggregator.

In order to avoid noise at the final computation of the aggregate value, in current cryptographic solutions for **PPDCA** protocols, data is encrypted appropriately, such that an Aggregator cannot learn any information from the encrypted samples, except for some aggregate result over users' data. The cryptographic solutions assume an *honest-but-curious* threat model with a fully trusted key dealer, which distributes keys to users and to the Aggregator. The amount of trust that is placed in a single party during the protocol may not be realistic for real world scenarios in which parties are mistrustful. Furthermore, in a resource constraint environment such as the smart metering scenario, extra features such as accommodating a dynamic population of users during the protocol execution and achieving resiliency to failures, affect the efficiency of the protocol.

1.4 Goals and outline

Although the most suitable approach for the **PPDCA** problem seems to be the one using cryptographic protocols, existing solutions still suffer from several limitations. First, existing protocols only support a basic set of aggregate functions: their extension thereof seems to be a very suitable research challenge. Moreover, existing cryptographic protocols for **PPDCA** suffer from unrealistic key management requirements due to their reliance on a fully trusted key dealer and the need for updating the key material for the entire user population. Existing users of the scheme are also affected in case of faults because users which already participate in the protocol, need to receive new keys. In case of low-resource devices such as smart meters, it is of great importance to support dynamic groups and resiliency to failures with low communication cost, due to the resource constraints of the devices. Finally, the cryptographic protocols follow the *honest-but-curious* threat model, in which the Aggregator is semi-trusted to follow the protocol's rules. We conclude that it is important to introduce a stronger security model taking into account more powerful adversaries seeking to deviate from the protocol rules, to maliciously tamper with the global results. The objectives of this dissertation can be summarized in a few points:

1. Provide new functions an Aggregator can perform on data for secure data collection and analysis that are not available from the existing cryptographic protocols. We stress that the extended functions should come with an ideal payoff for privacy without compromising it to a large extent–i.e: *Learning aggregate statistics over the entire population of users is doable and acceptable, but learning how each user of the population behaves violates individual privacy.*

2. Design a suitable cryptographic protocol for a dynamic population of users with resiliency to faults. We emphasize that the support of *dynamicity* and *fault-tolerance* should not jeopardize individual privacy.

3. Formalize novel security definitions that specifically strengthen existing privacy definitions with respect to Aggregator obliviousness. We lower the amount of trust that should be placed on any single entity. Moreover, we propose an integrated security definition that guarantees both privacy and verification on computations.

The structure of this dissertation is organized as follows:

- In Chapter 2, we provide cryptographic material that will help the reader interpret the technical material of this dissertation: The Chapter starts with some historical breakthroughs and continues with a comprehensive analysis of security definitions. Next, we highlight the cryptographic primitives that are employed along this dissertation and we explain the method of reasoning for security in the provable security framework.

- In Chapter 3, we present a detailed analysis of the state-the-of-art. We start our analysis with relevant noise-based techniques for Privacy Preserving Data Collection and Analysis. Then, we introduce a thorough analysis of cryptographic protocols, which are classified based on the threat model that they adhere to. Finally, we proceed into a taxonomy of current cryptographic protocols and we identify their shortcomings.

- In Chapter 4, we present our protocol for privacy preserving clustering. The underlying idea is based on a novel transformation of multidimensional vectors that represent data with bi-vectors. Privacy is preserved with individual randomness properly chosen, such that clustering on data can be performed by an untrusted party without violating individual privacy.

- In Chapter 5, we discuss our solution for enhanced functionality in a smart-grid environment. Namely, an untrusted data Aggregator which can be an energy supplier in a real world application is interested in learning continuous periods of high energy consumption of a home. The underlying idea is a randomization of a delta encoding function that takes as input the differences of energy consumption. The delta encodings along with an order preserving encryption scheme allows the untrusted data Aggregator to exclude undesired data from its analysis.

- In Chapter 6, we propose a privacy preserving protocol for sum computation. In contrast with previous work that is based on the trustworthiness of a key dealer to guarantee individual privacy, our solution relaxes this requirement. We also formulate the new privacy requirements with a stronger privacy model. Our protocol is suitable for a dynamic population of users. In this scenario, users can spontaneously join or leave the protocol without any coordination. Existing users are not affected with extra computational and communication overhead since there is no need for a new key distribution phase. Finally, we provide benchmark results for our prototype implementation on PCs, single board computers (`cubieboard` [2]) and mobile devices.

- In Chapter 7, we formalize security properties for aggregation protocols that entail a dual security guarantee. The protocol assures verification of the correctness of computations performed by an untrusted Aggregator and individual privacy against a malicious party. We also implemented our protocol and we present our benchmark results. Our solution achieves constant time verification and is provably secure under a new assumption whose security evidence is proved in the generic group model.

- Finally in Chapter 8, we conclude with the results of this dissertation and we discuss future research avenues.

Chapter 2

Preliminary Topics

Contents

In this section we provide the necessary background in order to enable the reader to interpret the cryptographic machinery that is employed in this dissertation. We first give a historical overview of cryptography. We also present the hardness assumptions that the security of our protocols is based on. Finally, we discuss the provable security framework and we present the cryptographic primitives used in our protocols.

2.1 Brief history

The word cryptography stems from the Greek world κρυπτός (kryptos) which means *hidden* and γραφείν (graphein) which depicts the notion of *writing*. It is the first descendant of the more general term of cryptology (λογία (logy) means study). The other descendant is cryptanalysis. Cryptography aims to write secret codes which are decoded only with the employment of a secret key. Even if the history has encountered the first notion of secret communication back in 2000 BC, cryptography has become a science only in the last decades.

Historians have observed some unusual notation to the egyptian *hieroglyphics* used to decorate the tombs of kings [49, Chapter 2]. They deliberately used some cryptic symbols, which were not part of the existing alphabet, in order to make the stone more *royal*. But it is not clear yet if the main purpose of these cryptic symbols was to hide the underlying message or to act as an eye-catcher to decoy people to decipher it.

The first official use of cipher was observed in 500 BC. Ancient Greeks in order to hide messages they implemented a transposition cipher with a *skytale* [98], which is a wooden cylinder. The sender wraps a strip of parchment wound or leather around it and the message is written across the strip. Then, the sender unwraps the strip and random alphabet symbols were written to random positions of the strip. In order to decipher, the receiver wraps again the strip in a same diameter cylinder that acts as the secret key. During this period, Polybious invented the first substitution cipher, known as *Polybious square* whereby letters grouped in a square were substituted with a sequence of numbers. Some centuries later, at 58 BC, the *ceasar* cipher, which took its name from the senator Gaius Julius Caesar, is based on Polybious idea, and swifts alphabet letters based on a fixed pattern.

Al-Kindi, an Arab mathematician and philosopher made a tremendous progress in ciphers. After identifying the weakness of single alphabet substitution ciphers due to frequency analysis he realized the need for polyalphabet ciphers. The *vigenère* cipher was first described by *Giovan Battista Bellaso* but the name has been wrong misattributed to Blaise de Vigenére due to a stronger description of Bellaso's cipher. Namely, *vigenère* cipher encrypts by applying different alphabets to each letter of the plaintext.

In the 20th century the employment of electro-mechanical rotor machines were securing the communication from host to host. The most well known is *Enigma* that has been used by Nazi and was broken by Allies forces in Bletchley Park. *Enigma* is a polyalphabet stream cipher whereby at each key pressing the three rotors change the circuit connections between the keyboard and the lampboard, thus enabling a different cipher letter for the same plaintext letter; resulting in $\approx 1.76 \cdot 10^{15}$ different keys. James H. Ellis captured the idea of public key encryption in 1969 [136], however it did not become known since he was committed to the Government Communication Headquarters (*GCHQ*). In 1973 Clifford Cocks first realized a public key encryption scheme based on the difficulty of factoring large prime numbers but it was never published, since for the same reasons with Ellis, his work was classified information for *GCHQ*. The work was kept secret until 1997 [136]. Public key encryption became widely known by Diffie and Hellman in 1975 from the publication of their research results. Rivest, Shamir and Adleman in 1976 rediscovered the RSA algorithm, since Cocks' work was not published [1].

2.2 Provable security

2.2.1 Computational security

Computational security definitions bound the adversary to be successful only if it runs in any reasonable time, where reasonable is polynomial on input of the key size. In the seminal work by Goldwasser and Micali [78], the authors introduced the notion of semantic security which is employed to describe computational security. Informally, semantic security guarantees that whatever function $f(x)$ a polynomial bounded adversary can infer for the plaintext x by having access to some auxiliary information $I(k)$, where k is the size of the ciphertext, it can be deduced by another adversary who has not access to the ciphertext. Semantic security models the adversaries as polynomial time algorithms and analyzes the success probabilities with asymptotics. The latter allows to analyze the growth of complexity of the adversary with respect to the size of the key and the auxiliary information.

Definition 1. *(Semantic Security [74, Chap 5]) An encryption scheme* $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ *is semantically secure if for every function* $f : \{0,1\}^k \to \{0,1\}^*$, $I : \{0,1\}^k :\to \{0,1\}^*$ *for every polynomial time adversary A:*

$$\Pr[A(\mathsf{Enc}(k,x), I(k)) = f(x)] - \Pr[A(I(k)) = f(x)] = \frac{1}{2} + \epsilon$$

for a negligible function ϵ.

The above definition has an equivalent version [74, Chap 5] in terms of indistinguishability of two ciphertexts:

Definition 2. *(Message indistinguishability) An encryption scheme* $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ *is message indistinguishable if for any two messages* x_1, x_2 *and any polynomially bounded adversary A:*

$$\Pr[A(\mathsf{Enc}(k,x_1)) = 1] - \Pr[A(\mathsf{Enc}(k,x_2)) = 1] = \frac{1}{2} + \epsilon$$

for a negligible function ϵ. The unduly revolutionary idea that paved the way to treat cryptography not in an ad-hoc framework where experience and intuition play a significant role in solutions but in a general scientific domain, with rigorous mathematical formulations is due to Goldwasser and Micali [77, 78] in the 1980s. The idea designates that in order to prove the security of a protocol, it suffices to construct an efficient polynomial time algorithm that reduces the security of the new protocol to an alleged hard problem. That is, as long as the underlying problem is intractable the scheme is secure. From another perspective it means that once there is a probabilistic polynomial time (PPT) algorithm $\mathcal{A}$ that is able to break the protocol that it is analyzed, then there is an efficient PPT algorithm $\mathcal{B}$ that can break the underlying alleged hard problem. The latter connotes a contradiction so the initial proposition of the security of the scheme holds.

As already mentioned by Bellare [16] the term provably secure is misleading. The existence of reductionist security proofs does not imply a rigorous proof that the scheme is secure in a real world depiction. In contrast, it provides evidence for the security of the scheme as long as an assumption for the hardness of a well known problem holds. In order to clarify things hereafter whenever it is said that *a protocol is provably secure*, it means that there is a reductionist proof from the hardness assumption of the protocol to an alleged hard problem.

2.2.2 Proof roadmap

The technique of providing a reductionist security proof is the following:

- **Protocol description**: In the first phase formal definitions about the functionality of the protocol are provided. More specifically the engaged parties are presented as algorithms. Information about the characteristics of the algorithms are also mentioned: *deterministic*, *probabilistic*, *interactive*, *non-interactive*. At this phase no details about the initialization of the protocol are provided.

- **Security definition**: The security definitions entail rigorous mathematical formulations about what security guarantees the protocol should provide. The definition comes in terms of a polynomial bounded adversary and a quantitative definition of probabilities that depict the chances that the adversary can mount a reasonable attack to the protocol.

- **Threat model**: In the threat model the capabilities of the adversary are clarified. That is, how the polynomial algorithm of the adversary is upper bounded in terms of computational power (polynomially), how the adversary is going to attack the scheme (adaptive, non-adaptive) and what the adversary is allowed to know. In general, the less restrictions to the adversarial model the stronger security guarantee is assured. Adversary's interactions with the protocol can be presented in the form of a *game*, which is played between the adversary and a simulator that simulates the execution of the protocol. In a nutshell, there is a *learning phase* in the game where the adversary learns auxiliary information about the protocol and a *challenge phase* that follows. During the challenge phase a challenge is submitted to the adversary by the simulator and the former succeeds in the game if it can provide a correct answer to the challenge with non-negligible probability.

- **Protocol realization**: During the protocol realization the tools that realize the protocol description are presented. Moreover parameters are analyzed in order to meet security definitions and efficient functionalities.

- **Reductionist proof**: A reductionist proof from the protocol to a hard problem is described: The steps for the construction of an algorithm $\mathcal{B}$ are presented. More specifically $\mathcal{B}$ employs as a subroutine an adversary $\mathcal{A}$ against the proposed scheme, in order the former to break a believed hard problem. The proof should be as tight as possible in order to be meaningful. That is, if $\mathcal{A}$'s running time is polynomially bounded by t with success probability e, then B running time and success probability should be as close as possible to t, e respectively.

2.2.3 Game-hopping technique

Within the *game-based* framework after the definition of a game G, then the advantage **Adv** of the adversary is modeled as the probability of an event S to occur. This is the *target probability*, which should be negligible when the adversary has to compute something (e.g: forge a signature) or it should be negligible between her choices when she has to distinguish between two events (indistinguishability security of encryption). It is quite common both for technical reasons and for clarity and simplification in order to avoid flaws, the proof to follow the *game-hop* technique [54, 135]. The original game is defined in G_0 and then a sequence of games $G_1, G_2, G_3, \ldots, G_n$ are further presented. As previously mentioned, this happens in order to make the proof more readable and easy to analyze but also because there are difficulties in proving the negligibility of the target probability. The goal of the designer is to show that the advantage of the adversary in each game is negligible close to its predecessor: $\Pr[S_i] = \Pr[S_i + 1], i \in [0, n]$ and that $\Pr[S_n]$

equals the *target probability*.

2.2.4 Random oracle

So far we demonstrated the roadmap for provable security and the techniques that are employed towards it. When the reductionist proof is possible without making any further assumptions then the protocol is provably secure in the *standard-model*. However this is not always possible. In the vast majority of existing protocols a source of randomness is mandatory. But it is hard to instantiate a truly random function. A random function can be viewed as a hard-coded lookup table. On the left side of the lookup table there is the input to the function that consists of all bit-strings of the universe and on the right side there is the random output, different for each input. This construction is totally impractical in terms of storage and of retrieval of the random output for a specific bit string. The problem is mitigated with the random oracle [17], which is viewed as an extra party in the protocol. Everybody can query the oracle. At the beginning of the protocol the oracle's state is empty. Then, each time a party is querying the oracle with a bit string, the oracle checks if that has been queried before with the same input. If yes, it returns the same random output string. If no, it returns a new random output string and it stores the new pair of input-output to its state. The assumption that is made in the proofs in the random oracle is that they pretend the existence of a function that outputs truly random bit strings like in an ideal scenario.

The random oracle model allowed for the construction of security proofs that were infeasible in the standard model. However, when it comes to the real-world model the random oracle is substituted with a hash function which is not truly random. Researchers have come up with several scenarios [33], whereby the implementation of a protocol using real hash functions suffer from severe security flaws despite the security proof in the random oracle model. Moreover the random oracle model does not depict all types of adversaries. Attackers that have physical access to the machine that implements a cryptographic protocol can mount different types of *side channel attacks* in order to learn the secret key: a) by observing the running time of secret key conditional loop [100], b) by observing the power analysis of a device [101], c) by capturing the acoustic signal that is generated by the vibrations of electronic components of a computer [70], or d) by having access to the memory of a PC and applying a *cold-boot-attack* [83]. So the question to answer is: *why are we still applying the idea of the random oracle model for security proofs*? The reason is because a security proof in the random oracle model is better than no security proof at all. Moreover, when the adversary does not focus on attacking the real world pseudorandom function then the random oracle provides security guarantees for generic attacks of the scheme. Last but not least, during the instantiation of the random oracle with a specific pseudorandom function, if there are weaknesses for this specific function then we can simply substitute it with another one, without changing the protocol overall.

2.2.5 Generic group model

It may not always be possible to rely the security of a cryptographic protocol on well known problems. As such, the reductionist proof should be made on a new problem. In order to provide security evidence of the new problem Shoup introduced the *generic group model* [134]. This is a theoretical framework, whereby the hardness of the new assumption is analyzed [11, 26, 137]. According to the generic group model, an adversary has access to elements of the underlying mathematical group in which the problem is assumed to be hard, through a black box. The black box acts as a simulator which uses a random encoding function in order to reply for encodings of elements the adversary asks for. Moreover the simulator associates with each random encoding a polynomial which is not forwarded to the adversary. The simulator also simulates the mathematical operations of the underlying group and any other information the adversary is allowed to have access to. Finally, the adversary outputs a solution to the problem which is translated into an equation of the corresponding polynomials. The probability to successfully evaluate a polynomial of degree d to 0 with a random variable assignment from a group of order p, bounds the probability of an adversary to break the assumption according to the Schwartz-Zippel lemma:

Lemma 1. *Let $p(x_1, \cdots, x_n)$ be a polynomial of total degree d. Assume that p is not identically zero. Let $\mathbb{S} \subseteq \mathbb{F}$ be any finite set. Then, if we pick $y_1, \cdots, y_n$ independently and uniformly from*

$\mathbb{S}$,

$$\Pr[(p(y_1, \cdots, y_n) = 0] \leq \frac{d}{|S|}$$

2.3 Mathematical background

In this section we review mathematical structures that are used in this thesis and we highlight the mathematical problems that the security of the proposed protocols is based on. We start our review with a black-box definition of what a bilinear pairing is without presenting the underlying mathematical and algorithmic subtleties.

2.3.1 Bilinear pairings

Miller and Koblitz [99, 114] proposed the neat idea to use groups that come from algebraic geometry such as elliptic curve groups since no sub-exponential algorithm is known for the discrete logarithm problem in elliptic curve groups. The reason of the broad usefulness come from the existence of efficient algorithms for mathematical operations under these groups and from the fact, that security scales much nicer with the security parameters. Elliptic curves with a richer structure can be used to instantiate a bilinear pairing. Two examples of pairings over elliptic curves are the Weil and Tate [67] pairings. An elliptic curve is defined over a cubic equation $y^2 = x^3 + ax + b$.

Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be three cyclic groups of order p. A bilinear pairing from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$ is a map e:

$$e(\mathbb{G}_1, \mathbb{G}_2) \rightarrow \mathbb{G}_T$$

satisfying the following properties:

1. *bilinearity*: $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$, for $g_1, g_2 \in \mathbb{G}_1 \times \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$
2. *Computability*: there exists an efficient algorithm that computes $e(g_1, g_2)$ $\forall g_1, g_2 \in \mathbb{G}_1 \times \mathbb{G}_2$
3. *Non-degeneracy*: $e(g_1, g_2)$ generates $\mathbb{G}_T$. That is, if g_1, g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, then $e(g_1, g_2)$ generates $\mathbb{G}_T$.

We say that a bilinear pairing for cryptographic protocols is admissible if the aforementioned three properties are satisfied. Bilinear pairings can be categorized in the following three types with respect to supported functionalities and the underlying group structure:

- **Type I**: In type I pairings $\mathbb{G}_1 = \mathbb{G}_2$.
- **Type II**: In type II pairings $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is an efficient computable map $\phi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$.
- **Type III**: In type III pairings $\mathbb{G}_1 \neq \mathbb{G}_2$ and there are not efficient computable maps $\phi_1 : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ or $\phi_2 : \mathbb{G}_1 \rightarrow \mathbb{G}_2$.

Supersingular curves are employed for **Type I** symmetric pairings and are less efficient than curves for asymmetric pairings [68, 122]. MNT [117] or BN [14] non-supersingular curves are used to realize **Type II** and **Type III** asymmetric bilinear pairings. Cryptographic pairings are used for different type of protocols: three-party key exchange, identity based encryption, group signatures, homomorphic signatures, aggregate signatures [27–30, 91].

Before presenting the hardness assumption we define the notion of a negligible function f:

Definition 3. *A function $f : \mathbb{Z} \rightarrow \mathbb{R}$ is negligible if for all $c > 0$ there exists $\lambda_0 > 0$ such that for all $\lambda \geqslant \lambda_0 : f(\lambda) \leq \frac{1}{\lambda^c}$*

2.3.2 Algorithmic complexity

An *algorithm* is a finite set of steps used to solve a computational problem that takes as input variable length of data and outputs the result. The term *deterministic algorithm* refers to algorithms that on equivalent inputs they output equivalent results following exactly the same algorithmic path of steps. A *probabilistic algorithm* receives as input, a randomness r from a source of randomness $r \xleftarrow{\$} R$ as "*random coins*" and its output-algorithmic path is not equivalent even on equivalent data inputs.

The running time of an algorithm reflects its efficiency. It is defined as the number of steps the *algorithm* has to execute in order to terminate. As a step we consider the cardinality of the set that includes the mathematical operation as multiplications, exponentiations, additions, subtractions and divisions used by the *algorithm* . Even if this is not extremely precise as it does not entail the machine instructions that are used to implement the mathematical operations of the data inputs, it is meaningful and provides a rigorous approach in order to compare protocols and primitives. An algorithm is said to be *efficient* if its running time is *polynomial* on the size of inputs.

2.3.3 Hardness assumptions

Definition 4. *(Discrete Logarithm (**DL**) Problem) Let $\mathbb{G}$ be a cyclic group of prime order p with generator g. The **DL** problem is defined as follows: given $h \in \mathbb{G}$ compute $x \in \mathbb{Z}_p$ such that $g^x = h$. We say that x is the discrete logarithm of h to the base $g : x = \log_g(h)$. The advantage $\mathbf{Adv}_{\mathcal{A}}^{\mathbf{DL}}$ of an algorithm $\mathcal{A}$ in solving **DL** is defined as:*

$$\mathbf{Adv}_{\mathcal{A}}^{\mathbf{DL}} = \Pr[x \leftarrow \mathcal{A}(h, g) : x = \log_g(h)]$$

Definition 5. *(Discrete Logarithm (**DL**) Assumption) Let $\mathcal{G}(\lambda)$ be a **DL** problem generator that outputs h on input of the security parameter λ. We say that **DL** assumption holds if the advantage $\mathbf{Adv}_{\mathcal{A}}^{\mathbf{DL}}$ of a probabilistic polynomially-time algorithm $\mathcal{A}$ is a negligible function $\epsilon(\lambda)$ on input of the security parameter λ.*

Definition 6. *(Computational Diffie-Hellman(**CDH**) Problem) Let $\mathbb{G}$ be a cyclic group of prime order p and g a generator of $\mathbb{G}$. The **CDH** problem is: Given $U = (g, g^x, g^y)$ for uniformly random elements $x, y \in \mathbb{Z}_p$, compute g^{xy}. The advantage $\mathbf{Adv}_{\mathcal{A}}^{\mathbf{CDH}}$ of an algorithm $\mathcal{A}$ in solving **CDH** is defined as:*

$$\mathbf{Adv}_{\mathcal{A}}^{\mathbf{CDH}} = \Pr[g^{xy} \leftarrow \mathcal{A}(U)]$$

Definition 7. *(Computational Diffie-Hellman(**CDH**) Assumption) Let $\mathcal{G}(\lambda)$ be a **CDH** problem generator that outputs U on input of the security parameter λ. We say that **CDH** assumption holds if the advantage $\mathbf{Adv}_{\mathcal{A}}^{\mathbf{CDH}}$ of a probabilistic polynomially-time algorithm $\mathcal{A}$ is a negligible function $\epsilon(\lambda)$ on input of the security parameter λ.*

Definition 8. *(Decisional Diffie-Hellman(**DDH**) Problem) Let $\mathbb{G}$ be a cyclic group of prime* *order p and g a generator of $\mathbb{G}$. The* **DDH** *problem is: Given $U = (g, g^x, g^y, g^z)$ for uniformly random elements $x, y, z \in \mathbb{Z}_p$, decide whether $z = xy$. Define the probability distribution X that samples elements from $Z = (g, g^x, g^y, g^{xy})$ and the probability distribution Y that samples elements from $U = (g, g^x, g^y, g^z)$. The advantage $\mathbf{Adv}_{\mathcal{A}}^{\mathbf{DDH}}$ of an algorithm $\mathcal{A}$ in solving* **DDH** *is defined as:*

$$\mathbf{Adv}_{\mathcal{A}}^{\mathbf{DDH}} = |\Pr[\mathcal{A}(X) = 1] - \Pr[\mathcal{A}(Y) = 1]|$$

Definition 9. *(Decisional Diffie-Hellman(***DDH***) Assumption) Let $\mathcal{G}(\lambda)$ be a* **DDH** *problem generator that outputs U on input of the security parameter λ. We say that* **DDH** *assumption holds if the advantage $\mathbf{Adv}_{\mathcal{A}}^{\mathbf{DDH}}$ of a probabilistic polynomially-time algorithm $\mathcal{A}$ is a negligible function $\epsilon(\lambda)$ on input of the security parameter λ.*

Definition 10. *(Quadratic Residuosity (***QR***) Problem) Let $N = pq$ be a product of two large primes. Define the probability distribution X over elements drawn from $\widehat{\mathbb{QR}}_N$, the group of quadratic non residues in $\mathbb{Z}_N^*$ and define the probability distribution Y with elements drawn from $\mathbb{QR}_N$, the group of quadratic residues in $\mathbb{Z}_N^*$. The* **QR** *problem is to computationally distinguish between probability distributions X and Y. The advantage $\mathbf{Adv}_{\mathcal{A}}^{\mathbf{QR}}$ of an algorithm $\mathcal{A}$ in solving* **QR** *is defined as:*

$$\mathbf{Adv}_{\mathcal{A}}^{\mathbf{QR}} = |\Pr[\mathcal{A}(X) = 1] - \Pr[\mathcal{A}(Y) = 1]|$$

Definition 11. *(Quadratic Residuosity (***QR***) Assumption) Let $\mathcal{G}(\lambda)$ be a* **QR** *problem generator that outputs X, Y on input of the security parameter λ. We say that* **QR** *assumption holds if the advantage $\mathbf{Adv}_{\mathcal{A}}^{\mathbf{QR}}$ of a probabilistic polynomially-time algorithm $\mathcal{A}$ is a negligible function $\epsilon(\lambda)$ on input of the security parameter λ.*

Definition 12. *(Decisional Composite Residuosity (***DCR***) Problem) Let $N = pq$ be a product of two large primes. Define the probability distribution X over elements drawn from random elements in the set of $\mathbb{Z}_{N^2}^*$ and define the probability distribution Y with elements drawn from $\langle x^N \bmod N^2, x \in \mathbb{Z}_{N^2}^* \rangle$. The* **DCR** *problem is to computationally distinguish between probability distributions X and Y. The advantage $\mathbf{Adv}_{\mathcal{A}}^{\mathbf{DCR}}$ of an algorithm $\mathcal{A}$ in solving* **DCR** *is defined* *as:*

$$\mathbf{Adv}_{\mathcal{A}}^{\mathbf{DCR}} = |\Pr[\mathcal{A}(X) = 1] - \Pr[\mathcal{A}(Y) = 1]|$$

Definition 13. *(Decisional Composite Residuosity (**DCR**) Assumption) Let $\mathcal{G}(\lambda)$ be a **DCR** problem generator that outputs X, Y on input of the security parameter λ. We say that **DCR** assumption holds if the advantage $\mathbf{Adv}_{\mathcal{A}}^{\mathbf{DCR}}$ of a probabilistic polynomially-time algorithm $\mathcal{A}$ is a negligible function $\epsilon(\lambda)$ on input of the security parameter λ.*

Definition 14. *(Bilinear Computational Diffie-Hellman (**BCDH**) Problem) Let $e(\mathbb{G}_1 \times \mathbb{G}_2) \rightarrow \mathbb{G}_T$ be a bilinear pairing, g_1 a generator of $\mathbb{G}_1$ and g_2 a generator of $\mathbb{G}_2$ and p the order of $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$. The **BCDH** problem is defined as follows:*

*Given $U = (g_1, g_1^x, g_1^y, g_1^z) \in \mathbb{G}_1$ and $V = (g_2, g_2^x, g_2^y) \in \mathbb{G}_2$ for random $x, y, z \in \mathbb{Z}_p$ compute $W = e(g_1, g_2)^{xyz}$. The advantage $\mathbf{Adv}_{\mathcal{A}}^{\mathbf{BCDH}}$ of an algorithm $\mathcal{A}$ in solving **BCDH** is defined as:*

$$\mathbf{Adv}_{\mathcal{A}}^{\mathbf{BCDH}} = \Pr[W \leftarrow \mathcal{A}(U, V)]$$

Definition 15. *(Bilinear Computational Diffie-Hellman (**BCDH**) Assumption) Let $\mathcal{G}(\lambda)$ be a **BCDH** problem generator that outputs U and V on input of the security parameter. We say that **BCDH** assumption holds if the advantage $\mathbf{Adv}_{\mathcal{A}}^{\mathbf{BCDH}}$ of a probabilistic polynomially-time algorithm $\mathcal{A}$ is a negligible function $\epsilon(\lambda)$ on input of the security parameter λ.*

2.4 Cryptographic primitives

In this section, we introduce the cryptographic primitives that are used throughout this dissertation.

2.4.1 Pseudorandom generators (PRG)

As Kolmogorov conceptually defined it, a k bit string is random if a polynomial time algorithm cannot reproduce it in less that k steps. However the difficulty of finding real random bit sequences for the keys has shifted the interest to the existence of pseudorandom generators (PRG). Informally, a pseudorandom generator G is a deterministic polynomial time algorithm that takes as inputs a *seed* of length l and outputs a bit string of length L where $L >> l$, whose output is *computationally indistinguishable* from a uniform distribution.

We now give the definition of computational indistinguishability for two distribution ensembles, which are modeled as sequences of random variables $X = \{X_n\}_n \in \mathbb{N}, Y\{Y_n\}_n \in \mathbb{N}$. Before going into the formal definition we expand the notation $\Pr[A(X_n) = 1] = \sum \Pr[A(x) = 1] \Pr[X_n = x]$ which is used for the definition.

Definition 16. *For any polynomial time algorithm $\mathcal{A}$ two distribution ensembles $X = \{X_n\}_n \in \mathbb{N}, Y = \{Y_n\}_n \in \mathbb{N}$ are computationally indistinguishable if:*

$$\Pr[\mathcal{A}(X_n) = 1] - \Pr[\mathcal{A}(Y_n) = 1] = \epsilon$$

where e is negligible function on n.

Definition 17. *A pseudorandom generator G is a function G: $\{0,1\}^l \to \{0,1\}^L$, whose outputs are computational indistinguishable from uniform bit strings of the same length L.*

2.4.2 Pseudorandom functions (PRF)

Let the family of all functions in the universe from a domain X to a range Y to be $Func[X, Y]$. A truly random function $f \xleftarrow{\$} Func[X, Y]$ is chosen randomly from the set of $Func$. The set of all these functions is $|Y|^{|X|}$ (gigantic number). It is true that for any random function f with range size L chosen randomly from $Func[X, Y]$, $\Pr[f(x) = y] = 2^{-L}$. The randomness is not parametrized neither by the size of X and Y nor by the size of the domain. We define a pseudorandom function $f_k : X \to Y$ as a function from the set of all functions from X to Y as soon as a particular key k is fixed.

Definition 18. *Let Func=$\{F : X \to Y\}$ be a function family for all functions F that map elements from the domain X to the range R. Then a $PRF = \{f_k : X' \to Y'\} \subseteq Func$ for $k \xleftarrow{\$} K$, where K is the key space.*

The security of a PRF is modeled with a game which is known as *real or random* security game [75]. Intuitively, an adversary $\mathcal{A}$ is given access to an oracle that on input x from a domain

X, flips a coin $b \xleftarrow{\$} \{0,1\}$ and if $b = 0$ then it outputs $y = f(x)$, for $f \in Func[X,Y]$, otherwise it outputs $y = f_k(x)$. $\mathcal{A}$ issues queries to the oracle polynomially many times on input of the security parameter λ. Finally $\mathcal{A}$ outputs a guess b' for the bit b.

The advantage of a probabilistic polynomial time algorithm $\mathcal{A}$ in the PRF game is

$$\mathbf{Adv}_{\mathcal{A}}^{PRF} = \Pr[b \xleftarrow{\$} \{0,1\}; b' \leftarrow \mathcal{A}(y) : b' = b]$$

Definition 19. *A PRF is computationally secure if all probabilistic polynomially time algorithms $\mathcal{A}$ have advantage in the PRF game: $\frac{1}{2}+\epsilon(\lambda)$, for a negligible function ϵ on the security parameter λ.*

2.4.3 Pseudorandom permutations (PRP)

A permutation is a bijective function where the domain and the range are equal. Similarly with the random functions, let $Perm[X]$ to be the set of all permutations for the domain X. Then a pseudorandom permutation (PRP) is a randomly chosen permutation from the set $Perm[X]$, keyed under a secret key k. Similarly with a PRF, the security of a PRP is described with a game in which a polynomially bounded adversary $\mathcal{A}$ tries to distinguish permutations between the family of all permutations from the pseudorandom ones. An oracle receives an input value $x \in X$ from the adversary $\mathcal{A}$. It flips a random coin $b \xleftarrow{\$} \{0,1\}$. If $b = 0$ then it chooses a random permutation Π from the set of all permutations $Perm[X]$ and returns to $\mathcal{A}$, $y = \Pi(x)$, otherwise if $b = 1$ the oracle chooses a random $k \xleftarrow{\$} K$ from the keyspace K, sets a pseudorandom permutation Π_k and forwards to $\mathcal{A}, y = \Pi_k(x)$. $\mathcal{A}$ submits queries to the oracle polynomially many times on input of the security parameter λ.

The advantage of a probabilistic polynomial time algorithm $\mathcal{A}$ in the PRP game is

$$\mathbf{Adv}_{\mathcal{A}}^{PRP} = \Pr[b \xleftarrow{\$} \{0,1\}; b' \leftarrow \mathcal{A}(y) : b' = b]$$

Definition 20. *A PRP is computationally secure if all probabilistic polynomially time algorithms $\mathcal{A}$ have advantage in the PRP game: $\frac{1}{2}+\epsilon(\lambda)$, for a negligible function ϵ on the security parameter*

λ.

2.4.4 Hash functions

A hash function H is a keyed function that maps elements of arbitrary length from a domain X to a finite range Y of length l. The number l is the hash length of H. In practice the hash function is not keyed (this implies that the adversary knows how to evaluate the function, or it has access to an oracle that evaluates H on behalf of the adversary without publishing the secret key) and its formal security definitions have been introduced in [126]. The three desired properties of a cryptographically secure hash function are the followings:

First-preimage resistance A hash function H is *first-preimage resistant* if it is polynomial time computable on input of an element $x \in X$ but there is no polynomial time algorithm that can correctly guess x by given only $y = H(x)$. More formally we define the advantage of an adversary $\mathcal{A}_{\mathsf{FPR}}$ that tries to break *first-preimage resistance* as:

$$\mathbf{Adv}^{H_k}_{\mathcal{A}\mathsf{FPR}} = \Pr[k \xleftarrow{\$} \mathcal{K}\ ;\quad x \xleftarrow{\$} X\ ;\ y \leftarrow H_k(x)\ ;\ x' \leftarrow A(k, y) : H_k(x') = y]$$

Definition 21. *A hash function $H : X \to Y$ is first-preimage resistant if the advantage of an adversary $\mathbf{Adv}^{H_k}_{\mathcal{A}\mathsf{FPR}}$ is negligible.*

Second-preimage resistance A hash function H is *second-preimage resistant* if an adversary that learns x_1 and $y_1 = H(x_1)$ cannot find $x_2 \neq x_1$ such that $y = H(x_1) = H(x_2)$. The advantage of an adversary $\mathcal{A}$ that seeks to break the *second-preimage* property is:

$$\mathbf{Adv}^{H_k}_{\mathcal{A}\mathsf{SPR}} = \Pr[k \xleftarrow{\$} \mathcal{K}\ ;\quad x \xleftarrow{\$} \mathcal{X}\ ;\ y \leftarrow H_k(x)\ ;\ x' \leftarrow A(k, x, y) : (x \neq x') \wedge (H(x) = H(x'))]$$

Definition 22. *A hash function $H : X \to Y$ is second-preimage resistant if the advantage of an adversary $\mathbf{Adv}^{H_k}_{\mathcal{A}\mathsf{SPR}}$ is negligible.*

Collision resistance The collision resistance property assures that any polynomially bounded adversary cannot find any $x_1 \neq x_2$ such that $y = H(x_1) = H(x_2)$. More formally the advantage of an adversary $\mathcal{A}_{\mathsf{CR}}$ against the collision resistance property is defined as:

$$\mathbf{Adv}^{H_k}_{\mathcal{A}\mathsf{CR}} = \Pr[k \xleftarrow{\$} \mathcal{K}\ ;\ \ (x, x') \leftarrow A(k) : (x \neq x') \wedge (H(x) = H(x'))]$$

Definition 23. *A hash function $H : X \to Y$ is collision resistant if the advantage of an adversary $\mathbf{Adv}^{H_k}_{\mathcal{A}\mathsf{CR}}$ is negligible.*

2.4.5 Digital signatures

A signature scheme $\Sigma = (\mathbf{KeyGen}, \mathbf{Sign}, \mathbf{Ver})$, which is used to provide message integrity and sender's authenticity, consists of a message space $\mathcal{M}$, a signature space $\mathcal{S}$, and three algorithms:

- $\mathbf{KeyGen}(1^\lambda)$: It takes as input the security parameter 1^λ and outputs the secret signature key sk and the public verification key vk.
- $\mathbf{Sign}(\mathsf{sk}, m)$: It takes as input a message $m \in \mathcal{M}$, the signature key sk and outputs the signature σ_m.
- $\mathbf{Verify}(\mathsf{vk}, \sigma_m)$: It takes as input a candidate signature σ_m and the public verification key vk and it outputs 1 if σ_m is a valid signature of m or $\perp$ otherwise.

A signature scheme Σ is correct if for all $\lambda \in \mathbb{N}$, all $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathbf{KeyGen}(1^\lambda)$, all $m \in \mathcal{M}$ and all signatures $\sigma_m \leftarrow \mathbf{Sign}(\mathsf{sk}, m)$ then it is always true that $\mathbf{Verify}(\mathsf{vk}, \sigma_m) = 1$.

The existential unforgeability against adaptive chosen message attacks security property of a signature scheme is defined through a game with the following phases:

- **Setup** : Adversary $\mathcal{A}$ receives the public verification key vk and the security parameter 1^λ from the challenger which runs the $\mathbf{KeyGen}(1^\lambda)$ algorithm of the signature scheme Σ. The challenger also keeps secret the signature secret key sk.
- **Learning** : During the learning phase $\mathcal{A}$ submits signature queries m to the the challenger which responds to $\mathcal{A}$ with the signature σ_m after running the $\mathbf{Sign}(\mathsf{sk}, m)$ algorithm.
- **Challenge** : $\mathcal{A}$ outputs a pair m, σ_m.

$\mathcal{A}$ is said to win the aforementioned game if **Verify**$(\mathsf{vk}, \sigma_m) = 1$ and m has not been queried during the **Learning** phase.

Definition 24. *A signature scheme guarantees existential unforgeability against adaptive chosen message attacks if the advantage* $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{Sign-Forge}}$ *in the above game of a probabilistic polynomially-time algorithm* $\mathcal{A}$ *is a negligible function* $\epsilon(\lambda)$ *on input of the security parameter* λ.

For a detailed analysis of the various threat models for digital signature schemes the reader can refer to [125].

2.4.6 Message Authentication Codes

A message authentication code (MAC) scheme is a cryptographic primitive that provides integrity and sender's authenticity, as digital signatures, but in a symmetric setting. It consists of a message space $\mathcal{M}$, a MAC space $\mathcal{S}$, a secret key space $\mathcal{K}$ and the following algorithms:

- **KeyGen**(1^λ) : It takes as input the security parameter 1^λ and outputs the secret MAC key mk.
- **Mac**(mk, m) : It takes as input a message $m \in \mathcal{M}$, the MAC key $\mathsf{mk} \in \mathcal{K}$ and outputs a tag $t_m \in \mathcal{S}$.
- **Verify**(mk, t_m) : It takes as input a candidate tag t_m, the secret MAC key mk, and a message m. It outputs 1 if $t_m = \mathbf{Mac}(\mathsf{mk}, m)$ or $\perp$ otherwise.

A MAC scheme is correct if for all $\lambda \in \mathbb{N}$, all $(\mathsf{mk}) \leftarrow \mathbf{KeyGen}(1^\lambda)$, all $m \in \mathcal{M}$ and all tags $t_m \leftarrow \mathbf{Mac}(\mathsf{mk}, m)$ then it is always true that $\mathbf{Verify}(\mathsf{mk}, t_m) = 1$.

The security of a MAC scheme is defined through the following game, which conceptually guarantees that an adversary $\mathcal{A}$, which has access to the security parameter λ and the tag algorithm $\mathbf{Mac}(\mathsf{mk}, m)$, to forge a valid tag t'_m, such that $\mathbf{Verify}(\mathsf{mk}, t'_m) = 1$ and m' has not been given as input to the $\mathbf{Mac}(\mathsf{mk}, m)$ algorithm:

- **Setup** : Adversary $\mathcal{A}$ receives the security parameter 1^λ from the challenger which runs the **KeyGen**(1^λ) algorithm of the MAC. The challenger also keeps secret the secret key mk.

- **Learning** : During the learning phase $\mathcal{A}$ submits tag queries m to the the challenger which responds to $\mathcal{A}$ with the tag t_m after running the **Mac**(mk, m) algorithm.

- **Challenge** : $\mathcal{A}$ outputs a pair m, t_m.

$\mathcal{A}$ is said to win the aforementioned game if **Verify**$(\mathsf{mk}, t_m) = 1$ and m has not been queried during the **Learning** phase.

Definition 25. *A* MAC *scheme guarantees existential unforgeability against adaptive chosen message attacks if the advantage* $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{MAC-Forge}}$ *in the above game of a probabilistic polynomially-time algorithm* $\mathcal{A}$ *is a negligible function* $\epsilon(\lambda)$ *on input of the security parameter* λ.

The above definition does not take into account *replay* attacks, which can be mitigated with a time-stamp or a counter. Verification algorithms that are time dependent are vulnerable to timing attacks as long as an adversary has access to a verification oracle which publishes the results of the **Verify**(mk, t_m) algorithm. As such, the **Verify**(mk, t_m) algorithm implementation should run in constant time.

2.5 Summary

In this Chapter we reviewed the basic building blocks that will make the manuscript intelligible. We started with a brief history on cryptography. Next, we proceeded into a primer on the proof technique by analyzing the core steps of a reductionist proof. We also investigated the different types of proofs with respect to the assumptions and finally we presented the cryptographic primitives that are employed in this dissertation.

Chapter 3

Data Collection and Analysis vs Privacy

Contents

3.1 Introduction

Privacy Preserving Data Collection and Analysis protocols (**PPDCA**) exist in the literature with a wide variety of security and functional definitions. In this chapter we survey current related work on **PPDCA** protocols. We start our analysis with results pertaining to *noise-based* techniques which we further classify into two categories: *ad-hoc* based solutions with no formal privacy definition, and differential privacy-based solutions with rigorous formalizations. Afterwards, we investigate *cryptographic* solutions which we categorize them with an assessment based on the threat model that each protocol adheres to. After presenting a taxonomy based on the different characteristics of the cryptographic protocols, we highlight the shortcomings of the current **PPDCA** protocols and we conclude the chapter in section 3.6.

3.2 Noise-based solutions

3.2.1 Ad-hoc

We start with noise-based techniques, in which data is protected with non-cryptographic solutions. Basically n users obfuscate their data, that are forwarded to an untrusted third party called Aggregator. The latter during an *analysis* phase learns a statistical function f over users' data. *Ad-hoc* based noise solutions aim to address the problem with a lack of a precise privacy definition. Each noise-based approach tunes the privacy definition with respect to the functionality f that has to be computed by the Aggregator. In general, the ad-hoc approach consists of a special combination of sensitive data with noise during the *collection* phase in order to allow an untrusted Aggregator $\mathcal{A}$ to perform certain computations on the data [19, 20, 139], without compromising users' privacy.

First Aggrawal *et al.* [7] proposed to add noise r_i to each data value x_i, thus the user returns $y_i = x_i + r_i$ as its own obfuscated value. The Aggregator is interested in learning $\sum_{i=1}^{n} x_i$. The problem boils down to the estimation of the accumulative distribution X_+ knowing the accumulative distribution R_+ of all r_i and each individual distribution Y_i of each perturbed value $y_i = x_i + r_i$. The proposed solution estimates X_+ by applying the Bayes theorem on the underlying distributions. Moreover the authors in [7] suggested to quantify privacy with respect to the confidence c an adversary has that a data value belongs to a specific range:

Definition 26. *Given a confidence interval $c\%$ an adversary has for the possible data range of a single value x_i to be in $[a - b]$, then the scheme guarantees $|a - b|$ privacy level with $c\%$ confidence.*

However, as identified by Aggrawal *et al.* [5] this definition is not rigorous since it does not take into account the underlying probability distributions of the original data values and the perturbed data. The above observation led to a different privacy definition that includes the underlying probability distributions through the *differential entropy* of a random variable. Before examining the contributions as presented in [5] we are giving definitions of the entropy that are used to quantify privacy by the authors.

Definition 27. *The differential entropy of a random variable X with domain Ω and* PDF $= f(X)$ *is defined as:*

$$H(X) = -\int_{\Omega} f(x) \log_2(f(x))\, \mathrm{d}x$$

.

Definition 28. *The conditional differential entropy of a random variable X, conditioned on a random variable Y, with domain Ω and joint* PDF $= f(X,Y)$, *conditional* PDF $= f(X|Y)$ *is defined as:*

$$H(X|Y) = -\int_{\Omega} f(x,y) \log_2(f(x|y))\, \mathrm{d}x\mathrm{d}y$$

.

The authors first proposed a privacy metric inherent to the random variable $X : 2^{H(X)}$ and they also inserted the knowledge of the aggregate value in order to quantify privacy with all the known information by an adversary. The proposed privacy metric $2^{H(X|Y)}$ entails the conditional entropy of a random variable X by knowing another random variable $Y = X + R$. They also came up with another approach to reconstruct the cumulative distribution X_+ from Y_+ within an *expectation maximization* framework. The heuristic is that after applying the *expectation maximization* algorithm on the perturbed data distribution the result will converge to the original data distribution.

The additive noise mechanism for individual privacy has been shown to leak the original data. More precisely in [95] the authors analyzed the additive noise mechanism by modeling it with random matrices. Afterwards they applied signal filtering techniques and they manage to separate the perturbed data from the noisy elements, by applying random matrices theory on the perturbed data. A transformation based on random projections for evaluating euclidean distance and inner product computations is presented in [108]. Namely the authors by exploring the following lemma they manage to randomize both row-wise and column-wise matrices with a random projection mechanism π in order to maintain data utility for inner product computations:

Lemma 2. *(Johnson-Lindenstrauss [90]) Given* $0 < e < 1$*, let* $Q \subseteq \mathbb{R}^d$ *be a set of* n *points. Then, there exists a map* $\pi : \mathbb{R}^d \to \mathbb{R}^k$*, where* $k > 8ln(n)/e^2$*, such that* $\forall x, y \in Q$*:*

$$(1-e)\|x-y\|^2 \leq \left\|\pi(x)-\pi(y)\right\|^2 \leq (1+e)\|x-y\|^2$$

Intuitively the lemma demonstrates that by projecting a vector from a d-dimensional space to a lower random k-dimensional space the pairwise distance of two vectors is preserved within a small factor.

Oliveira and Zaïane [120] examined geometrical transformation on vectors to preserve the cosine similarity of two vectors for clustering data. The underlay observation is that by randomly scaling two vectors v_1 and v_2 or by rotating them with a common angle ϕ during the *collection* phase the cosine similarity is preserved for the *analysis* phase since it quantifies the angle of the two original vectors. A hybrid approach that combines rotation, scaling and vector translation is also presented. Privacy is measured as the variance of the difference between the original data X and the transformed data Y: $\mathsf{VAR}(X-Y)$.

The ad-hoc approach of noise perturbation techniques lacks a generic privacy definition. Namely, all the aforementioned solutions introduce their own privacy quantification techniques without describing a possible adversarial behavior. As such the noise-based solutions previously described cannot be compared due to the absence of a common formal privacy model. This lack laid the ground for differential privacy based solutions with generic formal privacy definitions.

3.2.2 Differential privacy

The idea of differential privacy has its roots in the seminal work of Tore Dalenius [51]. In essence Dalenius paved the way to the cryptographic security definition of *semantic security* [78]. Informally he underpinned the following: *nothing can be revealed for individual data inputs* x_i

to an adversary which has access to a function f, that cannot be learnt without access to f.

This observation has led to the *ad omnia* definition of differential privacy. Within the *differential privacy* framework, users hold personal sensitive data and a Curator stores the data. The goal of the Curator, which is trusted [60, 84, 140], is to output the result of a statistical function f, which takes as input users' data, obfuscated with some noise, such that an adversary cannot learn individual data inputs. The aforementioned security model has been augmented with a stronger security model in which there is a semi-honest Curator, which is not trusted [4, 15, 42, 71, 124, 132] and users distributively introduce noise to their data before sending them to the Curator. Differential privacy solutions are also categorized with respect to the interaction model: when the untrusted party is allowed to interact with the Curator, which holds the data, in order the latter to issue statistical noisy information for the entire population of users, then solutions which follow this model are named as interactive [22, 60, 62]. In the case of non-interaction between the Curator and the untrusted party, the Curator publishes the noisy statistical information once and goes off-line [43, 44, 64]. Before delving into solutions that adhere to *differential privacy*, we give the background needed for the context that is presented afterwards.

3.2.2.1 Differential Privacy Background

Definition 29. *A statistical database SD is a collection of data points x_i modeled as a multi-set of rows $\mathcal{R}^m = (\{x_{1,j}\}_{j=1}^m, \{x_{2,j}\}_{j=1}^m\{x_{3,j}\}_{j=1}^m \cdots, \{x_{n,j}\}_{j=1}^m)$ each of size m.*

Informally, a statistical database is a data structure in which the values are perceived to represent a sample over a population. The database is held by the Curator which is publishing useful statistics for the underlying population as a whole.

Definition 30. *Two statistical databases SD, SD' are considered as neighboring if*

$$|(SD - SD') \cup (SD' - SD)| = 1$$

Intuitively two statistical databases are neighboring if one database SD can be obtained by

SD', by adding or removing one row.

A Curator is replying to query requests. A query $\mathbf{Q}$ is a function f that takes as input all the rows of a database SD and outputs a single value. The query might be the average of all values of the database based on a predicate: *Average age of patients with HIV*, or the sum value of all users' inputs: *Total consumption of house tenants in a specific geographical area.*

Definition 31. *The L1 sensitivity of f is:*

$$\Delta(f) = max\big\|f(SD) - f(SD')\big\|$$

for all neighboring SD, SD'.

The sensitivity of a function defines a lower bound on how much noise should be added to individual data points such that revealing the value of f does not compromise the confidentiality of individual data points. From another point of view it explains how much an individual data input can change the result of f.

Definition 32. *Given a random variable X, we say that X follows the Laplace distribution if its probability density function $\mathcal{L}(x, \mu, b) = \frac{1}{2b}e^{-\frac{|x-\mu|}{b}}$, where μ, b are the location and the scale parameters, respectively.*

After giving the definitions that will help the reader clearly interpret differential privacy solutions we categorize existing solutions with respect to the amount of trust that should be handled by the Curator.

3.2.2.2 Trusted Curator

In the trusted Curator model the Curator randomizes the database with a randomized algorithm $\mathcal{M}$ that injects some noise to each query response. An Aggregator issues queries to the entire database to learn some statistical function f over SD. The goal of the Curator is to generate $F = \mathcal{M}(f(SD)) \in \mathbf{T}$ such that the knowledge of F does not violate individual privacy and simultaneously the noise added by $\mathcal{M}$ is as small as possible. Hence, an Aggregator which acts as an adversary cannot infer from F the absence or the presence of an individual row in the database. We now proceed into the formal definitions of differential privacy as presented in [56,60]:

Definition 33. *A randomized algorithm $\mathcal{M} : SD \rightarrow \mathbf{T}$ guarantees ϵ-differential privacy if for any neighboring statistical databases SD and SD':*

$$\Pr[\mathcal{M}(SD) \in \mathbf{T}] \leqslant e^{\epsilon} \Pr[\mathcal{M}(SD') \in \mathbf{T}]$$

The parameter of ϵ is publicly defined. *Differential privacy* mechanism in the trusted Curator model, guarantees that an adversary by learning the result of a statistical function $F \approx f$ over a randomized database $\mathcal{M}(SD)$ does not compromise the individual privacy of the participants in the underlying sample population that constitutes SD. In another context *differential privacy* assures that by injecting appropriate noise, the result of $F(\mathcal{M}(SD))$ is not affected with the addition or deletion of a row.

The privacy mechanisms that first addressed differential privacy, relied on the *Laplace noise* [58,60] technique. Briefly, the Curator which holds individual data values $\{x_i\}_{i=1}^{n}$ in a database SD, computes $f(SD)$ and perturbs the result of f with noise sampled from *Laplace* distribution. The scale of the noise is calibrated to the sensitivity of f. The following theorem formalizes the solution:

Theorem 1. *Let $\boldsymbol{Q}$ be a query and $\mathbf{f} = \mathbf{Q}(SD)$ the response results. A randomized algorithm $\mathcal{M}$ preserves ϵ-differential privacy if it adds independent noise sampled from a zero-mean Laplace distribution with scale parameter $b = \frac{\Delta(\mathbf{Q})}{\epsilon}$ at each dimension i of the response transcript $\mathbf{F}[i] = \mathbf{f}[i] + \mathcal{L}(\frac{\Delta(\mathbf{Q})}{\epsilon})$.*

The proof can be found in [60]. The noise added from the Laplace distribution introduces an error at the computation of f which is measured by the standard deviation σ in the difference of the transcripts: $\mathsf{error}_{lap} = \sigma|\mathbf{f} - \mathbf{F}| = \sigma|\mathcal{L}(\frac{\Delta(\mathbf{Q})}{\epsilon})| = \sqrt{2}\frac{\Delta(\mathbf{Q})}{\epsilon}$. When the sensitivity of the query $\Delta(\mathbf{Q})$ is large or the ϵ parameter is small, this yields a more flatten curve of $\mathcal{L}(\frac{\Delta(\mathbf{Q})}{\epsilon})$ with increased noise. The noise mechanism is also independent of the distribution of the values of the database. Solutions in which the Curator is not trusted to perform the computation of f over plaintext data belong to the semi-honest Curator model and are described in what follows.

3.2.2.3 Semi-honest Curator

Ács *et al.* [4] presented a solution for distributive noise computation without any trusted Curator. In this model users independently add noise to their data before transmitting them to the Curator. However, simply adding noise from the *Laplace* distribution adds considerable noise to the final aggregation. In [4] the noise is chosen following the Gamma distribution based on the observation that a *Laplace* random variable can be simulated as a difference of two other independent identical random variables drawn from a Gamma distribution. Predominantly, the staple of the solution is the divisibility property of the Gamma and the Laplace distributions [130]:

Lemma 3. *Let X be a random variable that follows the zero-mean Laplace distribution with probability density function at $x = \mathcal{L}(x, \mu, b) = \frac{1}{2b}e^{-\frac{|x|}{b}}$. Define two random variables G_1 and G_2 that follow the Gamma distribution with probability density function at $x = \mathcal{G}(n, \lambda) = \frac{\frac{1}{\lambda}^{\frac{1}{n}}}{\Gamma(\frac{1}{n})}x^{\frac{1}{n}-1}e^{-\frac{x}{\lambda}}$. Then X can be simulated as $X = \sum_{i=1}^{n}(G_1 \sim \mathcal{G}(n, \lambda) - G_2 \sim \mathcal{G}(n, \lambda))$, for a scale parameter λ and shape $n \geq 1$, where $\sum_{i=1}^{n}(G_1 \sim \mathcal{G}(n, \lambda)) = \mathcal{G}(\frac{1}{\sum_{i=1}^{n}\frac{1}{n}}, \lambda)$.*

Rastogi and Nath [124] presented a different technique with distributive noise in case of a semi-honest Curator. Users first transform its n-dimensional data to $k << n$ dimensions with the *Discrete Fourier Transformation* (DFT). Then they apply noise in the transformed data, that is sampled from 4 Gaussian random variables to simulate a Laplace random variable according to following proposition:

Proposition 1. *Let Z be a random variable drawn from a Laplace distribution $\mathcal{L}(0, b)$ and Y_i for $i \in \{1, 2, 3, 4\}$ be four Gaussian random variables. Then $Z = Y_1^2 + Y_2^2 - Y_3^2 - Y_4^2$ is a random variable following a $\mathcal{L}(\lambda^2/2)$ Laplace distribution.*

As such, the original answer query is compressed and accurately approximated with the k coefficients of the DFT. This results in a small error in the final response without violating differential privacy.

Goryczka *et al.* [79] observed the inefficiency of the aforementioned solutions due to the multiple random variables that have to be generated in order to simulate a variable from a Laplace distribution. Namely, two Gamma random variables are needed in [4] and four Gaussian random variables in [124]. Thus, they proposed a solution for distributive noise in which each user has to generate a random variable from a Laplace distribution combined with a common Beta random variable. The common random variable has to be distributed to all nodes, thus affecting the robustness of the scheme.

The randomization mechanism with noise sampled from a Laplace distribution proportionally calibrated to the query sensitivity is not always a solution [131]. In scenarios where the output of $\mathbf{Q}$ is non-numerical then Laplace noise is useless. In this context, McSherry and Talwar [113] proposed a solution for *differential privacy* in auctions. Differently than with the Laplace mechanism, in which the sensitivity of a function dictates the minimum amount of induced noise, in their model there is a utility function $u(SD, y)$ with respect to the database SD and the output y of the query $\mathbf{y}$. The solution proposes to use noise sampled from exponential distribution, proportional to $exp((u(SD, y)\frac{\epsilon}{\Delta(u)})$, where the sensitivity $\Delta(u)$ is measured with respect to the utility function of two neighboring statistical databases SD, SD'. Tailored solutions for specific scenarios such as streaming data have been also been studied under the *differential privacy* model in the literature [25,34,40,41,57,61,65,96,97,115]. In [133] the authors came up with the neat idea to combine ad-hoc based privacy definitions with differential privacy, after discerning a gap in the latter and the leaked knowledge an adversary obtains after learning statistics over a database, which is not entailed in differential privacy definitions.

Solutions that preserve differential privacy introduce an error in the result of f. That error in specific applications in which precision in the computation of f is of crucial importance, may be unacceptable. Consider the use case of a billing protocol in which users are reluctant to reveal their real power consumption. On the other hand users will never engage in a protocol that will allow an energy supplier to apply a charging policy based on a noisy aggregate result. As such, noise-based techniques are not always suitable for specific **PPDCA** protocols. In the remainder of this Chapter we present customized cryptographic solutions for **PPDCA** protocols in which each user encrypts its individual data value, such that an aggregate function over the plaintext data can be computed from the ciphertexts.

3.3 Cryptographic protocols

In order to tackle accurate and precise data analysis with individual privacy, different solutions that blend cryptographic primitives have been proposed in the literature. Cryptographic **PPDCA** protocols (cf. figure 3.1) involve n users $\{\mathcal{U}_i\}_{i=1}^n$ which hold personal sensitive data x_i. On the other hand an Aggregator $\mathcal{A}$ seeks to infer in cleartext a function f, which takes as input n values from the plaintext space and outputs the result. Users are reluctant to reveal their individual data and as such they encrypt it with an encryption algorithm E. $\mathcal{A}$ holds a secret key sk_A that allows to reveal nothing but F. F takes as input the encrypted data and the secret key sk_A. In a *collection* phase users send their data $\mathsf{E}(x_i)$ to $\mathcal{A}$. After collecting all data $\{\mathsf{E}(x_i)\}_{i=1}^n$, $\mathcal{A}$ proceeds in the *data analysis* phase, in which it learns $F(\{\mathsf{E}(x_i)\}_{i=1}^n) = f$.

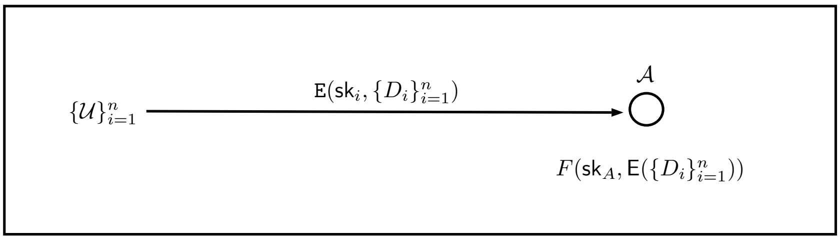


Figure 3.1: Overview of cryptographic techniques for **PPDCA** protocols.

The goal in the end of the protocol is two-fold: a) Users' privacy is not compromised–i.e: An adversary cannot learn anything by observing messages that are exchanged in between users and the Aggregator $\mathcal{A}$ and b) an untrusted Aggregator $\mathcal{A}$ performs *oblivious* computations over $\{\mathsf{E}(x_i)\}_{i=1}^n$ such that it learns $F(\{\mathsf{E}(x_i)\}_{i=1}^n) = f$ and nothing else. In the majority of the current cryptographic solutions the privacy requirements are expressed within a game-based definition. The adversary $\mathcal{A}$ is a *honest-but-curious* party which submits two data sets to a challenger such that when they are given to f as inputs the result is equivalent. The challenger chooses uniformly and at random one of the data set, it encrypts it and it gives the result to $\mathcal{A}$. Following the indistinguishability based security definition of encryption, $\mathcal{A}$ wins the *Aggregator obliviousness* game if it correctly guesses with non-negligible probability which data set has been encrypted by the challenger.

First we target protocols that guarantee individual privacy in the presence of a trusted Aggregator. We then present solutions with a *honest-but-curious* Aggregator. The *honest-but-curious* Aggregator model is further classified in solutions with a fully trusted key dealer, which is responsible to distribute keys to users and to the Aggregator, and in techniques which avoid the use of a fully trusted key dealer.

3.3.1 Trusted Aggregator

Önen *et al.* [121] transformed a symmetric encryption algorithm to an additively homomorphic one in order to allow a tree based aggregation of sensor values. Under this model sensor nodes correspond to the nodes of a tree $\mathcal{T}$ and the sink node is the root of $\mathcal{T}$. The nodes share keys with the parent nodes and the sink. For data value x, secret key k and node i, data is encrypted as $\mathsf{Encrypt}(x, k) = x + f_k(ctr + i)$ for a semantically secure pseudorandom permutation f_k keyed by k.

Castelluccia *et al.* [35] designed a scheme for privacy preserving aggregation for wireless devices. In this scenario a sink-node receives data from all the wireless devices and it computes a function over the encrypted data. Their solution entails a transformation of a symmetric key algorithm to a homomorphic function. Briefly, all nodes and the sink share a secret key k and encrypt their data as follows: $c_i = \mathsf{Enc}(x_i, k) = x_i + k \bmod M$. All wireless sensor nodes are organized in a tree based construction and the root of the tree is a sink node which receives encrypted data from the child nodes. Upon receiving all nodes' ciphertexts the sink node decrypts the sum $\mathsf{Dec} = \sum_{i=1}^{n} c_i - \sum_{i=1}^{n} k = \sum_{i=1}^{n} x_i \bmod M$.

The trusted Aggregator threat model tackles only for privacy against external adversaries. However in a real-world scenario the Aggregator itself is untrusted, since it is curious to learn individual values. In what follows we analyze current **PPDCA** protocols, which assume a *Honest-but-curious* Aggregator.

3.3.2 Honest-but-curious Aggregator

Towards a stronger security model, various protocols strengthen the threat model with a *honest-but-curious* Aggregator which tries to violate individuals' privacy. Solutions that tackle privacy in the presence of *honest-but-curious* Aggregators are further classified with respect to the amount of trust that is put to a key dealer, which is responsible to distribute keys.

3.3.2.1 Trusted key dealer

In [132] the authors proposed a protocol for privacy in aggregation of data by an untrusted Aggregator. They were the first to put forth formal security definition of *Aggregator obliviousness* (AO). According to the *Aggregator obliviousness* (AO) definition, n users contribute with encrypted data and the only allowed leakage in the protocol is the evaluation of a function $f = \sum_{i=1}^{n} x_i$ on the ciphertexts such that: $f(x_1, \ldots, x_n) = F(c_1, \ldots, c_n)$ where x_i corresponds to plaintext values and c_i belongs to the ciphertext space. Each user encrypts $\hat{x}_i = x_i + r_i$ as $c_{i,t} = g^{\hat{x}_{i,t}} H(t)^{\mathsf{sk_i}} \in \mathbb{G}$ for a collision resistant hash function $H : \mathbb{Z} \to \mathbb{G}$, time interval t, secret key $\mathsf{sk_i} \in \mathbb{Z}_p$ and a random generator g for the cyclic group $\mathbb{G}$ of prime order p for which Decisional Diffie-Hellman is hard. The noise r_i is chosen from a geometrical distribution in order to achieve differential privacy on the final result. Secret keys are chosen by a fully trusted key dealer which also transmits to the untrusted Aggregator $\mathsf{sk_0} = -\sum_{\mathsf{i=1}}^{\mathsf{n}} \mathsf{sk_i}$. Aggregator computes the discrete logarithm: $\sum \hat{x}_i = \log_g^{\prod c_{i,t} \cdot H(t)^{\mathsf{sk_0}}}$ and learns the result with the aid of a fully trusted key dealer.

In a similar direction Bilogrevic *et al.* [21] aggregate data in order to compute in a privacy preserving manner second order statistics: an Aggregator monetizes data with respect to the distance from real random data. Due to the adaptation of the same aggregation mechanism as in [132] this solution lends inefficient data analysis due to computation of discrete logarithm for aggregation and weak security model since there is a trusted key deal in the protocol. The distance metric that is used in order to rank the data is the Jensen-Shannon (JS) inequality [105]. Similarly as in [132] users receive secret keys by a trusted dealer and they further randomize data with Geometrical distribution in order to assure differential privacy. Users choose noise

r_i from a geometrical distribution as in [132] and encrypt both $x_i + r_i$ and $x_i^2 + r_i$ in order to compute the mean and the variance that is needed for the JS inequality distance metric.

Benhamouda *et al.* [18] build upon the DDH based scheme of [132], in order to provide a tighter reductionist proof. Moreover, their scheme takes advantage of the small ciphertext size that DDH-based schemes enjoy. The core idea is to base their security on *smooth projective hash functions* (SPHF) with key-homomorphism. SPHF define two hash functions. The one takes as input a secret key hk and a value x form a domain $\mathcal{X}$, and outputs the hash h_1. The second takes as input a public key hp, a value $x \in \mathcal{L}$ and a witness w that $x \in \mathcal{L}$ and outputs the hash h_2. The construction assures $h_1(\mathsf{hk}, x) = h_2(\mathsf{hp}, x, w)$. Similarly as in [132] users encrypt the time series data $x_{i,t}$ in a ciphertext $c_{i,t} = g^{x_{i,t}} \cdot \mathsf{Hash}(\mathsf{sk}_i, H(t))$, where Hash is a SPHF and H is a hash function. The Aggregator decrypts by using its secret key $\mathsf{sk}_A = \sum_{i=1}^{n} \ mathsfsk :$ $\sum x_{i,t} = \log_g^{\prod c_{i,t} \cdot \mathsf{Hash}(\mathsf{sk}_A, H(t))}$.

Joye and Libert [92] tackled the inefficient aggregate approach of [132] due to the computation of a logarithm, with an encryption scheme which is based on Paillier [123] cryptosystem. The neat idea that allows the Aggregator to decrypt and to learn the sum is the following: *Discrete logarithms in the subgroup of* $\{(1+N)^x \bmod N^2\}$*, with* $x \in \mathbb{Z}_N$*, for an RSA composite number* N *can be computed fast by expanding the term* $(1+N)^x \bmod N^2$ *as* $(1+xN)$ *thanks to the binomial theorem.* During the protocol execution a trusted dealer distributes keys $\mathsf{sk_i}$ to each user and a decryption key $\mathsf{sk}_0 = -\sum_{i=1}^{n} \mathsf{sk}_i$ to the untrusted Aggregator. For each time interval t user i encrypts data value $x_{i,t}$ as $c_{i,t} = (1 + x_{i,t}N)H(t)^{\mathsf{sk}_i} \mod N^2$. Finally the untrusted Aggregator decrypts and learns the sum $\sum x_i$ with the use of the secret key sk_0, without computing any discrete log. Despite the significant improvements with respect to computation overhead of the Aggregator, the protocol does not scale well for dynamic leaves and joins and also there is a fully trusted key dealer.

T.-H. Hubert Chan *et al.* [42] targeted a protocol to address failures of users due to communication errors or software/hardware failures (fault tolerance). The key building block in their solution is a binary interval tree based model where each node computes the aggregation of the descendant nodes. The leaf nodes include the data values of the users. The root node

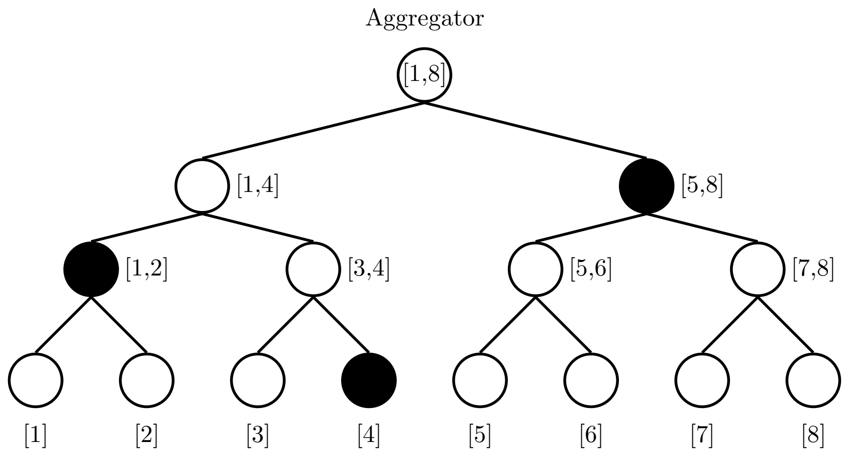


Figure 3.2: The tree based construction by T.-H. Hubert Chan *et al.* [42]. In case of a failure of node 3, Aggregator estimates the sum by summing the black nodes which are the disjoint set that covers the remaining users.

is the untrusted Aggregator. As in [132] users encrypt their plaintext values with the additive homomorphic scheme in [132]—after obfuscating it with random noise from a geometric distribution $\texttt{Geom}(a)$. Thus, due to the tree based construction each leaf belongs to different groups of aggregations. Whenever user k due to faults does not participate in the protocol the Aggregator can still evaluate the sum from the redundant aggregation information imposed at each level of the tree by adjacent disjoint nodes, as can be depicted in figure 3.2. However there is an error to the final result proportional to the size of failure nodes.

The authors in [104] addressed the increased error imposed by the scheme in [42] with another approach. They invented a novel grouping technique of users known as *interleaving grouping* (cf. Figure 3.3). In a basic scheme (cf. figure 3.3) a key dealer chooses n groups of secret key pools $\mathcal{S}_1, \ldots, \mathcal{S}_n$, whereby each pool contains c keys. Then, it chooses randomly a subset $\mathcal{S}'$ of q secret keys and assigns them to the Aggregator. The remaining keys $\mathcal{S}''_i$ are further divided in n groups and each user is assigned with $\mathcal{S}_i = (\bigcup_{i=1}^{n} \mathcal{S}''_i) \bigcup \mathcal{S}'$ keys. For encryption at time interval t each user first applies random Geometric noise r_i to each data value x_i and derives a secret key $k_i = \sum_{s_j \in \mathcal{S}_i} H(\mathsf{f}_{s_j}(t)) - \sum_{s''_j \in \mathcal{S}''_i} H(\mathsf{f}_{s''_j}(t)) \bmod M$, where $M = 2^{\lceil \log_2^{n\Delta} \rceil}, x_i \in \{0, \ldots\ Delta\}, \mathtt{f}$ is a pseudorandom function indexed by $s_j, \mathtt{f} : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$ and H maps λ-bit elements to a random value in $[0, \ldots, M]$. User i then sends the encrypted value $c_i = (x_i + r_i + k_i) \bmod M$ to the

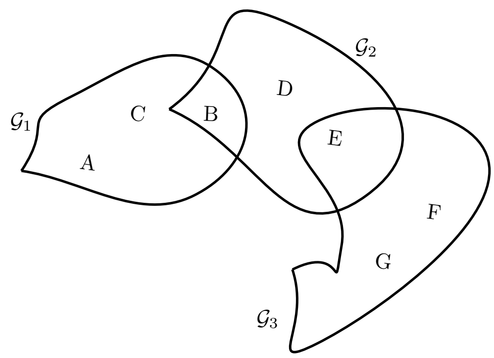


Figure 3.3: Three interleaving groups $\mathcal{G}_1, \mathcal{G}_2$ and $\mathcal{G}_3$. Node B receives keys from groups $\mathcal{G}_1, \mathcal{G}_2$ and node E from groups $\mathcal{G}_2, \mathcal{G}_3$. Aggregator keeps group keys for the groups $\mathcal{S}_1'', \mathcal{S}_2''$ and $\mathcal{S}_3''$. Finally it can only get the sum of all nodes.

Aggregator. For decryption, the Aggregator computes the decryption key $k_0 = \sum_{s'_j \in \mathcal{S}'_i} H(\mathsf{f}_{s'_j}(t))$ for each group and reveals the sum $\sum_{i=1}^{n} x_i + r_i = \sum_{i=1}^{n} (c_i) - k_0 \bmod M$. Dynamic leaves and joins are supported since only a small fraction of users is forced to receive new keys.

Jawurek [87] enhanced existing solutions by restricting the Aggregator to learn f over old data values. This security property is referred as *freshness*. The protocol assures differential privacy and Aggregator obliviousness with the help of a fully trusted key authority $\mathcal{K}$ which sets up the parameters for Pailier public key cryptosystem [123] and remains on-line. Namely, each user chooses randomness r_i to encrypt each data value x_i as $c_i = (1+N)^{x_i} r_i^N \bmod N^2$ for sufficient large prime numbers p and q such as $N = pq$. The untrusted Aggregator multiplies all the ciphertexts $c = \prod_{i=1}^{n} c_i$ transmits it to the key authority which decrypts the result that reveals the sum of the underlying plaintexts. To guarantee freshness, users sign their ciphertext, the encrypted randomness and the time that is coupled with each data value x_i and they forward it to the Aggregator. The Aggregator sends the signatures and the ciphertexts to the key authority. The key authority first checks if the time interval of each query is the latest one and further validates the signatures. If the two steps are successful then it decrypts the computation and sends it to the untrusted Aggregator.

Rastogi *et al.* [124] proposed to blend homomorphic encryption with differential privacy. Users receive by a trusted key dealer secret keys that follow a linear relationship: $r = \sum_{i=1}^{n} r_i$.

During encryption each user encrypts the data value x_i with randomness r_i by encrypting it with Pailler cryptosystem: $c_i = \mathsf{Enc}(x_i + r_i)$. Upon receiving all ciphertexts c_i the Aggregator sends $c = \prod_{i=1}^{n} c_i$ to each user for decryption. Due the distributed decryption process the Aggregator learns $\sum_{i=1}^{n} x_i$ as the exact sum. This bidirectional communication channel is not always possible and also increases the communication cost of the protocol.

Following the distributive noise addition mechanism, along with encryption, Äcs *et al.* [4] proposed to simulate Laplace noise with Gamma variables r_i (lemma 3). Each user chooses independent noise and encrypts the noise along with the data value: $\hat{x}_i = x_i + r_i$. The encryption algorithm that is employed is a additive homomorphic scheme [36] where each data is encrypted as: $\mathsf{Enc}(x) = x + k_i \bmod M$, for $x \in [0, M]$ and $k_i \in [0, M]$. However in order to decrypt the noisy sum the Aggregator has to know $\sum_{i=1}^{n} k_i$, which imposes a trusted key dealer which distributes secret keys to each user.

Chen *et al.* [45] presented a scheme based on a `XOR` based encryption scheme. Each user employs a PRG in order to compute a key r_i that is used to a `XOR` based stream cipher. The randomness r is sent to one trusted party which does not collude with the Aggregator and the ciphertext $c_i = x_i \oplus r_i$ is forwarded to the Aggregator. Finally, the Aggregator receives $\sum_{i=1}^{n} r_i$ and computes $f = \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} c_i \oplus \sum_{i=1}^{n} r_i$. The security of the scheme is based on the non-collusion requirement between the server and the Aggregator and to the security of the PRG. However an eavesdropper which acts as an external adversary can listen to the established channels between users, the trusted third party and the Aggregator, and finally she can learn individual inputs.

Günther *et al.* [82] designed a scheme for privacy preserving aggregations for participatory sensing. In their model (see figure 3.4) a set of mobile sensors are first registered in a registration authority, which plays the role of a fully trusted key dealer. Later on, users encrypt their data values and send them to the service provider. Data is encrypted with an additive homomorphic identity based encryption scheme [28]. An Aggregator submits a query for some users in order to learn their sum and receives the corresponding ciphertexts. The solution guarantees report unlinkability, which assures that an adversary cannot trace ciphertexts submitted by users, query

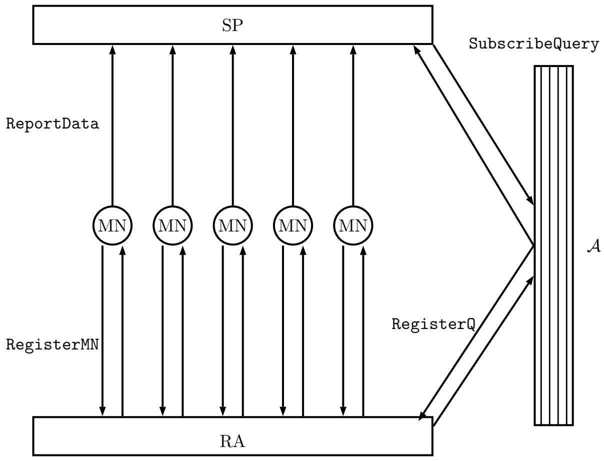


Figure 3.4: Mobile nodes (MN) first register to the registration authority (RA) in order to obtain their secret key in the `RegisterMN` phase. Aggregator ($\mathcal{A}$) also registers to RA to obtain a valid querier registration id during the `RegisterQ` phase. Users then report their readings encrypted with their secret key in the `ReportData` procedure. Finally whenever the querier $\mathcal{A}$ wants to learn the aggregate result for specific ids, it issues a query and it receives back a matching secret tag that allows him to decrypt the result.

privacy which hides the identifiers of users, node privacy which preserves the privacy of the data value submitted by each user and recipient anonymity which hides the receiver of a ciphtertext. However, in the adversarial model the Registration Authority acts as a fully trusted key dealer, which should be online during the protocol execution, and at the decryption phase there is a need to compute discrete logarithms, which restricts the plaintext space range to small values (32-bit numbers).

In contrast with the aforementioned solutions, wherein a fully trusted key dealer distributes keys to the users and to the untrusted Aggregator, such that the Aggregator learns a function f over the plaintext data values, in what follows we analyze solutions in a stronger threat model, whereby the protocol guarantees individual privacy without the assumption of a fully trusted key dealer.

3.3.2.2 No Key Dealer

Erkin *et al.* [63] modified Paillier additively homomorphic encryption scheme [123] in a nifty way in order to avoid distribution of keys by a trusted key dealer. Namely the encryption algorithm proceeds as follows: $\mathsf{Enc}(x) = g^x r^{R(i)}$, where the randomness $R(i)$ of each user i is the amount of randomness sent to all users subtracted by the randomness received by each other user in the network; combined with the public parameter N of Pailler cryptosystem: $R(i) = N + \sum_{i=1,j\neq i}^{n} r_{(i\to j)} - \sum_{i=1,j\neq i}^{n} r_{(j\to i)}$. By this approach when the Aggregator aggregates all the ciphertexts, the randomness is annihilated: $\prod_{i=1}^{n} c_i = g^{\sum_{i=1}^{n} x_i} r^{\sum_{i=1}^{n} R(i)} = g^{\sum_{i=1}^{n} x_i} r^N = E(\sum_{i=1}^{n} x_i)$. Finally, Aggregator decrypts and learns the sum without any further communication with the users with the decryption key of Paillier cryptosystem. However the assumption for a static population of users in which each user has to communicate with all other users in the system increases the communication costs.

Barthe *et al.* [15] proposed a solution which is not based on a trusted key dealer as well. In their model the Aggregator is not a single participant but a coalition of servers. Between the users and the Aggregators there is a service provider. Each user i establishes an ephemeral Diffie-Hellman key $\mathsf{k}_{i,j} \in \mathbb{G}$ with each Aggregator j, where $\mathbb{G}$ is Diffie-Hellman group of prime order q. At each time interval $t \in \mathcal{T}$ the service provider receives from the users blinded readings $r_{i,t} = x_{i,t} + \sum_{j=1}^{m} H(t, \mathsf{k}_{i,j})$ for a hash function $H : (\mathcal{T}, \mathbb{G}) \to \mathbb{Z}_N$, for $N << q$. The service provider forwards to each Aggregator j an index w containing the users that must be included for aggregation and subsequently Aggregator j adds random noise $n(j)$ from a geometric distribution $\mathtt{Geom}(a)$ to the ephemeral key $H(t, \mathsf{k}_{i,j})$. Aggregators finally transmit the noisy ephemeral keys $s = n(j) + H(t,\ mathsfk_{i,j})$ to the service provider which learns the noisy sum by subtracting from the sum of all blinded readings $\sum r_{i,t}$ the noise s. The proposed solution induces high communication costs due to the ephemeral keys that user of the system has to be establish with each Aggregator.

Danezis *et al.* [52] proposed a protocol for non-linear computation over encrypted data. Their solution is based on a secret sharing mechanism between the users and a set of authorities that aim the Aggregator to learn a function over the encrypted data. Users share secret keys with all the authorities. Their solution permits sum, multiplications and boolean function evaluations ($\mathtt{NOT}, \mathtt{AND}, \mathtt{NAND}, \mathtt{OR}, \mathtt{NOR}, \mathtt{XOR}$). The payoff of this approach is an increased overhead with respect to the communication cost of each user since it has to share secrets with all the authorities.

3.3.3 Further related work

PPDCA protocols can be viewed as an instance of Multi Party Computations (MPC). MPC has its foundation in the typical two party computation model, where two mutually distrustful parties without revealing their inputs to each other want to perform a joint computation. Extended for the multi-party environment where multiple clients contribute their secret inputs, theoretical results show that any efficiently computed functionality f can be computed in a MPC setting. In the current MPC scenarios the network is not homogeneous. That is, the parties that participate in the protocol do not share the same computational resources, neither it is assumed for all parties to be able to collude. Moreover there is a set of parties which do not contribute with inputs to the computation of the function and are the only interested in learning the final result. The theoretical foundations of heterogeneous MPC have been presented by Seny Kamara *et al.* [93]. However, the increased communication cost hinders their real world deployment. Moreover, in case of an eavesdropper MPC non-heterogenous protocols require the data producers to encrypt the data values with the public key of the Aggregator. That induces extra computational costs, since the Aggregator has to decrypt each ciphertext received by each user.

The functional encryption [31] paradigm addresses the same goals as the **PPDCA** protocols: Given a function f and an encryption of x the key holder of sk should learn the output of $f(x)$ and nothing else. However, when it comes to a multi-input function [76], then the current constructions fall short of being practical due to their realization with impractical primitives as indistinguishability obfuscation, which in turn is based on inefficient multi-linear maps.

3.4 Taxonomy

In order to achieve *Aggregator obliviousness* existing solutions use a fully trusted key dealer, which distributes keys to the users and to the Aggregator, following a *honest-but-curious* Aggregator model, whereby the Aggregator is trusted to follow the protocol's rules. The shortcoming of the fully trusted key dealer in terms of security is that the key dealer could at any time decrypt individual encrypted values. Moreover there is an extra computational and communication overhead in case users dynamically enter or leave the protocol due to the need of a new key distribution phase. Furthermore, in a ubiquitous environment in which faults are very likely to occur, protocols with a fully trusted key dealer fall short to provide *fault tolerance*. Techniques that avoid the existence of a fully trusted key dealer often require extra rounds of communication either between the users and the Aggregator, such that the later can partially decrypt the result throughout secret-sharing mechanism, or between each user in the protocol in order to agree on a secret encryption key. Solutions with no key dealer, impose an increased communication cost and as such, they are not suitable for *dynamicity* and *fault tolerance*. Moreover in current techniques, there is a restriction on the range of the possible values a user can encrypt in order the Aggregator to efficiently aggregate the data and learn f. We proceed in a detailed taxonomy (cf. table 3.1) according to various properties for the aforementioned cryptographic protocols:

- *Collusion resistance* (CR) assures the privacy of individual data inputs in case of collusions between malicious user and *honest-but-curious* Aggregator.

- *Communication Model* can be *User-User* (UU) which dictates users to share information between each other before sending their data values to the Aggregator, *unidirectional* (UD) in which each user does not receive any information from the Aggregator in order the latter to learn the function f, or *bi-directional* (UB) whereby the Aggregator after receiving all ciphertexts is engaged with an extra round of communication with the users.

- *Dynamicity* refers to the property of a dynamic leave or join of a user without any key re-distribution phase.

- *Fault tolerance* corresponds to a **PPDCA** protocol in which the Aggregator is able to analyze data in presence of users' failures due to communication errors or hardware/software failures.

- *Freshness* is guaranteed when the Aggregator cannot correlate fresh and non-fresh data in order to learn f. Freshness is correlated with the notion of time for time-series data.

- *Aggregator Complexity* refers to the computational cost of the Aggregator for the computation of F.

3.5 Current deficiencies

After the taxonomy made in table 3.1 we are able to highlight the deficiencies of current work that motivates the remainder this dissertation.

- In the analysis that we made we observed that the aggregate function f learnt by the Aggregator is restricted to basic mathematical operations or binary computations over the input data values. Thus, we turn our attention to functionalities beyond these in Chapters 4 and 5.

- We identified a lack in existing protocols to support dynamicity, collusion resistance, fault tolerance and aggregation efficiency, in an *unidirectional* communication model at the same time. This observation led us to design and analyze a protocol suitable for a dynamic population users, that supports collusions, is resilient to users' failures, and provides aggregate efficiency in an *unidirectional* communication model. The protocol is provable secure in the random oracle model and is presented in Chapter 6.

- Moreover, a general observation with respect to the threat model is the fact that all existing solutions but the protocol in [87] assume a *honest-but-curious* Aggregator, which does not deviate from the protocol rules. Even though authors in [87] introduced the novel property of freshness that does not allow a malicious Aggregator to aggregate old data values, the solution requires a trusted party which can decrypt at any time user messages. We incorporate in our model a malicious Aggregator in Chapter 7, that is able to learn the sum over an entire population and constructs a proof that allows anyone to verify the correctness of computations. Our protocol is provably secure in the random oracle model under a new assumption that is analyzed in the generic group model.

3.6 Summary

In this Chapter, we presented the current state of the art for **PPDCA** protocols. We started our analysis with ad-hoc protocols in the literature that add noise to each individual data value.

We then discussed solutions, that are compliant with the differential privacy framework in which noise is tuned appropriately such that an adversary cannot recognize the existence or absence of a specific data value from the final result. However, noise-based techniques introduce an error to the computation of f that may not be acceptable in applications that require precision in the computation of the function f. Next, we surveyed customized cryptographic solutions, categorized with respect to the amount of trust that needs to be placed in third parties. After making a taxonomy of protocols we identified their deficiencies with respect to their weak threat model and to unsupported functionalities. Therefore, in the next chapters we show solutions to the explained shortcomings of current **PPDCA** protocols.

Scheme	CR	CM	Dynamicity	FT	Freshness	AC
Önen [121]	N/A[a]	UD	✗	✗	✗	NDL[b]
Castelluccia *et al.* [35]	N/A	UD	✓	✗	✗	NDL
Shi *et al.* [132]	✓	UD	✗	✗	✗	DL[c]
Bilogrevic *et al.* [21]	✓	UD	✗	✗	✗	DL
Benhamouda *et al.* [18]	✓	UD	✗	✗	✗	DL
Joye *et al.* [92]	✓	UD	✗	✗	✗	NDL
Chan *et al.* [42]	✓	UD,BD	✓	✓	✗	NDL
Li *et al.* [104]	✓	UD	✓	✓	✗	NDL
Jawurek *et al.* [87]	✓	UD	✓	✓	✓	NDL
Rastogi *et al.* [124]	✓	BD	✗	✓	✗	NDL
Äcs *et al.* [4]	✓	UD	✗	✗	✗	NDL
Chen *et al.* [45]	✓	UD	✓	✓	✗	NDL
G̈unther *et al.* [82]	✓	UD	✓	✗	✗	DL
Erkin *et al.* [63]	✗	UU	✗	✗	✗	NDL
Barthe *et al.* [15]	✗	UD	✓	✓	✗	NDL
Danezis *et al.* [52]	✗	UD	✓	✓	✗	NDL

Table 3.1: Taxonomy of **PPDCA** protocols.

[a]Collusion resistance property does not have any mean in case of a trusted Aggregator.
[b]Aggregator learns f without the need of a discrete logarithm computation.
[c]Aggregator has to compute a discrete logarithm in order to learn the result.

Chapter 4

Privacy Preserving Clustering

Contents

4.1 Introduction

Untrusted third parties tend to leverage user information more and more to achieve better content delivery. In particular recommendation systems collect data about users and their interactions with their environment in order to deliver the most appropriate and personalized content. The leveraged information, spanning users' social relations and personal interests, consists of highly sensitive data and hence raises the problem of privacy. A naive solution to the aforementioned problem could be to encrypt data before analyzing them. This would not solve the problem as operations after encryption are not be feasible. A more suitable solution could be to encrypt data homomorphically thus statistical properties on data after encryption can be computed. Even though this solution seems approachable, the current homomorphic encryption schemes fall short of giving a solution for a global analysis system applied to some large scale dataset.

One of the basic building blocks in the vast majority of data analysis scenarios is similarity detection. By analyzing users' dataset, a recommendation engine can discover similar profiles and thus recommend to a newly arrived user some content that was already consumed by other existing "similar users". Online advertisers sought to increase their revenues by inspecting the online behavior of users. That implies an outsourcing of personal sensitive information by online retailers to the advertisers.

The aforementioned applications imply a privacy violation risk. Since the input to the data analysis operations is personal sensitive private information and operations performed over them, individual privacy may be not be protected. As such, users and companies either tend not to submit their data for further analysis to untrusted parties or they give limited access to it due to individual privacy violation risks [86, 107, 112, 127]. Radical solutions include a restriction on the available data analysis operations an Aggregator can perform from the analyzer perspective, which degrades the accuracy on data analysis.

In this Chapter we present a privacy preserving protocol for similarity detection. Cosine similarity can recognize similar vectors based on the formed angle between the vectors. Our privacy preserving mechanism first maps users' data into vectors and then each user individually encrypts its data, such that the geometrical representation of the vectorized data is being preserved. The solution is provably secure under the security of pseudorandom generators. The accuracy of the proposed solution is then evaluated with the study on users' personality characteristics.

4.2 Related Work

Several techniques have been proposed in order to obfuscate data such that when users submit their data to an Aggregator–which seeks to combine all data in order to infer useful statistics over their the entire data–individual data privacy is being protected but specific data mining algorithms can be applied on it. Privacy preserving data mining by adding noise on data has been first proposed in [5, 8]. The solution has been proposed for privacy preserving decision trees as a solution to derive association rules from databases. In [120] the authors proposed geometrical transformation for data clustering. Transformation though, is data dependent and does not scale for multidimensional data.

Data anonymization asks for unlinkability on data records and users. K-anonymity [128,138] has been proposed as a solution to protect the release of data to an untrusted party such that the personal private information for each data record cannot be distinguished from $k-1$ records. Suppression and generalization are two techniques to achieve k-anonymity. By generalization [85] specific attributes are generalized in order to protect user anonymity. With suppression [129] specific data are not released.

In [106] cryptographic tools are used to protect user data privacy when the id3 tree is constructed for association rules. The id3 tree is a widely known technique for data classification. The categorical data of a set of records is being constructed by choosing the attributes that contain the higher information gain. Information gain is expressed as conditional entropy and the problem of id3 construction is approximated by finding the attributes for which the information gain is maximized. The authors assume that data are split horizontally, thus the Aggregator in order to compute the conditional entropy of two users, should separately and privately obtain the data from both. It turns out that information gain for an attribute between two users is expressed as $(u_1 + u_2) \cdot \log(u_1 + u_2)$. The problem has been addressed as a secure multi-party computation of this expression for two users.

Privacy preserving data classification on horizontally partitioned data has been addressed in [48, 94] as well. The solution is based on a privacy preserving protocol for sum computation based on randomization and privacy preserving union set computation. Those two functionalities can securely be used by an untrusted party to infer the global confidence of an attribute in order to infer the association rules that will classify the data. In [119] privacy preserving clustering on vertically partitioned data is addressed by submitting only the similarities on data and not the real data. However, how the users compute the similarities while at the same time their privacy is preserved, is not clearly addressed. Vaidya et al. [72] designed a protocol for secure dot product computation without the use of a trusted party. However the communication cost for computing all the dot products between users is high.

As opposed to previous solutions we propose a scheme that is data independent and assures higher level of privacy. Previous solutions do not scale for multidimensional data [120] and also there is no concrete security analysis with respect to the leakages of the protocol for example. We did not tackle our similarity problem with respect to data anonymization as anonymization protects the metadata and not the actual data. Moreover, data separation techniques in which data are split in between different sites are not always a real world scenario in which each user holds its data in its entire form.

4.3 Problem Statement

4.3.1 Similarity and privacy

We assume a set of n users. Each user $\mathcal{U}_i$ holds its personal sensitive private data $\mathcal{D}_i$. A honest but curious Aggregator $\mathcal{A}$ seeks to group together similar users according to their data. We consider each $\mathcal{D}_i$ as a multidimensional vector of size m: $\mathcal{D}_i = (d_1, d_2, d_3, \cdots, d_m)$. After the data collection, $\mathcal{A}$ is applying some operations F in order to learn the similarity degree f over any pair of data vectors, such that $\mathcal{A}$ can further form clusters. During the detection of similarities in between data the privacy of users should not be compromised. As a consequence, we are looking for an obfuscation mechanism $\phi : \mathbb{R}^m \rightarrow \mathbb{R}^m$ such that for any two vectors $\mathbf{x}, \mathbf{y}$:

$$f(D_i, D_j) = F(\phi(D_i), \phi(Dj))$$

where ϕ will preserve the privacy of individual data and at the same time similarity detection through cosine computation can be computed.

4.3.2 Cosine similarity

Cosine similarity is a widely used distance metric for numerical data. Cosine similarity [111] depicts the geometrical similarity of two objects in an Euclidean space by measuring the angle θ formed by their vector representation in n-dimensional Euclidean space. The dot product $< \mathbf{a} \cdot \mathbf{b} >$ of two vectors $\mathbf{a},\mathbf{b}$ is $< \mathbf{a} \cdot \mathbf{b} >= ||a|| \cdot ||b|| \cos\theta$, where $||a|| = \sqrt{\sum_{i=0}^{n} {a_i}^2}$ is the norm of vector $\mathbf{a}$ and a_i denotes the i^{th} coefficient of this vector. Thus,

$$\cos\theta = \frac{< \mathbf{a} \cdot \mathbf{b} >}{||a|| \cdot ||b||}$$

and the more similar the data the closer the angle between their corresponding vectors is and the closer to 1 their cosine. The cosine similarity is our similarity detection function f.

4.3.3 Correctness and Privacy

Definition 34. *(Privacy Preserving Data Analysis(PPDA)) In a Privacy Preserving Data Analysis scheme a set of n users $\mathcal{U}_i$ are encrypting their data and afterwards the data are sent to the Aggregator $\mathcal{A}$ for analysis. PPDA consists of the following algorithms:*

Setup(1^λ) *It is a randomized that on input of the security parameter 1^λ outputs the secret keys k of the users.*

Encrypt$(\mathsf{sk}, \mathcal{D}_i) \rightarrow \bar{\mathcal{D}}_i$*: It takes as input user data and secret encryption key and it outputs its encryption.*

$\textsf{Analyze}(\bar{\mathcal{D}_i}, \bar{\mathcal{D}_j}) \rightarrow f(\bar{\mathcal{D}_i}, \bar{\mathcal{D}_j})$: *It takes as input two encrypted data vectors and it outputs the result of a data analysis algorithm* $F(\bar{\mathcal{D}_i}, \bar{\mathcal{D}_j})$, *such that* $F(\bar{\mathcal{D}_i}, \bar{\mathcal{D}_j}) = f(D_i, D_j)$, *where* f *is a similarity detection algorithm.*

Definition 35. *(Correctness) A PPDA scheme is correct if for all pairwise combinations of data* $\mathcal{D}_i$, $\mathcal{D}_j$ *the Aggregator* $\mathcal{A}$ *executes* $\textsf{Analyze}(\textsf{Encrypt}(\mathsf{sk}, \mathcal{D}_i))$ *and obtains* $F(\bar{\mathcal{D}_i}, \bar{\mathcal{D}_j}) = f(\mathcal{D}_i, \mathcal{D}_j)$.

Intuitively, the privacy guarantee we require from a PPDA scheme is that given encrypted vectors $\bar{\mathcal{D}_i}$ an adversary cannot learn any information about the plaintext $\mathcal{D}_i$ although it may learn the output of the function f. For the threat model we assume external adversaries, which do not know the common θ angle. We formalize its security through a game between an Adversary $\mathcal{A}$ and the oracles $\mathcal{O}^{\mathsf{PPDA}}_{\mathsf{Encrypt}}, \mathcal{O}^{\mathsf{PPDA}}_{\mathsf{C}}$ as follows:

At the **Learning** phase $\mathcal{A}$ submits vectors D to the $\mathcal{O}^{\mathsf{PPDA}}_{\mathsf{Encrypt}}$ oracle and the latter returns the encryption of D with secret key sk.

During the **Challenge** phase:

- $\mathcal{A}$ submits one pair of bi-vectors $V_0 = (D_0, D_1), V_1 = (D_2, D_3)$ to $\mathcal{O}^{\mathsf{PPDA}}_{\mathsf{Encrypt}}$, that have not been asked during the learning phase.
- $\mathcal{O}^{\mathsf{PPDA}}_{\mathsf{Encrypt}}$ selects uniformly at random a bit b and returns to A $\bar{\mathcal{D}_i} = \mathsf{Encrypt}(\mathsf{sk}, D_b)$.
- A returns b'.
- if $b' = b$ then $\mathcal{A}$ wins the game.

Definition 36. *A PPDA scheme is secure if for any adversaries* $\mathcal{A}$*, the probability of correctly guessing* b *is:*

$$\Pr[\mathcal{A}^{PPDA}] \leq \frac{1}{2} + \epsilon(\lambda)$$

, for a negligible function ϵ *on input of the security parameter* λ.

4.4 Solution

4.4.1 Idea of Solution

The idea of the solution is to apply some transformations to original vectors which on the one hand preserve the angle between any pair of them and on the other hand assure privacy. Since rotation in a two dimensional space preserves angles, we apply this transformation to two-dimension vectors named as sub-vectors which originate from the original data vector. Additionally, these sub-vectors are further randomly scaled and thus obfuscated while still not having an impact on the angle.

We observed security leakages when the encryption mechanism does not entail both random scalings and rotations. If each user only selects random scaling as the encryption mechanism then an adversary by obtaining a good guess for a coefficient of a user's vector it can recover the specific two dimensional vector by computing the inverse of the guessed element and multiplying it by the encrypted coefficient.

On the other hand, thanks to rotations, the aforementioned problem is mitigated but the following one appears when random vector rotations are used: if two users with secret vectors $\mathcal{D}_i$, $\mathcal{D}_j$ respectively have the same value at the same position of their vectors then only by encrypting with a rotation matrix $\mathcal{R}_\theta$ of angle θ, the corresponding encrypted vectors would have the same value at this position. This violates the security definition 36. Thus, in order for the cosine similarity to be securely preserved after the encryption of the vectors, both random scaling and rotations are applied. Hence, thanks to the rotation, the adversary cannot discover similarities between one vector's coordinates. The mapping of vectors into subvectors also decreases the probability of discovering the original vector since the scaling factor differs from subvector to subvector.

4.4.2 Preliminaries

Vector scaling

Vector scaling with a scaling factor s is defined by a multiplication operation between the vector v and the identity matrix S in which the main diagonal has been substituted with the scale factor s.

$$v \cdot S = v \cdot \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix}$$

Vector Rotation Vector rotation with an angle θ is defined by a matrix multiplication between the vector v and the rotation matrix $\mathcal{R}_\theta$: $v \cdot R = v \cdot \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$

4.4.3 Protocol description

We now describe the details of the protocol with respect to Definition 34.

- $\textsf{\textbf{Encrypt}}(\mathsf{sk}, D_i)$ During the encryption phase each user $\mathcal{U}_i$ holds a vector $\mathcal{D}_i = (d_1, d_2, d_3, \ldots, d_m)$ of size m. It generates subvectors of 2 dimensions $d_i^{(k,l)} = \begin{pmatrix} d_k \\ d_l \end{pmatrix}$. If m is odd then $(m+1)/2$ are constructed, otherwise if m is even then we have $m/2$ subvectors. In general we have $\lceil m/2 \rceil$ subvectors. Afterwards each user chooses a random scaling factor for each subvector and it scales each subvector $d_i^{(k,l)}$ with the random scaling factor s_i: $S_i^j = s_i^j \cdot d_i^{(k,l)}$, obtained with a pseudorandom generator $PRG_1(z_1)$, that takes as input a random looking seed z_1. That is, if any of the coefficients of the subvector $d_i^{(k,l)}$ have been previously selected to form a vector then the old random scale factor s_i must be used for $d_i^{(k,l)}$. Then the intermediate vector $\mathcal{S}_i$ is further rotated with a rotation matrix $\mathcal{R}_\theta$, where θ is the rotation angle: $\bar{d}_i^{k,l} = S_i^j \cdot \mathcal{R}_\theta = S_i^j \cdot \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$. For the computation of θ a pseudorandom generator is also used to generate the common to all users angle $\theta = PRG_2(z_2)$, where z_2 is a random looking seed.

  Finally each user $\mathcal{U}_i$ sends $\bar{\mathcal{D}}_i = (\bar{d}_i^{(1,2)}, \bar{d}_i^{(2,3)}, \cdots, \bar{d}_i^{(k,l)}), \forall k, l \in [0, \cdots, m]$ to the Aggregator $\mathcal{A}$. Hereafter we will write d_i^j to denote the j^{th} subvector of user $\mathcal{U}_i$ and $\bar{d}_i^j$ for the j^{th} encrypted subvector of user $\mathcal{U}_i$. As such, the encryption mechanism consists of random scalings and rotations by an angle θ: $\textsf{\textbf{Encrypt}}(\mathsf{sk}, D_i) = s_i^j \cdot d_i^{k,l} \cdot \mathcal{R}_\theta$, where $\mathsf{sk} = (s_i^j, \mathcal{R}_\theta)$.

- $\textsf{\textbf{Analyze}}(\bar{\mathcal{D}}_i, \bar{\mathcal{D}}_j)$ The analyzer then performs computation F over the encrypted data in order to learn the similarity f of two data vectors: $\forall \mathcal{U}_i, \mathcal{U}_j, i \neq j$:

$$F(\bar{d_i}, \bar{d_j}) = \begin{cases} \cos(\bar{d}_i^{1,2}, \bar{d}_j^{1,2}) \\ \vdots \\ \cos(\bar{d}_i^{\lceil m/2 \rceil}, \bar{d}_j^{\lceil m/2 \rceil}) \end{cases}$$

Aggregator computes the similarity between two vectors on the encrypted data, by applying the cosine similarity algorithm on the encrypted data. As such, $F = f$ and $\mathcal{A}$ learns the cosine similarity between vectors of data.

4.4.4 Correctness

Theorem 2. *The PPDA scheme presented above is correct.*

Proof. It is known that $\cos(a,b) = \frac{<a \cdot b>}{\|a\| \cdot \|b\|} = \frac{a^T \cdot b}{\|a\| \cdot \|b\|}$. For the proof of the theorem we need to prove the following three lemmas:

Lemma 4. *The transpose of an orthogonal matrix A, A^T is equal to its inverse A^{-1}*

Proof. It is known that:

$$A \cdot A^{-1} = I_A \tag{4.1}$$

where I_A it's the identity matrix of A. Also we obtain:

$$A \cdot A^T = \begin{bmatrix} A_{1,1}^T \cdot A_{1,1} & \cdots & A_{1,m}^T \cdot A_{1,m} \\ A_{n,1}^T \cdot A_{n,1} & \cdots & A_{n,m}^T \cdot A_{n,m} \end{bmatrix} = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & 1 \end{bmatrix} = I_A$$

From (1), (2) we have that for any orthogonal matrix A, $A^T = A^{-1}$ ☐

Lemma 5. *The multiplication two vectors a, b with a rotation matrix $\mathcal{R}$ preserves its cosine similarity.*

Proof. $\cos(Ra, Rb) = \frac{<Ra \cdot Rb>}{\|Ra\| \cdot \|Rb\|} = \frac{(Ra)^T \cdot Rb}{\|Ra\| \cdot \|Rb\|} = \frac{a^T R^T \cdot Rb}{\|a\| \cdot \|b\|} = \frac{a^T R^{-1} \cdot Rb}{\|Ra\| \cdot \|Rb\|} = \frac{a^T \cdot b}{\|a\| \cdot \|b\|} = \cos(a, b)$ where $\|Ra\| = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \|a\|$ and $\|Rb\| = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \|b\|$ □

Lemma 6. *The random scaling of two vectors a, b with different random scaling factors r_1 and r_2 preserves its cosine similarity.*

Proof. $\cos(r_1 a, r_2 b) = \frac{<r_1 a \cdot r_2 b>}{\|r_1 a\| \cdot \|r_2 b\|} = \frac{(r_1 a)^T \cdot r_2 b}{r_1 \|a\| \cdot r_2 \|b\|} = \frac{r_1 a^T \cdot r_2 b}{r_1 \|a\| \cdot r_2 \|b\|} = \frac{a^T \cdot b}{\|a\| \cdot \|b\|} = \cos(a, b)$ □

From lemma 4, 5 and 6 it is true that multiplication of a random vector and random scaling is a correct *Privacy Preserving and Data Analysis* mechanism, since encryption preserves cosine similarity: $F = f$. The proof of lemma 5 is based on lemma 4: the rotation matrix $\mathcal{R}$ is orthogonal and as such $\mathcal{R}^{-1} = \mathcal{R}^T$. Furthermore the rotation doesn't change the vector norms.

□

4.5 Security

For the privacy analysis we will show how to correlate the success probabilities of an adversary, which will try to distinguish encryptions with a one-time pad, which uses a PRG to generate a pseudorandom bit-stream that is used as a key, from encryptions with a one-time pad that chooses uniformly at random keys, with the success probabilities of an adversary $\mathcal{A}$ that tries to break our scheme. $\mathcal{A}$ depicts external adversaries, which are not aware of any source of randomness: neither the random scaling factor nor the common random angle θ.

Theorem 3. *The PPDA scheme presented above is secure according to definition 36 if $PRG_{1,2}$ are secure pseudorandom generators.*

Proof. We show how to relate the success probabilities of an adversary $\mathcal{B}$ who tries to distinguish uniformly random strings from PRG outputs. We name the distinguishing probability of adversary $\mathcal{B}$ against the PRG $\mathbf{Adv}_{\mathcal{B}}^{PRG}$ which is negligible function ϵ. When adversary $\mathcal{A}$ of our scheme sends two vectors V_0, V_1 then when $\mathcal{B}$ receives them it flips a random coin and if $b = 0$ it returns the encryption of $V_0, \mathsf{Encrypt}(\mathsf{sk}, V_0)$ as in the described

PPDA scheme, using pseudorandom generators PRG_1, PRG_2 to obtain the key sk, otherwise when $b = 1$ it chooses sk uniformly at random and encrypts with the $\mathsf{Encrypt}$ algorithm V_1. When $b = 0$ the view of $\mathcal{A}$ is exactly as in the real experiment and succeeds with probability $\mathbf{Adv}_{\mathcal{A}}^{PPDA}$. When $b = 1$ then $\mathcal{A}$ succeeds with probability 1/2: $\mathbf{Adv}_{\mathcal{A}}^{\mathsf{sk}\leftarrow R} = 1/2$. By definition $\mathbf{Adv}_{\mathcal{A}}^{PRG} = |\mathbf{Adv}_{\mathcal{A}}^{PPDA} - 1/2| \leq \epsilon \implies \mathbf{Adv}_{\mathcal{A}}^{PPDA} \leq 1/2 + \epsilon$.

□

4.6 Evaluation

We also demonstrate the correctness of our protocol with an experimental evaluation procedure. We obtained data from a personality experiment. We first cluster the data based on cosine similarity, using hierarchical clustering. The same clustering algorithm is further applied over the encryption of the same data using ϕ which as already described combines rotation and random scaling. We proceed with an analysis of the data and next with the clustering algorithms that we use.

4.6.1 Data Set

The dataset contains results from the Foursquare Personality Experiment[1] which uses the mobile social network Foursquare[2], combined with a standard personality test to link between personality (as defined by the five-factor model [73]) and the places that people visited. To the best of our knowledge, this is the first time that it has been possible to correlate personality with place on such a granular level.

When accessing the experiment, users sign in using their Foursquare account, allowing us to access the list of venues which they have 'checked in' to on the Foursquare service. We access only this list, storing the venues that the user has been to and the number of times they have visited each venue, but without accessing or storing the information about the individual checkins - we do not store *when* each visit to the venue occurred, nor the order in which venues were visited. Once users have accessed the system they then take a 44-item personality test [88,

[1] `http://www.cs.cf.ac.uk/recognition/foursqexp`
[2] `http://www.foursquare.com`

89], revealing their five-factor personality scores. The five-factor model gives each person a score between 1 and 5 for each of the five personality traits: Openness, Conscientiousness, Extraversion, Agreeableness and Neuroticism. The users, who participated in the study were a self-selecting group comprised of 173 people who both use Foursquare online location based tagging system and are willing to take part in a personality-based experiment.

4.6.2 Clustering

Clustering algorithms seek to group similar objects together. Similarity is measured with a distance metric which in our case is cosine similarity. Hierarchical clustering is a widely known approach for clustering. It constructs a binary tree of clustering objects that successively are merged under the same cluster with respect to the linkage metric. The linkage metric links clusters and objects together. It acts as an intergroup similarity measure. Two most popular linkage metrics are the *complete* metric which defines the maximum similarity between two objects as a verification to whether or not one object would be merged under the same cluster with another one and the *single* metric in which the minimum similarity is treated as the intergroup similarity metric. At the first step of the algorithm each object belongs to each own cluster. Then all the possible pairwise similarities between objects with respect to the defined distance metric are defined. Afterwards the algorithm iteratively merge clusters with respect to the linkage metric until there would be one cluster with the all the examined objects.

4.6.3 Results

We applied the hierarchical algorithm over the personality dataset with the complete linkage metric and based on cosine similarity. The data consists of 173 different 5 dimensional vectors describing users' personality with respect to the 5 personality traits as previously described. We did not include venue visits frequency since we believe that personality traits are considered much more sensitive data compared to location information and that users would be more interested in hiding such information. We consider similarity on 3 subvectors per user data: the subvectors are constructed with the $(1^{st}, 2^{nd}), (3^{rd}, 4^{th})$ and $(1^{st}, 5^{th})$ coordinates of the original vector respectively. Any pairwise subvector could behave chosen such that the union of the set of subvectors entails all the coefficients. The main similarity metric is computed as the average of the similarities between subvectors. In order to protect their privacy, every user chooses a random scaling factor per two dimensions. After the random scaling process users apply the rotation operation to their partially obfuscated subvectors.

In figure 4.1 we plot the two dendograms of hierarchical clustering before and after the encryption of data with our algorithm. The horizontal axis of the plot corresponds to cluster indexes that are formed by the algorithm and the vertical axis to the linkage similarity based on cosines. Clusters are connected with upside-down U-shaped lines. The clusters are exactly the same due to the correctness of the algorithm as has been previously proved. All the cosines between all the coefficients of 2 over 173 elements has been computed. That results into a set of 14878 distances. For the linkage function we chose the *complete* option. Thus, two clusters will be merged together according to the maximum distance between their elements. Results shown in figure 4.1 demonstrate the correctness of our protocol.

4.6.4 Discussion

In our experiments we used as a similarity metric an aggregate output of each three per user similarities. This is the average of cosine similarities. Thus, during the clustering the similarity between points depicts similarities between the averages. We could have demonstrated three different scenarios during the clustering process one for each subvector in order to check the correctness of our obfuscation mechanism but since this has been demonstrated once the other experiments would not add extra knowledge. We also want to state that the aggregate function should not always be used for every case. This would imply an inconsistency on correctness since many inputs could evaluate the same average similarity. Suppose for instance that data consist of user interests on m items and for each user n similarities per two dimensions are computed. Then a single aggregate function on user n similarities might group together during clustering dissimilar objects that average the same similarities but on different inputs.

4.7 Summary

The interplay between data analysis and privacy is emerging rapidly. Researchers from machine learning area have highlighted the merit of data analysis operations. However this exposure of personal sensitive data, facilitates privacy violations. Adversaries by gaining access to personal information can learn the real identity of users and overcome data legal regulations and restrictions. That postulates a mechanism that would shield individual data confidentiality.

In this chapter we presented a mechanism for privacy preserving clustering that is based on geometrical transformation of objects. Data are encrypted appropriately such that operations with respect to cosine similarity detection are compatible. We proceed into an analysis of the security risks of each operation and we concluded that the most secure way is a combination of random scalings and rotations. Without scaling and only with rotation, similarities on the same position coordinates are possible to occur by external adversaries. This is mitigated by a random scaling factor, which is different per user and per subvectors with no common coefficients. We proceed into an experimental evaluation of a scheme in order to demonstrate its correctness. Personality traits have been obtained by 173 users and identical clustering results have been observed before and after the obfuscation proposed solution.

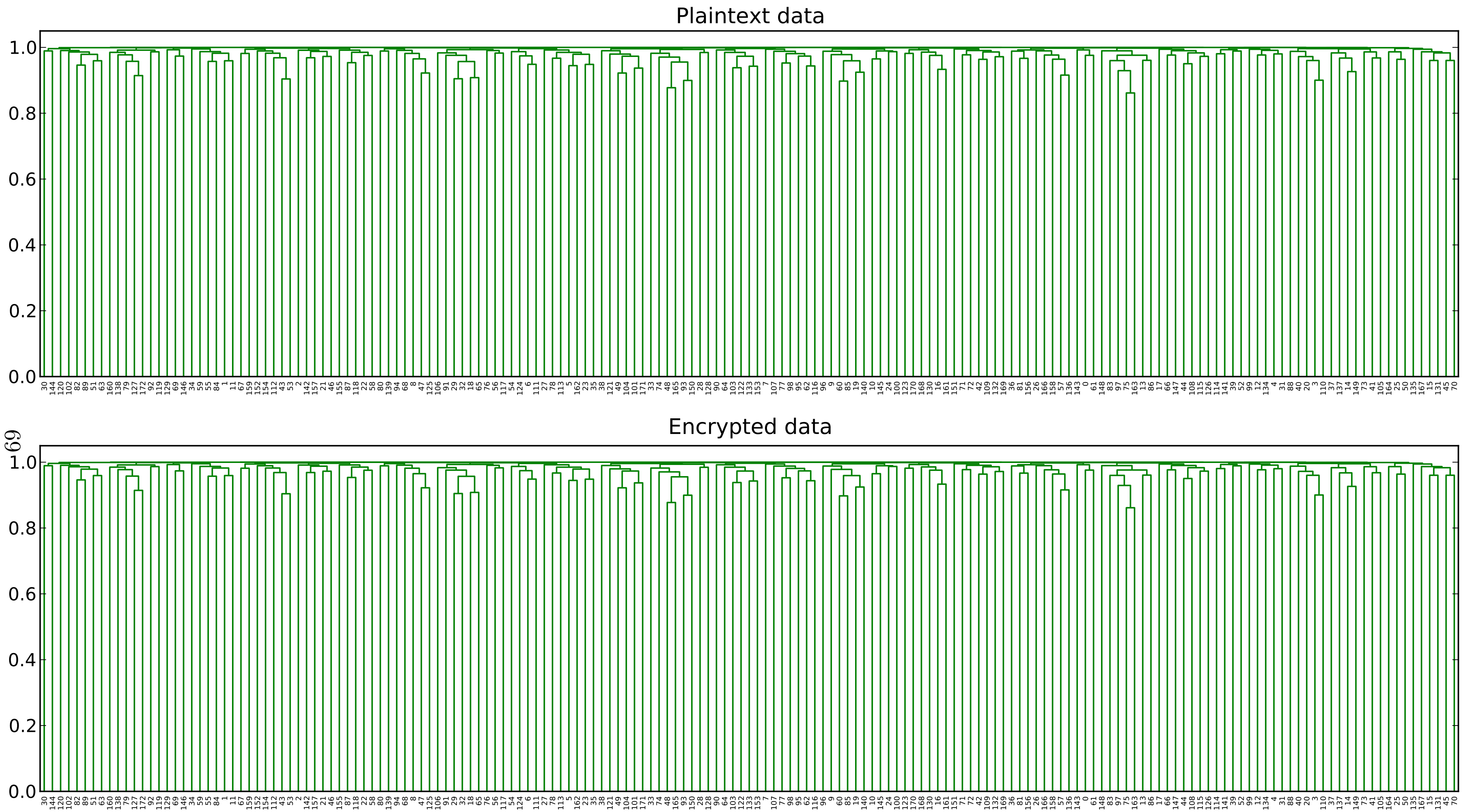


Figure 4.1: Hierarchical Clustering

Chapter 5

Privacy Preserving Statistics in the Smart Grid

Contents

5.1 Introduction

Smart meters are devices deployed in households to measure the energy consumption in specific time intervals. They do not only measure electricity consumption but gas and water commodity as well. The motivation for the wide deployment of smart meters is many-fold. Suppliers can more precisely learn the time intervals houses consume more energy and thus tune appropriately the billing of each customer and predict the potential energy demand. On the other hand, home tenants can receive energy advices and can also change their energy consumption habits. In particular, a customer learning the period of the highest consumption may prefer to consume in a more efficient way.

In this Chapter, we consider the problem of computing continuous maximum energy consumption over meterings sent by individual smart meters in a privacy preserving manner. Following the analysis that we made in Chapter 3, such type of statistics do not exist in the current literature. We assume that both the supplier and individual smart meters are interested in determining the interval in which the smart meter consumes the most. Such an operation cannot be performed by a smart meter alone because of its lack of resources and in particular its lack of memory: The smart meter would need an important number of values in order to find out the maximum value corresponding to a "continuous" consumption. On the other hand, outsourcing these computations to the supplier will naturally leak periodical consumptions which definitely are very sensitive information. We therefore propose a solution, in which smart meters send their periodical metering to the supplier in a privacy preserving manner while still allowing this entity to compute the time interval of the maximum consumption. The proposed solution is based on an order preserving encryption (OPE) which by definition preserves the order of plaintext values after their encryption without revealing any additional information. Additionally, in order to filter out spontaneous peaks (due to some erroneous switch-on/switch-offs of home devices for instance), the smart meter also sends the differences of consecutive consumption values in an *on-the-fly* approach whereby the smart meter does not need to store auxiliary information. Thanks to the differences the supplier is able to determine the period of maximum consumption that is continuous. The proposed solution is provably secure with a reductionist proof to the

POPF-CPA assumption [23] which corresponds to the security notion that characterizes the security of OPE.

5.2 Problem Definition

In this section we precisely define the problem we are trying to address and the environment in which we envision our protocol to run. We seek for *Privacy Preserving Smart Grid Statistics* (**PPSGS**) scheme for a set of smart meters. The smart meters are sending their meterings to a supplier and the supplier should identify the time interval at which each smart meter reports the maximum consumption. The supplier learns nothing but the time period of the maximum consumption.

5.2.1 Entities

1. **Smart meters**. We assume a set of n smart meters, each one denoted as sm_i. These are deployed in separate households across a geographical region. The smart meters are universally programmed to send energy consumption at a fixed time interval t_i starting from time t_1 and ending at time t_e. Each smart meter has an embedded private key in a tamper resistant hardware module.

2. **Supplier**. An energy supplier collects information from each smart meter and computes the time interval corresponding to the maximum consumption individually for each smart meter, thus acting as an Aggregator $\mathcal{A}$.

Table 1 describes the notations used throughout the Chapter.

5.2.2 Protocol Definitions

Definition 37. *(Privacy Preserving Smart Grid Statistics)(PPSGS) A PPSGS scheme consists of the following algorithms:*

$\mathsf{\textbf{Setup}}(1^\lambda)$ *It is a randomized lagorithm that on input of the security parameter* 1^λ *outputs for each user a secret key* $\mathsf{sk_i}$ *and mac key* mk_i.

Encrypt$(x_{i,t}, \mathsf{sk_i}, \mathsf{mk}_i) \rightarrow (c_{i,t}, d_{i,t}, \mathsf{mac}_{i,t})$ Each smart meter sm_i encrypts its meterings $x_{i,t}$ for time interval t using its secret encryption key $\mathsf{sk_i}$. It also computes the discretized differences of consecutive meterings $d_{i,t}$. The output of the algorithm is the ciphertext value $c_{i,t}$, the discretized differences d_i and an integrity value $\mathsf{mac}_{i,t}$ computed with a MAC key mk_i.

Analyze$(\{c_{i,t}\}, \{d_{i,t}\}, \mathsf{mac}_{i,t}, \mathsf{mk}_i) \rightarrow t_j$ *The supplier takes as input encrypted meterings* $\{c_{i,t}\}$*, differences* $\{d_{i,t}\}$*, MACs* $\mathsf{mac}_{i,t}$ *and the MAC key* mk_i *and it outputs a tag* t_j *for each meter* sm_i *that specifies an interval of the maximum consumption.*

Definition 38. *(Correctness) A PPSGS scheme is correct if for all individual smart meters* sm_i *that submit their meterings to a supplier, after running* Analyze$(\{c_{i,t}\}, \{d_{i,t}\})$ *algorithm, the supplier outputs the maximum consumption of* sm_i *with probability 1.*

Notations	
$\mathsf{sk_i}$	secret encryption key of user i
$\mathsf{mk_i}$	mac of user i
sm_i	Smart meter i
t_j	Time interval t_j
$x_{i,t}$	Energy consumption of smart meter i at time interval t
$c_{i,t}$	Encrypted energy consumption of smart meter i at time interval t
miw	Maximum interval window defined by the supplier
$d_{i,t}$	Difference of $x_{i,t}$ - $x_{i,(t-1)}$ metering values

Table 5.1: Protocol notations

5.2.3 Privacy Definition

We consider a *honest-but-curious* adversary model: Although following the steps of the protocol correctly, the supplier will try to discover the content of the meterings sent by each smart meter. Message forgery attacks are prevented thanks to the use of existentially unforgeable message authentication codes (MACs). We namely present our privacy requirement:

Third party obliviousness(TPO). We adapt the security notions of aggregate obliviousness in [132] to define our privacy requirements: The third party, which in our environment is the supplier, cannot learn anything more than the time interval of maximum energy consumption.

Consider an energy supplier that receives the encryptions of each smart meter sx_i. The supplier can only learn the time interval that corresponds to the maximum consumption of each sx_i and not the metering value in plaintext.

We formulate the third party obliviousness privacy definition with a game $\mathbf{Game}^{TPO}$, which is played between the challenger $\mathcal{C}$ and a probabilistic polynomial time (PPT) adversary $\mathcal{A}^{TPO}$ We assume a data structure $\mathbf{T}_i$ which depicts time series data indexed by i. $\mathbf{T}$ can be seen as an $i \times j$ matrix, in which each row $\mathbf{T}_i$ corresponds to a different time serie for time intervals j. $\mathcal{A}^{TPO}$ hass access to the game's oracles in the following phases:

Learning. During the learning phase $\mathcal{A}^{TPO}$ can issue two type of queries:

- **Type I**: $\mathcal{A}^{TPO}$ submits encryption queries $(x, \mathbf{T}_{i,j})$ to $\mathcal{O}^{\mathsf{TPO}}_{\mathsf{Encrypt}}$ oracle and $\mathcal{O}^{\mathsf{TPO}}_{\mathsf{Encrypt}}$ returns to $\mathcal{A}^{TPO}$ $c_{i,j}$, which corresponds to an order preserving ciphertext $c_{i,j}$ for time interval $\mathbf{T}_{i,j}$.

- **Type II**: $\mathcal{A}^{TPO}$ issues queries $(\mathbf{T}_{i,k}, \mathbf{T}_{i,l})$ to the $\mathcal{O}^{\mathsf{TPO}}_{\mathsf{Diff}}$ oracle and the latter replies with the corresponding difference $x_{i,k} - x_{i,l} \iff$ i) an encryption query for a message x for the time serie $\mathbf{T}_i$ has not been submitted for a **Type I** query or ii) a message x was part of an encryption query to the $\mathcal{O}^{\mathsf{TPO}}_{\mathsf{Encrypt}}$ oracle for $\mathbf{T}_{i,j} > \mathbf{T}_{i,k}, \mathbf{T}_{i,l}$.

Challenge. $\mathcal{A}^{TPO}$ submits two differences of plaintext values $d_0 = x_1 - x_0, d_1 = x_3 - x_2$ to $\mathcal{O}^{\mathsf{TPO}}_{\mathsf{C}}$ oracle which correspond to messages for time intervals $(\mathbf{T}_{i,k^0},\ \mathbf{T}_{i,l^0})$, $(\mathbf{T}_{i,k^1},\ \mathbf{T}_{i,l^1})$ respectively. The latter choses uniformly and at random $b \xleftarrow{\$} \{0, 1\}$ and returns to $\mathcal{A}^{TPO}$ the encryptions of one pair corresponding to either the encryptions of (x_1, x_0) if $b = 0$ or the encryptions of (x_3, x_2) if $b = 1$.

Guess: At the end of the game the adversary outputs its guess b'.

We say that $\mathcal{A}$ wins the Third party obliviousness game if its guess $b' = b$ and none of the $(\mathbf{T}_{i,k^0},\ \mathbf{T}_{i,l^0})$, $(\mathbf{T}_{i,k^1},\ \mathbf{T}_{i,l^1})$ have been queried at the learning phase to either $\mathcal{O}^{\mathsf{TPO}}_{\mathsf{Encrypt}}$ or $\mathcal{O}^{\mathsf{TPO}}_{\mathsf{Diff}}$ oracle.

Definition 39. *(Third party obliviousness). Let* $\Upsilon = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Analyze})$ *be a PPSGS scheme with associated plaintext size* $\mathcal{M}$ *and ciphertext size* $\mathcal{N}$*.* Υ *ensures third party obliviousness if for all PPT adversaries* $\mathcal{A}$ *the probability of winning the aforementioned game is* negligible*:* $\Pr[b' = b] \leq \frac{1}{2} + \epsilon(\lambda)$*, where* $\epsilon(\lambda)$ *is a negligible function and* λ *is the security parameter.*

5.3 Overview of PPSGS

In this section we give a brief description of our solution. Our PPSGS scheme achieves third-party obliviousness thanks to an order preserving encryption scheme in which the order of numerical items in the plaintext space is preserved in the ciphertext space as well. Each smart meter is equipped with a tamper resistant hardware module in which a secret key is embedded. This secret key is being used to encrypt meterings at each time interval. Thanks to the cryptographic primitive of order preserving functions a keyed order preserving functions chosen uniformly and at random is indistinguishable from an ideal one. Thus nothing more than the order is revealed to the supplier who is acting as a data analysis entity.

For the accuracy of the analysis once the supplier has identified the time interval in which a smart meter has consumed the maximum it can verify from the extra information composed by the differences between each consumption, that actually there is a valid continuous maximum energy consumption "around" this time interval. If the differences converge to 0 then it has a strong indication that the meterings around that particular interval are part of a continuous maximum consumption. Albeit the goal of publishing differences is to allow energy suppliers determine continuous maximum energy consumptions, researchers have raised the interest for the design of privacy preserving protocols for spike detections so as to energy operators identify overloaded power lines [53]. As such our solution is suitable for this case as well. The advantage of our approach is that the smart meters do not have to store the differences or the ciphertexts in order to perform the analysis but these are computed and sent immediately *on-the-fly*. From the supplier perspective the verification of a maximum continuous consumption interval is performed in a batch way with a single operation. Moreover as it will be established in section 5.2.3, the differences do not jeopardize the privacy requirements of the scheme.

The information from the identification of a continuous energy consumption will improve the forecasts of energy consumption and will allow better energy allocation in advance from energy producers. Moreover, the information of the maximum energy consumption interval can be sent back to the tenants in order to swift their increased energy habits into low tariff periods. This operation cannot be performed locally at each smart meter because their resources are not sufficient for big data analysis operations. On the other hand, an integrity mechanism is needed in order for the supplier to be assured that the meterings are sent from existing and authenticated smart meters.

5.4 Protocol Description

In this section, we formally define our **PPSGS** protocol. Before describing our protocol in full details we give a brief description of what an order preserving encryption scheme is.

5.4.1 Order preserving encryption (OPE)

Privacy preserving queries on databases have raised the interest for non conventional symmetric encryptions [6]. Recently, in [23], Boldyreva et. al. formally defined an Order Preserving Encryption (OPE) scheme. An OPE leaks the order of plaintext data and ideally nothing more. An order preserving function (OPF) is a function g such that for $a < b$ then $g(a) < g(b)$. A symmetric encryption scheme is then an order preserving encryption scheme if the encryption function Enc is an order preserving function. The construction is based on the observation that an OPF with domain D of size M and range R of size N is a bijection of all combinations of M out of N. The security of an OPE has been analyzed in [24] with strict security definitions and bounds. The authors described how an "ideal" random order preserving function (ROPF) should behave. The new security definition employs the notion of *window one wayness*. That is the probability of the adversary to successfully identify the range of a plaintext message given many randomly chosen ciphertexts. They also introduce the notion of *distance window one wayness* where the adversary is further restricted to identify the interval r between two plaintexts given a large set of ciphertexts.

5.4.2 Our Protocol

The protocol consists of two phases. During the first phase each smart meter encrypts with an OPE its meterings and it sends it to the supplier along with a MAC. Afterwards, in a second phase the supplier collects all the encrypted values from each sm_i and sorts them. Since the encryption uses OPE the supplier can discover the ordering of the ciphertexts. The purpose of the protocol is for the supplier to identify high energy consumption periods for each householder. Simply by using an order preserving encryption scheme, which preserves the order of the plaintexts at the ciphertext space would solve the problem, since home tenants tend to spontaneously switch on/off high energy appliances. That results in a faulty inference by the energy supplier. As such the supplier must not only recognize peaks for high electricity consumptions but also confirm a continuous duration of the maximum consumption. To address this requirement along with its meterings, each smart meter sm_i sends discretized differences between consecutive meterings in such a way that the supplier can only verify the interval where the consumption differences equal 0 which is interpreted as a continuous maximum energy consumption.

We now describe the protocol according to the definition in section 5.2.3 :

Setup(1^λ) It is a randomized lagorithm that on input of the security parameter 1^λ outputs for each user a secret key $\mathsf{sk_i}$ and mac key mk_i. The mac key mk_i is shared with the supplier under and a confidential channel.

Encrypt$(x_{i,t}, \mathsf{sk_i}, \mathsf{mk}_i) \rightarrow \{c_{i,t}, d_{i,t}, \mathsf{mac}_{i,t}\}$ Each sm_i encrypts its meterings $x_{i,t}$ with its secret key $\mathsf{sk_i}$ using an OPE scheme. For each ciphertext $c_{i,t}$ for time interval t it also sends t as auxiliary information associated with each ciphertext. For each two sequential time intervals each smart meter sends $d_{i,t}$. Each smart meter then applies the MAC with the MAC key mk_i to the encrypted data $c_{i,t}$ and the discretized differences $d_{i,t}$ and sends $c_{i,t}||MAC_{\mathsf{mk_i}}(c_{i,t}, d_{i,t})$ to the suppliera long with $d_{i,t}$.

Analyze$(\{c_{i,t}\}, \{d_{i,t}\}, \mathsf{mac}_{i,t}, \mathsf{mk}_i) \rightarrow t_j$: The supplier collects at each time interval t the encrypted smart meterings from each sm_i. If the computed MAC by the supplier matches the MAC it obtained from the sm_i then it continues with the execution of the protocol otherwise it halts. Since the order is preserved it can identify the maximum energy consumption at time interval t_j for each sm_i. To assure a continuous duration of the maximum consumption, the supplier verifies:

$$\sum_{w_{start}}^{w_{end}} d_{i,t} \stackrel{?}{=} 0 \tag{5.1}$$

inside the miw that is specified by the supplier. The miw interval has a starting point w_{start} and an end point w_{end}. In the beginning the w_{end} is set to t_j and $w_{start} = t_j - miw$. Inside this window the analyzer checks if equation 5.1 holds in order to validate a continuous maximum energy consumption around t_j, where each d_i defines the differences of two consecutive meterings. The differences from the meterings are discretized in order to avoid inequalities from 0 even for small variations. This requirement obviously captures spontaneous switch on/offs of a high energy consumption appliance that will erroneously record maximum consumptions. If equation 5.1 is not true, it continuously checks the condition by sliding the window one position to the right until $w_{start} = t_j$. By sliding the window 1 position we mean that we advance the corresponding time frequency by 1. That is, if the smart meter reports meterings every 1 second for instance, $miw = k$ and $t_j = 23h40m40s$ then the supplier will verify equation 5.1 for $w_{start} = t_j - k$ and $w_{end} = t_j$ and will move the interval 1 second every time the condition does not hold. So the second iteration would be from $w_{start} = t_j - k + 1$ to $w_{end} = t_j + 1$ until $w_{start} = t_j$ and so on. If none of the corresponding delta differences inside miw does not satisfy the condition then the second maximum t_j is selected and the procedure restarts.

Correctness. The correctness of PPSGS depends on the correctness of the order preserving encryption scheme and on the fact that if the discretized differences of plaintext meterings are equal to 0 then:

$$\sum_{w_{start}}^{w_{end}} d_{i,t} \stackrel{?}{=} 0$$

Indeed, consider a smart meter sm_i which detects the set of plaintext values $\{x_{i,t_{j_1}}, x_{i,t_{j_2}}, x_{i,t_{j_3}}, \ldots, x_{i,t_{j_n}}\}$. These plaintext values after decreasing ordering, they form the ordered set $\mathtt{O}_p$ indexed by j which is the time interval . For every two consecutive values $x_{i,t_j}, x_{i,t_{j+1}}$ the sm_i computes the difference $d_{i,t} = x_{i,t_{j+1}} - x_{i,t_j}$ and then sends to the supplier along with the encrypted values $\{c_{i,t_{j_1}}, c_{i,t_{j_2}}, c_{i,t_{j_3}}, \ldots, c_{i,t_{j_n}}\}$ the differences discretized by a parameter ϕ $[d_{i,t}]_{\phi}$.

Thanks to the OPE the supplier can reconstruct the same ordered set $\mathtt{O}_c$ from the ciphertexts but instead of plaintext values it obtains the corresponding for the time interval j ciphertext values. If around the maximum time interval t_j there are not big difference variations then after the discretization of the differences $[d_{i,t}]_\phi = 0$ and $\sum_{w_{start}}^{w_{end}} d_{i,t} \stackrel{?}{=} 0$.

5.5 Privacy Analysis

We show in this section that the published differences do not affect the privacy requirement for third party obliviousness, which requires that nothing more other than the interval in which the smart meter has consumed the maximum energy for at least miw time interval, is revealed. We assume that the OPE in our protocol is instantiated as in [6] from the set of all possible OPE functions fixed by the secret key of the smart meter. If the OPE acts as a pseudorandom OPE fixed by a secret key then nothing more than the ordering is revealed. For our reduction we will use the POPF-CPA security definition from [23].

Let an OPE scheme $OPE = (K, Enc, Dec)$, with plaintext space $\mathcal{D}$ and ciphertext space $\mathcal{R}, |\mathcal{D}| \leq |\mathcal{R}|$. We describe the oracles an adversary $\mathcal{A}$ has access to, during the POPF-CPA game:

During the learning phase a $\mathcal{A}$ submits order preserving encryption queries to $\mathcal{O}^{\mathsf{POPF-CPA}}_{\mathsf{Encrypt}}$ for a value x and the oracles replies with $Enc(x)$. At the challenge phase $\mathcal{A}$ submits to $\mathcal{O}^{\mathsf{POPF-CPA}}_{\mathsf{C}}$ oracle two pairs of same order plaintexts: $(x_0^0, x_1^0), (x_0^1, x_1^1)$ that have not been queried at the $\mathcal{O}^{\mathsf{POPF-CPA}}_{\mathsf{Encrypt}}$ oracle, during the learning phase. $\mathcal{O}^{\mathsf{POPF-CPA}}_{\mathsf{C}}$ flips a random coin $b \stackrel{\$}{\leftarrow} \{0, 1\}$ and returns to $\mathcal{A} : \mathsf{Encrypt}(x_0^b, x_1^b)$. Eventually $\mathcal{A}$ outputs a guess b^* for the bit b.

We say that $\mathcal{A}$ succeeds in the POPF-CPA game if its guess $b^* = b$.

Definition 40. *An OPE encryption scheme is CPA secure if for any adversary $\mathcal{A}$ against an order preserving pseudorandom function under chosen-ciphertext attack (POPF-CPA) the probability $\Pr[b' = b] \leq \frac{1}{2} + \epsilon(\lambda)$, where ϵ is negligible function on input of the security parameter λ.*

Theorem 4. *The PPSGS scheme presented in section 5.4 assures third party obliviousness under the POPF-CPA security of the underlying OPE encryption scheme.*

Proof. (Sketch) Let us assume there is an adversary $\mathcal{A}^{TPO}$ that breaks third party obliviousness as presented in section section 5.4 with non negligible probability ϵ. We show in what follows that there exists an adversary $\mathcal{B}$ that uses $\mathcal{A}^{TPO}$ to break the POPF-CPA game with non-negligible advantage. For ease of exposition, we denote $\mathcal{O}^{\mathsf{POPF-CPA}}_{\mathsf{Encrypt}}$, and $\mathcal{O}^{\mathsf{POPF-CPA}}_{\mathsf{C}}$ the oracles of the POPF-CPA game and by $\mathcal{O}^{\mathsf{TPO}}_{\mathsf{Encrypt}}, \mathcal{O}^{\mathsf{TPO}}_{\mathsf{Diff}}, \mathcal{O}^{\mathsf{TPO}}_{\mathsf{C}}$ the oracles that $\mathcal{A}^{TPO}$ has access to. Now to break the POPF-CPA game, aggregator $\mathcal{B}$ simulates the third party obliviousness game of our scheme for adversary $\mathcal{A}^{TPO}$ as follows:

- Whenever $\mathcal{A}^{TPO}$ submits queries $(x, \mathbf{T}_{i,j})$ to the $\mathcal{O}^{\mathsf{TPO}}_{\mathsf{Encrypt}}$ oracle, $\mathcal{B}$ calls the $\mathcal{O}^{\mathsf{POPF-CPA}}_{\mathsf{Encrypt}}$ oracle and returns c_i to $\mathcal{A}^{TPO}$.

- $\mathcal{B}$ whenever receives $(\mathbf{T}_{i,k}, \mathbf{T}_{i,l})$ queries for the $\mathcal{O}^{\mathsf{TPO}}_{\mathsf{Diff}}$ oracle, checks if for time interval $\mathbf{T}_i$ $\mathcal{A}$ has issued an encryption query or if a message x was part of an encryption query to the $\mathcal{O}^{\mathsf{TPO}}_{\mathsf{Encrypt}}$ oracle for $\mathbf{T}_{i,j} > \mathbf{T}_{i,k}, \mathbf{T}_{i,l}$. If none of the above holds the $\mathcal{B}$ forwards to $\mathcal{A}$ the difference $x_{i,k} - x_{i,l}$.

- $\mathcal{A}^{TPO}$ submits two plaintext values $d_0 = x_2 - x_0, d_1 = x_2 - x_1$ to $\mathcal{B}$ that correspond to the delta encodings that $\mathcal{A}$ receives during the protocol execution.

- Sequentally $\mathcal{B}$ so as to simulate the $\mathcal{O}^{\mathsf{TPO}}_{\mathsf{C}}$ oracle, it submits to $\mathcal{O}^{\mathsf{POPF-CPA}}_{\mathsf{C}}$ oracle the pairs $(x^0_a, x^1_b), (x^0_c, x^1_d)$, such that $d_0 = x_a - x_b, d_1 = x_c - x_d$.

- $\mathcal{O}^{\mathsf{POPF-CPA}}_{\mathsf{C}}$ in turn flips a random coin $b \xleftarrow{\$} \{0, 1\}$ and responds to $\mathcal{B}$ with c_a, c_b if $b = 0$ or with c_c, c_d if $b = 1$.

- $\mathcal{B}$ finally forwards to $\mathcal{A}$ the ciphertext pair that it received from $\mathcal{O}^{\mathsf{POPF-CPA}}_{\mathsf{C}}$.

The adversary $\mathcal{A}$ cannot tell whether it is interacting with the actual oracles or with adversary $\mathcal{B}$ during this simulated game. Now, $\mathcal{A}$ outputs a guess b' for the bit b. Note that if $\mathcal{A}$ has a non-negligible advantage ϵ in breaking the third party obliviousness of our scheme, then this entails that it outputs a correct guess b' for the bit b with a non-negligible advantage ϵ. Finally in order to win the POPF-CPA game $\mathcal{B}$ outputs the guess $b^* = b'$.

To conclude, if there is an adversary $\mathcal{A}$ which breaks the third party obliviousness of our solution, then there exists an adversary $\mathcal{B}$ which breaks the POPF-CPA game of [23] with some non-negligible advantage ϵ: $\mathbf{Adv}[\mathcal{A}^{\mathsf{TPO}}] \leq \mathbf{Adv}[B^{POPF-CPA}] \leq \epsilon(\lambda)$ □

5.6 Feasibility

5.6.1 Smart Meter Computation Cost

Real-world smart meters that are deployed in houses are equipped with low-cost, ultra-low power microcontrollers (MCU). We assume the existence of the widely used 16-bit RISC MSP430X MCU. They consist of flash memory that can be extended up to 256KB, read-only-memory and a distinct clock rate for their CPU that ranges from 8MHz to 25MHz. Some of them are equipped with a radio frequency transceiver for wireless communication. For the metering procedure they have sensors that measure energy and an analog-to-digital converter. We analyze the feasibility of the protocol with respect to space and time overhead based on a 16-bit RISC MSP430 MCU, with 256 KB flash memory, 20 MHz clock rate and an AES instruction set coming in the AES accelerator hardware module that can speed up AES encryption in CTR mode up to 8 times [80].

In table 5.2, we show the computational and storage overhead of our solution. Since our OPE like in [23] is based on the a symmetric block cipher, we refer to the performance analysis of AES in counter mode on a 16-bit RISC MSP430 MCU with an AES accelerator module described in [80] and further compute the cost of our solution. Results are shown in a per day analysis considering different time slots.

To compute the storage overhead of the solution we observe from real data [13] that the maximum energy consumption of smart meters deployed in a 1700 square foot home do not exceed 1000kW and therefore can be represented by 2 bytes. Since the minimum block size for AES is 128 bits (16 bytes) a metering value can be considered as 1 AES block. Thus for the computational overhead we consider the cost of 1 block AES encryption. I.e: the first row of table 5.2 shows that in 1 day we can have $24 * 60 * 60 = 86400$ meterings that correspond to $86400*2 = 172.8$ MB for a total computational cost of 13.3 million cycles for the OPE encryption of all the meterings.

Table 5.2: Per day computational and storage overhead of OPE

Period (seconds)	#Meterings	Flash(KB)	Time (Mcb)
1	86400	172.8	13.33
2	43200	86.4	6.32
3	28800	56.6	4.08
4	21600	43.2	2.99
5	17280	34.5	2.35
6	14400	28.8	1.93
7	12343	24.6	1.63
8	10800	21.6	1.41
9	9600	19.3	1.24
10	8640	17.2	1.10

Table 5.3: Space and computation analysis. Mcb denotes megacycles per block

5.6.2 Server Computation Cost

The procedure that dominates the computational overhead of the server is the sorting of the meterings. The server must first sort all per user encrypted meterings in a separate data structure. Each encrypted smart metering $c_{i,t}$ is associated with a tag which is the time interval j. We consider that the server holds a binary search tree (BST) for each user. The BST provides an efficient way to keep a set of elements sorted [50]. In the average case it has $O(\log N)$ complexity for insertions and $O(\log N)$ to find the maximum element of the BST. Thus the computational complexity per smart meter for m metering is $O(\log m)$.

For the verification of the maximum continuous interval the server also has to perform n additions ($\sum_{w_{start}}^{w_{end}} d_{i,t} \stackrel{?}{=} 0$) per smart meter, where n is the number of differences provided by smart meter inside the miw. The miw is orders of magnitude smaller than the meterings. Thus n additions are performed in the best case in which the server identifies a maximum continuous energy consumption inside the miw. In the worst case the server has to compute $O((n-1)\cdot\frac{TotalDuration}{miw})$ multiplications where $TotalDuration$ corresponds to the overall metering duration.

5.7 Summary

In this Chapter we presented a protocol for personalized statistics in a smart grid environment by showing that a reconciliation of privacy and utility is achievable. The solution is based on an encryption scheme that preserves the order of the plaintexts in the ciphertext space along with an appropriate delta encoding scheme. We proved the privacy of the protocol with a reduction proof to the POPF-CPA [23] assumption of the OPE. The storage and computational costs of the protocol are analyzed with real data. For the analysis we assumed real world microcontrollers.

.

Chapter 6

Private and Dynamic Time-Series Data Aggregation with Trust Relaxation

Contents

6.1 Introduction

In this Chapter, following the goals of this dissertation as presented in Chapter 1.4, we propose a Privacy Preserving Data Collection and Analysis (**PPDCA**) protocol that eliminates the need for key redistribution following a user join or leave and the need for fully trusted key dealer. As such we strengthen the threat model of current **PPDCA** protocols with enhanced functionalities of dynamicity and fault tolerance. The features of the enhanced protocol can be summarized as follows:

- *No key dealer.* Contrary to most of previous privacy preserving aggregation protocols, there is no trusted key dealer in our scheme. In contrast, we introduce a *semi-trusted* party called Collector which gathers partial key information from users through a *secure* channel.

- *Support for dynamic user populations.* No coordination is required to manage changes in the population of users. This is possible due to a self-generated key mechanism by which no key agreement between users is required.

- *Privacy.* With respect to privacy, the scheme assures *Aggregator obliviousness* as introduced by Elaine Shi *et al.* [132]. That is, the untrusted Aggregator only learns the sum and the average over users' private data at the end of the protocol execution. Moreover, we show that the Collector does not derive any information about the users' private data.

- *Efficiency.* Like Joye *et al.* [92] our scheme enables the computation of the sum and the average over a large number of users without restrictions on the range of users' values. It is also scalable in the sense that decryptions performed by the Aggregator do not depend on the number of users.

6.2 Related Work

Önen and Molva [121] introduced a scheme to compute aggregate statistics over wireless sensor networks with multilayer encryption by transforming a block cipher into a symmetrically homomorphic encryption. Even if the proposed solution provides generic confidentiality, the sink-Aggregator is fully trusted and shares keys with the sensors. In [63], the authors proposed a protocol for secure aggregation of data using a modified version of Paillier homomorphic encryption. The Aggregator which is interested in learning the aggregate sum of data is able to decrypt without knowing the decryption key. The idea behind the scheme is a secret sharing mechanism executed between users such that the aggregation of encrypted data reveals the sum if and only if all users' data is aggregated. However, this scheme suffers from an increased communication cost due to secret share exchange between users. A solution that blends multiparty computation with homomorphic encryption is also presented in [102], but contrary to our scheme it does not address the issue of dynamic group management.

The authors in [15, 42, 87, 124] studied privacy preserving data collection protocols with differential privacy. The combination of differential privacy with non conventional encryption schemes can provide an acceptable trade-off between privacy and utility. In [124], a secret sharing mechanism and additively homomorphic encryption are employed together with the addition of appropriate noise to data by the users. Upon receiving the encrypted values a second round of communication is required between users and Aggregator to allow for partial decryption and noise cancellation. At the end of the protocol, the Aggregator learns the differential private sum. Jawurek and Kerschbaum [87] eliminate this extra communication round between the users and the Aggregator by introducing a key manager which unfortunately can decrypt users' individual data. Barthe *et al.* [15] proposed a solution whereby each smart meter in the protocol establishes an ephemeral DH shared secret with all the Aggregators. In their scheme the service provider is willing to learn a noisy weighted sum. Interestingly dynamic leaves and joins are supported with the cost of shared secrets between the smart meter and all the Aggregators. Aggregators also, unless they collude they cannot learn individual meterings.

Chan *et al.* [42] devised a privacy preserving aggregation scheme that computes the sum of users' data, and handles user joins and leaves of smart meters and arbitrary user failures. The decrypted sum is perturbed with geometric noise which ensures differential privacy. Nonetheless, this solution calls for a fully trusted dealer that is able to decrypt users' individual data. The authors in [104] presented a solution to tackle the issue of key redistribution after a user joins or leaves. The propounded solution is based on a ring based grouping technique in which users are clustered into disjoint groups, and consequently, whenever a user joins or leaves only a fraction of the users is affected.

The existing work that resembles the most ours is the work of [92, 132]. Actually, Song *et al.* [132] employs an additively homomorphic encryption scheme with differential noise to ensure Aggregator obliviousness. The proposed solution is based on a linear correlation between the keys which is known to the untrusted Aggregator. However the decrypted sum is encoded as an exponent, thus forcing a small plaintext space. Whereas Joye *et al.* [92] designed a solution that addresses the efficiency issues of [132]. Notably, Joye *et al.* [92] introduced a nifty solution to compute discrete logarithms in composite order groups in which the decision composite residuosity problem is intractable. Still, the scheme in [92] depends on a fully trusted key dealer which renders the scheme impractical for a real world application. Moreover, both schemes do not tackle either the issue of dynamic group management or user failures.

6.3 Problem Statement

We consider a scenario where an Aggregator $\mathcal{A}$ would like to compute the aggregate sum of the private data of some users $\mathcal{U}_i$. Similarly to the work of [92] and [132], we restrict ourselves to time-series data which is a series of data point observations measured at equally spaced time intervals. A straightforward approach to compute the aggregate sum would be encrypting $\mathcal{U}_i$'s individual data using the public key of $\mathcal{A}$. This solution however relies on a *trusted* Aggregator which first decrypts the users' individual data using its secret key then computes the sum. To tackle this issue, [92] and [132] employ a combination of secret sharing techniques and additively homomorphic encryption to enable Aggregator $\mathcal{A}$ to compute the sum of users' data without compromising users' privacy. The idea is to have a *trusted third party* called *key dealer* that provides each user $\mathcal{U}_i$ with a secret share sk_i while supplying the Aggregator $\mathcal{A}$ with the secret key sk_A defined as $-\sum \mathsf{sk}_i$. Each user $\mathcal{U}_i$ encrypts its private data using its secret share sk_i and forwards the resulting ciphertext to the Aggregator, which in turn combines the received ciphertexts so as to obtain an encryption of the sum of the users' data that can be decrypted using the Aggregator's secret key sk_A.

Although such solutions prevent the Aggregator from learning users' confidential data, they suffer from two main limitations which we aim to address in this chapter. The first limitation is that they build upon the assumption that the key dealer is *trusted* and does not have any interest in undermining user privacy. Whereas the second shortcoming –which is generally overlooked– is that these solutions only support static groups of users and as a result they are fault intolerant. Namely, in the case of user failures, Aggregator $\mathcal{A}$ cannot compute the aggregate sum. Along these lines, we propose a solution for privacy preserving data aggregation of time-series data that draws upon the work of [92] and which in addition to supporting *dynamic group management* and arbitrary *user failures* does not depend on trusted key dealers. The idea is to introduce an *intermediary* untrusted party that we call Collector, which helps the Aggregator $\mathcal{A}$ with the computation of the sum of users' individual data, without any prior distribution of secret keys by a trusted dealer.

6.3.1 Entities

A scheme for dynamic and privacy preserving data aggregation for time-series involves the following entities:

- Users $\mathcal{U}_i$: At each specific time interval t, each user $\mathcal{U}_i$ produces a data point $x_{i,t}$ that it wants to send to an Aggregator. Each data point contains private sensitive information pertaining to user $\mathcal{U}_i$. To protect the confidentiality of the value of $x_{i,t}$ against the Aggregator and eavesdroppers, user $\mathcal{U}_i$ encrypts $x_{i,t}$ using some secret input sk_i and forwards the resulting ciphertext $c_{i,t}$ to the Aggregator. It also sends to the Collector some auxiliary information $\mathsf{aux}_{i,t}$ that will be used later to compute the aggregate sum of individual data. Without loss of generality, we denote $\mathbb{U}$ the set of users $\mathcal{U}_i$ in the system.

- Collector $\mathcal{C}$: It is an *untrusted party* which upon receiving the auxiliary information $\mathsf{aux}_{i,t}$ sent by users $\mathcal{U}_i \in \mathbb{U}$ at time interval t computes a function g of $\mathsf{aux}_{i,t}$. Hereafter, we denote aux_t the output of function g at time interval t.

- Aggregator $\mathcal{A}$: It is an *untrusted entity* which upon receipt of ciphertexts $c_{i,t}$ and the auxiliary information aux_t at time interval t computes the sum $\sum_{\mathcal{U}_i \in \mathbb{U}} x_{i,t}$ over the data points $x_{i,t}$ underlying ciphertexts $c_{i,t}$.

6.3.2 Privacy Preserving and Dynamic Time-Series Data Aggregation

A privacy preserving and dynamic time-series data aggregation protocol consists of the following algorithms:

- $\textsf{\textbf{Setup}}(1^\lambda) \rightarrow (\mathcal{P}, \mathsf{sk}_A, \mathsf{sk}_C, \{\mathsf{sk}_i\}_{\mathcal{U}_i \in \mathbb{U}})$: It is a randomized algorithm which on input of a security parameter λ, outputs the public parameters $\mathcal{P}$ that will be used by subsequent algorithms, the secret key sk_A of Aggregator $\mathcal{A}$, the secret key sk_C of Collector $\mathcal{C}$ and the secret keys $\{\mathsf{sk}_i\}_{\mathcal{U}_i \in \mathbb{U}}$ of users $\mathcal{U}_i$.

- $\textsf{\textbf{Encrypt}}(t, \mathsf{sk}_i, x_{i,t}) \rightarrow c_{i,t}$: It is a deterministic algorithm which on input of time interval t, secret key sk_i of user $\mathcal{U}_i$ and data point $x_{i,t}$, encrypts $x_{i,t}$ and outputs the resulting ciphertext $c_{i,t}$.

- $\textsf{\textbf{Collect}}((\mathsf{aux}_{i,t})_{\mathcal{U}_i \in \mathbb{U}}, \mathsf{sk}_C) \rightarrow \mathsf{aux}_t$: It is a deterministic algorithm executed by Collector $\mathcal{C}$ which on input of the auxiliary information $(\mathsf{aux}_{i,t})_{\mathcal{U}_i \in \mathbb{U}}$ provided by individual users $\mathcal{U}_i$ and Collector $\mathcal{C}$'s secret key sk_C computes a function g over $\mathsf{aux}_{i,t}$ and outputs the result aux_t.

- $\textsf{\textbf{Aggregate}}(\{c_{i,t}\}_{\mathcal{U}_i \in \mathbb{U}}, \mathsf{aux}_t, \mathsf{sk}_A) \rightarrow \sum x_{i,t}$: It is a deterministic algorithm run by Aggregator $\mathcal{A}$. It takes as inputs ciphertexts $\{c_{i,t}\}_{\mathcal{U}_i \in \mathbb{U}}$, auxiliary information aux_t supplied by Collector $\mathcal{C}$ and the secret key sk_A of the Aggregator $\mathcal{A}$, and outputs the sum $\sum x_{i,t}$, where $x_{i,t}$ is the plaintext underlying ciphertext $c_{i,t}$.

6.3.3 Privacy Definitions

In accordance with the work of [92,132], we assume in this chapter an honest-but-curious model. This means that while the participants in the protocol are interested in learning the individual data of users, they still comply with the aggregation protocol. Namely, users are always presumed to submit a correct input to the aggregation protocol. Actually, data pollution attacks where users submit bogus values to the Aggregator is orthogonal to the problem of privacy preserving data aggregation. We also assume that while users $\mathcal{U}_i$ may collude with either Aggregator $\mathcal{A}$ or Collector $\mathcal{C}$ by disclosing their private inputs, Aggregator $\mathcal{A}$ and Collector $\mathcal{C}$ never collude.

In this section, we present two formalizations: The first one defines privacy against Aggregator $\mathcal{A}$ which we call in compliance with previous work *Aggregator obliviousness*, whereas the second formalization defines privacy against Collector $\mathcal{C}$ which we refer to as *Collector obliviousness.*

6.3.3.1 Aggregator Obliviousness

Aggregator Obliviousness (AO) ensures that for each time interval t, the Aggregator learns nothing other than the value of $\sum_{\mathcal{U}_i \in \mathbb{U}} x_{i,t}$ from ciphertexts $c_{i,t}$ and the auxiliary information aux_t that it receives from users $\mathcal{U}_i \in \mathbb{U}$ and Collector $\mathcal{C}$ respectively. It ensures also that even if Aggregator $\mathcal{A}$ colludes with an arbitrary set of users $\mathbb{K} \subset \mathbb{U}$, it will only be able to learn the value of the aggregate sum of honest users (i.e. $\sum\limits_{\mathcal{U}_i \in \mathbb{U} \setminus \mathbb{K}} x_{i,t}$) and nothing else.

To formally capture the capabilities of an Aggregator $\mathcal{A}$ against the privacy of aggregation protocols, we assume that $\mathcal{A}$ is given access to the following oracles:

- $\mathcal{O}_{\mathsf{setup},\mathcal{A}}$: When called, this oracle provides Aggregator $\mathcal{A}$ with the public parameters denoted $\mathcal{P}$ of the aggregation protocol and any secret information sk_A that may be needed by Aggregator $\mathcal{A}$ to perform the aggregation.

- $\mathcal{O}_{\mathsf{encrypt}}$: When queried with time t, identifier uid_i of some user $\mathcal{U}_i$ and a data point $x_{i,t}$, oracle $\mathcal{O}_{\mathsf{encrypt}}$ outputs the encryption $c_{i,t}$ of $x_{i,t}$ in time interval t using $\mathcal{U}_i$'s secret key sk_i.

- $\mathcal{O}_{\mathsf{corrupt}}$: When queried with the identifier uid_i of some user $\mathcal{U}_i$, the oracle $\mathcal{O}_{\mathsf{corrupt}}$ returns the secret key sk_i of user $\mathcal{U}_i$.

- $\mathcal{O}_{\mathsf{collect},\mathcal{A}}$: When called with time t, this oracle returns the auxiliary information aux_t that Collector $\mathcal{C}$ computed during time interval t. We note that in schemes such as [92, 132] where a Collector is not needed, the Aggregator will not call this oracle.

Algorithm 1 Learning phase of the Aggregator obliviousness game

$(\mathcal{P}, \mathsf{sk}_A) \leftarrow \mathcal{O}_{\mathsf{setup},\mathcal{A}}$; // $\mathcal{A}$ executes the following a polynomial number of times
$\mathsf{sk}_i \leftarrow \mathcal{O}_{\mathsf{corrupt}}(\mathsf{uid}_i)$;
$\mathcal{A} \rightarrow t$;
// $\mathcal{A}$ is allowed to call $\mathcal{O}_{\mathsf{encrypt}}$ for all users $\mathcal{U}_\mathsf{i}$
$c_{i,t} \leftarrow \mathcal{O}_{\mathsf{encrypt}}(t, \mathsf{uid}_i, x_{i,t})$;
$\mathsf{aux}_t \leftarrow \mathcal{O}_{\mathsf{collect},\mathcal{A}}(t)$;

Algorithm 2 Challenge phase of the Aggregator obliviousness game

$\mathcal{A} \rightarrow t^*, \mathbb{S}^*$;
$\mathcal{A} \rightarrow \mathcal{X}^0_{t^*}, \mathcal{X}^1_{t^*}$;
$\langle (c^b_{i,t^*})_{\mathcal{U}_i \in \mathbb{S}^*}, \mathsf{aux}^b_{t^*} \rangle \leftarrow \mathcal{O}_{\mathsf{AO}}(\mathcal{X}^0_{t^*}, \mathcal{X}^1_{t^*})$;
$\mathcal{A} \rightarrow b^*$;

- $\mathcal{O}_{\mathsf{AO}}$: When called with a subset of users $\mathbb{S} \subset \mathbb{U}$ and with two time-series $(\mathcal{U}_i, t, x^0_{i,t})_{\mathcal{U}_i \in \mathbb{S}}$ and $(\mathcal{U}_i, t, x^1_{i,t})_{\mathcal{U}_i \in \mathbb{S}}$ such that $\sum x^0_{i,t} = \sum x^1_{i,t}$, this oracle flips a random coin $b \in \{0,1\}$ and returns an encryption of the time-serie $(\mathcal{U}_i, t, x^b_{i,t})_{\mathcal{U}_i \in \mathbb{S}}$ (that is the tuple of ciphertexts $(c^b_{i,t})_{\mathcal{U}_i \in \mathbb{S}}$) and the corresponding auxiliary information aux^b_t that Aggregator $\mathcal{A}$ should receive from the Collector in time interval t.

Aggregator $\mathcal{A}$ has access to the above oracles in two phases: a learning and a challenge phase. In the learning phase (cf. Algorithm 1), Aggregator $\mathcal{A}$ first calls the oracle $\mathcal{O}_{\mathsf{setup},\mathcal{A}}$ that provides $\mathcal{A}$ with the set of public parameters $\mathcal{P}$ associated with the aggregation protocol together with any secret information sk_A that Aggregator $\mathcal{A}$ may need to execute the aggregation correctly. Next, $\mathcal{A}$ compromises users $\mathcal{U}_i$ by calling the oracle $\mathcal{O}_{\mathsf{corrupt}}$ which returns the secret keys of compromised users. Then, $\mathcal{A}$ picks a time interval t and issues encryption queries $(t, \mathsf{uid}_i, x_{i,t})$ to the oracle $\mathcal{O}_{\mathsf{encrypt}}$ which outputs the corresponding ciphertexts $c_{i,t}$. Finally, $\mathcal{A}$ calls the oracle $\mathcal{O}_{\mathsf{collect}}$ to get the auxiliary information aux_t computed by Collector $\mathcal{C}$ in time interval t.

In the challenge phase (see Algorithm 2), Aggregator $\mathcal{A}$ chooses a subset $\mathbb{S}^*$ of users that were not compromised and a challenge time interval t^* for which it did not make an encryption query during the learning phase. $\mathcal{A}$ then submits two time-series $\mathcal{X}^0_{t^*} = (\mathcal{U}_i, t^*, x^0_{i,t^*})_{\mathcal{U}_i \in \mathbb{S}^*}$ and $\mathcal{X}^1_{t^*} = (\mathcal{U}_i, t^*, x^1_{i,t^*})_{\mathcal{U}_i \in \mathbb{S}^*}$ to the oracle $\mathcal{O}_{\mathsf{AO}}$, such that $\sum x^0_{i,t} = \sum x^1_{i,t}$. Oracle $\mathcal{O}_{\mathsf{AO}}$ accordingly flips a coin $b \in \{0,1\}$ and returns the encryption $(c^b_{i,t^*})_{\mathcal{U}_i \in \mathbb{S}^*}$ of the time-serie $\mathcal{X}^b_{t^*}$ and the auxiliary information $\mathsf{aux}^b_{t^*}$ computed by Collector $\mathcal{C}$ for time interval t^*. At the end of the challenge phase, Aggregator $\mathcal{A}$ outputs a guess b^* for the bit b.

We say that Aggregator $\mathcal{A}$ succeeds in the Aggregator obliviousness game, if its guess $b^* = b$.

Definition 41 (Aggregator Obliviousness)**.** *An aggregation protocol is said to ensure Aggregator obliviousness if for any Aggregator $\mathcal{A}$, the probability* $\Pr(b = b^*) \leqslant \frac{1}{2} + \epsilon(\lambda)$*, where ϵ is a negligible function, and λ is the security parameter.*

6.3.3.2 Collector Obliviousness

Collector Obliviousness (CO) guarantees that Collector $\mathcal{C}$ cannot infer any information about the private input of individual users $\mathcal{U}_i$ either from the messages it receives directly from the users or the protocol exchange between the users and the Aggregator. It also entails that even in the case where Collector $\mathcal{C}$ colludes with a set of users $\mathbb{K}$, it does not gain any additional information about the individual values of honest users $\mathcal{U}_i$ in $\mathbb{U} \setminus \mathbb{K}$.

To formally reflect the adversarial capabilities of Collector $\mathcal{C}$ against aggregation protocols, we assume that in addition to the oracles $\mathcal{O}_{\mathsf{encrypt}}$ and $\mathcal{O}_{\mathsf{corrupt}}$, Collector $\mathcal{C}$ is given access to the following oracles:

- $\mathcal{O}_{\mathsf{setup},\mathcal{C}}$: When queried, this oracle supplies Collector $\mathcal{C}$ with the public parameters denoted $\mathcal{P}$ of the aggregation protocol and any secret information sk_C that Collector $\mathcal{C}$ may need during the aggregation protocol.

- $\mathcal{O}_{\mathsf{collect},\mathcal{C}}$: When invoked with time t, identifier uid_i of some user $\mathcal{U}_i$ and ciphertext $c_{i,t}$, this oracle returns the auxiliary information $\mathsf{aux}_{i,t}$ that corresponds to ciphertext $c_{i,t}$ that user $\mathcal{U}_i$ computed during time interval t.

- $\mathcal{O}_{\mathsf{CO}}$: When called with a subset of users $\mathbb{S} \subset \mathbb{U}$ and with two time-series $(\mathcal{U}_i, t, x^0_{i,t})_{\mathcal{U}_i \in \mathbb{S}}$ and $(\mathcal{U}_i, t, x^1_{i,t})_{\mathcal{U}_i \in \mathbb{S}}$, this oracle flips a random coin $b \in \{0, 1\}$ and returns to Collector $\mathcal{C}$ an encryption of the time-serie $(\mathcal{U}_i, t, x^b_{i,t})_{\mathcal{U}_i \in \mathbb{S}}$ (i.e. the ciphertexts $(c^b_{i,t})_{\mathcal{U}_i \in \mathbb{S}}$) and the corresponding auxiliary information computed by users $\mathcal{U}_i \in \mathbb{S}$ (i.e. $(\mathsf{aux}^b_{i,t})_{\mathcal{U}_i \in \mathbb{S}}$).

Collector $\mathcal{C}$ accesses the aforementioned oracles in a learning and a challenge phase. In the learning phase (cf. Algorithm 3), Collector $\mathcal{C}$ first queries the oracle $\mathcal{O}_{\mathsf{setup},\mathcal{C}}$ which supplies $\mathcal{C}$ with

Algorithm 3 Learning phase of the Collector obliviousness game

$(\mathcal{P}, \mathsf{sk}_C) \leftarrow \mathcal{O}_{\mathsf{setup},\mathcal{C}}$; // $\mathcal{C}$ executes the following a polynomial number of times
$\mathsf{sk}_i \leftarrow \mathcal{O}_{\mathsf{corrupt}}(\mathsf{uid}_i)$;
$\mathcal{C} \rightarrow t$;
// $\mathcal{C}$ is allowed to call $\mathcal{O}_{\mathsf{encrypt}}$ and $\mathcal{O}_{\mathsf{collect},\mathcal{C}}$ for all users $\mathcal{U}_\mathsf{i}$
$c_{i,t} \leftarrow \mathcal{O}_{\mathsf{encrypt}}(t, \mathsf{uid}_i, x_{i,t})$;
$\mathsf{aux}_{i,t} \leftarrow \mathcal{O}_{\mathsf{collect},\mathcal{C}}(t, \mathsf{uid}_i, c_{i,t})$;

Algorithm 4 Challenge phase of the Collector obliviousness game

$\mathcal{C} \rightarrow t^*, \mathbb{S}^*$;
$\mathcal{C} \rightarrow \mathcal{X}^0_{t^*}, \mathcal{X}^1_{t^*}$;
$(\langle c^b_{i,t^*}, \mathsf{aux}^b_{i,t^*}\rangle)_{\mathcal{U}_i \in \mathbb{S}^*} \leftarrow \mathcal{O}_{\mathsf{CO}}(\mathcal{X}^0_{t^*}, \mathcal{X}^1_{t^*})$;
$\mathcal{C} \rightarrow b^*$;

the set of public parameters $\mathcal{P}$ of the aggregation protocol and the secret information sk_C that Collector $\mathcal{C}$ should have to execute the aggregation properly. Then, $\mathcal{C}$ calls the oracle $\mathcal{O}_{\mathsf{corrupt}}$ to compromise users in the system. Next, it selects a time interval t and submits encryption queries $(t, \mathsf{uid}_i, x_{i,t})$ to the oracle $\mathcal{O}_{\mathsf{encrypt}}$ which outputs the corresponding ciphertexts $c_{i,t}$. Finally, it issues queries $(t, \mathsf{uid}_i, c_{i,t})$ to the oracle $\mathcal{O}_{\mathsf{collect},\mathcal{C}}$ to get the auxiliary information $\mathsf{aux}_{i,t}$ generated by users $\mathcal{U}_i$ for time interval t and ciphertext $c_{i,t}$.

In the challenge phase (see Algorithm 4), Collector $\mathcal{C}$ selects a subset $\mathbb{S}^*$ of honest users and a challenge time interval t^* for which it did not make an encryption query in the learning phase. Then, $\mathcal{C}$ queries the oracle $\mathcal{O}_{\mathsf{CO}}$ with two time-series $\mathcal{X}^0_{t^*} = (\mathcal{U}_i, t^*, x^0_{i,t^*})_{\mathcal{U}_i \in \mathbb{S}^*}$ and $\mathcal{X}^1_{t^*} = (\mathcal{U}_i, t^*, x^1_{i,t^*})_{\mathcal{U}_i \in \mathbb{S}^*}$. $\mathcal{O}_{\mathsf{CO}}$ then picks randomly a bit $b \in \{0, 1\}$ and returns the tuple $(\langle c^b_{i,t^*}, \mathsf{aux}^b_{i,t^*}\rangle)_{\mathcal{U}_i \in \mathbb{S}^*}$ for the time-serie $\mathcal{X}^b_{t^*}$. At the end of the challenge phase, Collector $\mathcal{C}$ outputs a guess b^* for the bit b.

We say that Collector $\mathcal{C}$ succeeds in the Collector obliviousness game, if its guess $b^* = b$.

Definition 42 (Collector Obliviousness)**.** *An aggregation protocol is said to ensure Collector obliviousness if for any Collector $\mathcal{C}$, the probability* $\Pr(b = b^*) \leqslant \frac{1}{2} + \epsilon(\lambda)$*, where ϵ is a negligible function, and λ is the security parameter.*

6.4 Idea of Solution

The homomorphic scheme suggested by Joye and Libert [92] allows an untrusted Aggregator to evaluate the sum or the average without any access to individual data. However to support this functionality, a fully trusted dealer has to distribute secret keys to each user $\mathcal{U}_i$ and as a result, it will be able to decrypt. Our scheme extends Joye and Libert scheme [92] through two major enhancements :

- **No key dealer:** Our scheme does not require a trusted key dealer that might get individual private data samples.
- **Dynamic group management:** In the Joye and Libert scheme [92], each join or leave operation triggers a new key redistribution for all the users in the aggregation system, whereas in our protocol, join and leave operations are possible without any key update at the users. Hence, dynamic group management is assured with significantly lower communication and computation overhead. The proposed protocol is also resilient to user failures that may occur due to communication errors or hardware failures.

In order to eliminate the need for a fully trusted dealer and to support *dynamic group management* without inducing additional communication or computation overhead, we employ two techniques:

- *Responsibility splitting mechanism*: Each user $\mathcal{U}_i$ sends an encryption of its private data sample to Aggregator $\mathcal{A}$ and an obfuscated version of its secret key sk_i to the semi trusted Collector $\mathcal{C}$, in such a way that neither the Aggregator nor the Collector can violate the privacy of individual samples provided by users.
- *Self-generation of secret keys:* The secret keys used to encrypt individual data samples are generated independently by users with no coordination by a trusted key dealer.

An overview of our solution is depicted in figure 6.1. Each user $\mathcal{U}_i$ chooses independently its secret key sk_i whereas the untrusted Aggregator generates a random key sk_A. For each time interval t, Aggregator $\mathcal{A}$ publishes an obfuscated version $\mathsf{pk}_{A,t}$ of the secret key sk_A. Users $\mathcal{U}_i$

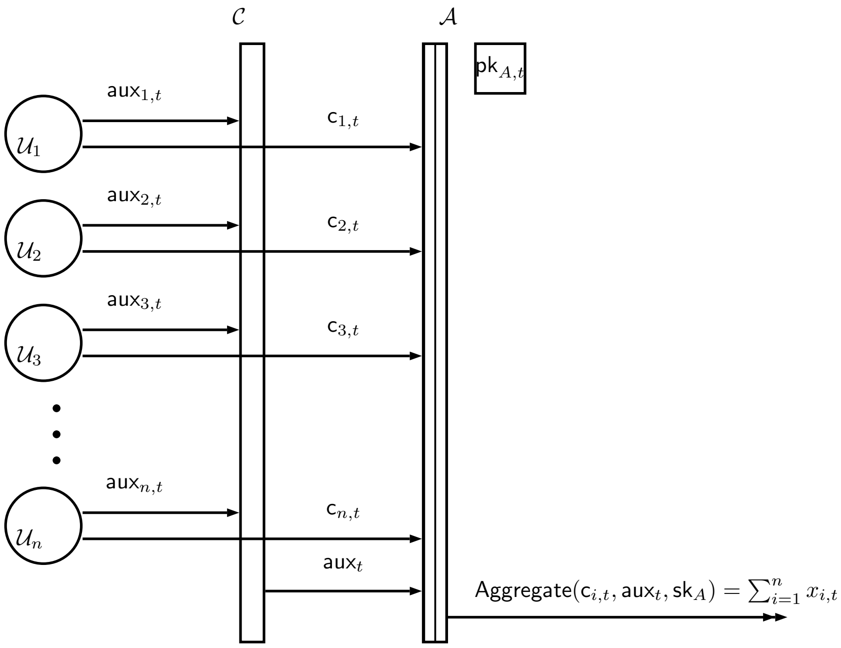


Figure 6.1: Overview of our protocol for a single time interval t. $\mathsf{pk}_{A,t}$ is public known value.

on the other hand encrypt their private data samples $x_{i,t}$ with their secret keys sk_i using the Joye-Libert cryptosystem, and send the corresponding ciphertexts $c_{i,t}$ to Aggregator $\mathcal{A}$. They also obfuscate their secret keys sk_i using $\mathsf{pk}_{A,t}$ and send the resulting auxiliary information $\mathsf{aux}_{i,t}$ to Collector $\mathcal{C}$ through a secure channel. Collector $\mathcal{C}$ computes a function $g(t)$ of the auxiliary information $\mathsf{aux}_{i,t}$ it has received and forwards the output aux_t to Aggregator $\mathcal{A}$. Upon receiving the ciphertexts $c_{i,t}$ and the auxiliary information aux_t, $\mathcal{A}$ uses its secret key sk_A and learns the sum $\sum x_{i,t}$ for the time interval t.

In this manner, we eliminate the need of a trusted key dealer that knows users' private keys while ensuring that neither the Aggregator nor the Collector can infer information about users' individual data, and we achieve efficient dynamic group management that does not call for any key update mechanism.

6.5 Protocol Description

Without loss of generality, we assume in the remainder of this section that the aggregation system comprises n users denoted $\mathbb{U} = \{\mathcal{U}_1, ..., \mathcal{U}_n\}$.

Now before providing the description of our solution, we first give a brief overview of the Joye-Libert (JL) scheme [92].

6.5.1 Joye-Libert Scheme

- $\mathsf{Setup}_{\mathsf{JL}}$: A trusted dealer $\mathcal{D}$ selects randomly two safe prime numbers p and q and sets $N = pq$. Then, it defines a cryptographic hash function $H : \mathbb{Z} \to \mathbb{Z}^*_{N^2}$ and outputs the public parameters $\mathcal{P}_{\mathsf{JL}} = (N, H)$. Finally, the dealer $\mathcal{D}$ distributes to each user $\mathcal{U}_i \in \mathbb{U}$ a secret key $\mathsf{sk}_i \in [0, N^2]$ and sends $\mathsf{sk}_A = -\sum_{i=1}^n \mathsf{sk}_i$ to the untrusted Aggregator $\mathcal{A}$.

  We note that hereafter all computations are performed "modN^2" unless mentioned otherwise.

- $\mathsf{Encrypt}_{\mathsf{JL}}$: For each time interval t, each user $\mathcal{U}_i$ encrypts its private data $x_{i,t}$ using the secret key sk_i and outputs the ciphertext $c_{i,t} = (1 + x_{i,t}N)H(t)^{\mathsf{sk}_i} \mod N^2$. We point out that ciphertexts $c_{i,t}$ fulfills the following property:

$$\begin{aligned} \prod_{i=1}^{n} c_{i,t} &= \prod_{i=1}^{n} (1 + x_{i,t}N)H(t)^{\mathsf{sk}_i} = (1 + \sum_{i=1}^{n} x_{i,t}N)H(t)^{\sum_{i=1}^{n} \mathsf{sk}_i} \\ &= (1 + \sum_{i=1}^{n} x_{i,t}N)H(t)^{-\mathsf{sk}_A} \end{aligned}$$

- $\mathsf{Aggregate}_{\mathsf{JL}}$: Upon receiving $c_{i,t}$ the untrusted Aggregator computes

$$P_t = \prod_{i=1}^{n} c_{i,t} H(t)^{\mathsf{sk}_A} = 1 + \sum_{i=1}^{n} x_{i,t}N \mod N^2$$

  and recovers $\sum_{i=1}^n x_{i,t}$ by computing $\frac{P_t - 1}{N}$ in $\mathbb{Z}$. The value $\frac{P_t-1}{N}$ is meaningful as long as $\sum_{i=1}^n x_{i,t} < N$.

We recall that the JL scheme is Aggregator oblivious in the random oracle model under the decisional composite residuosity (DCR) assumption (cf. [92]).

6.5.2 Description

Our protocol runs in four phases:

- **Setup**: A trusted third party $\mathcal{TP}$ selects two safe primes p and q, sets $N = pq$, and picks a cryptographic hash function $H : \{0,1\}^* \to \mathbb{Z}^*_{N^2}$. $\mathcal{TP}$ then publishes the public parameters $\mathcal{P} = (N, H)$ and goes offline. Next, Aggregator $\mathcal{A}$ generates a random secret key $\mathsf{sk}_A \in \mathbb{Z}^*_{N^2}$, and each user $\mathcal{U}_i \in \mathbb{U}$ independently chooses its random secret key $\mathsf{sk}_i \in [0, N^2]$ *without any coordination by a trusted key dealer.*

  It is important to note here that contrary to the JL scheme, the trusted third party $\mathcal{TP}$ does not know the individual secret keys of users $\mathcal{U}_i$, and once the public parameters $\mathcal{P}$ are published it can go offline.

- **Encrypt**: For each time interval t, each user $\mathcal{U}_i$ encrypts its private data $x_{i,t}$ using its secret key sk_i and the algorithm $\mathsf{Encrypt}_{\mathsf{JL}}$ as shown in subsection 6.5.1, and sends the resulting ciphertext $c_{i,t} = (1 + x_{i,t}N)H(t)^{\mathsf{sk}_i} \mod N^2$ to Aggregator $\mathcal{A}$.

- **Collect**: For each time interval t, Aggregator $\mathcal{A}$ publishes $\mathsf{pk}_{A,t} = H(t)^{\mathsf{sk}_A}$. Each user $\mathcal{U}_i$ then computes the auxiliary information $\mathsf{aux}_{i,t} = \mathsf{pk}_{A,t}^{\mathsf{sk}_i} = H(t)^{\mathsf{sk}_A \mathsf{sk}_i}$ using its secret key sk_i and sends $\mathsf{aux}_{i,t}$ to Collector $\mathcal{C}$ through a *secure channel.*

  Upon receiving $\mathsf{aux}_{i,t}$ $(1 \leqslant i \leqslant n)$ from users $\mathcal{U}_i \in \mathbb{U}$, Collector $\mathcal{C}$ computes

$$\mathsf{aux}_t = \prod_{i=1}^{n} \mathsf{aux}_{i,t} = \prod_{i=1}^{n} H(t)^{\mathsf{sk}_A \mathsf{sk}_i} = H(t)^{\mathsf{sk}_A \sum_{i=1}^{n} \mathsf{sk}_i}$$

  and sends the result to Aggregator $\mathcal{A}$.

  Notice here that $\mathcal{C}$ does not obtain the secret value $H(t)^{\mathsf{sk}_i}$ employed by users $\mathcal{U}_i$ during the encryption, rather it only learns an obfuscated encoding of it which is $\mathsf{aux}_{i,t} = H(t)^{\mathsf{sk}_A \mathsf{sk}_i}$.

- **Aggregate**: Upon receiving the ciphertexts $c_{i,t}$ $(1 \leqslant i \leqslant n)$ and the auxiliary information

aux_t, Aggregator $\mathcal{A}$ calculates:

$$P_t = (\prod_{i=1}^{n} c_{i,t})^{\mathsf{sk}_A} = ((1 + \sum_{i=1}^{n} x_{i,t}N)H(t)^{\sum_{i=1}^{n}\mathsf{sk}_i})^{\mathsf{sk}_A}$$
$$= (1 + \sum_{i=1}^{n} x_{i,t}N)^{\mathsf{sk}_A} H(t)^{\mathsf{sk}_A \sum_{i=1}^{n}\mathsf{sk}_i}$$

Since the order of $(1 + \sum_{i=1}^{n} x_{i,t}N)$ in $\mathbb{Z}_{N^2}^*$ is either N or divisor of N, we have:

$$P_t = (1 + \sum_{i=1}^{n} x_{i,t}N)^{\mathsf{sk}'_A} H(t)^{\mathsf{sk}_A \sum_{i=1}^{n}\mathsf{sk}_i} = (1 + \mathsf{sk}'_A \sum_{i=1}^{n} x_{i,t}N)H(t)^{\mathsf{sk}_A \sum_{i=1}^{\mathsf{sk}_i}}$$

where $\mathsf{sk}'_A = \mathsf{sk}_A \mod N$.

Finally, Aggregator $\mathcal{A}$ computes $I_t = \frac{\frac{P_t}{\mathsf{aux}_t} - 1}{N} = \mathsf{sk}'_A \sum_{i=1}^{n} x_{i,t}$ in $\mathbb{Z}$ and evaluates $R_t = {\mathsf{sk}'_A}^{-1} I_t \mod N = \sum_{i=1}^{n} x_{i,t} \mod N$ to obtain the sum of $x_{i,t}$. Notice that since $\mathsf{sk}_A \in \mathbb{Z}_{N^2}^*$, sk'_A is in $\mathbb{Z}_N^*$. Now to obtain the average of the data points $x_{i,t}$, Aggregator $\mathcal{A}$ computes $\frac{R_t}{n}$ in $\mathbb{Z}$.

As in [92], the result of the aggregation is meaningful as long as $\sum_{i=1}^{n} x_{i,t} < N$.

6.5.3 Privacy Analysis

Now the privacy of the above scheme can be stated as follows:

6.5.3.1 Aggregator Obliviousness

Theorem 5. *The proposed solution ensures Aggregator obliviousness under the decisional composite residuosity (DCR) assumption in* $\mathbb{Z}_{N^2}^*$.

Proof. Assume there is an Aggregator $\mathcal{A}$ that breaks the Aggregator obliviousness of our scheme with a non-negligible advantage ϵ. We show in what follows that there exists an Aggregator $\mathcal{B}$ that uses $\mathcal{A}$ to break the Aggregator obliviousness of the JL protocol (which is ensured under DCR) with a non-negligible advantage ϵ.

For ease of exposition, we denote $\mathcal{O}^{\mathsf{JL}}_{\mathsf{setup}}$, $\mathcal{O}^{\mathsf{JL}}_{\mathsf{corrupt}}$, $\mathcal{O}^{\mathsf{JL}}_{\mathsf{encrypt}}$ and $\mathcal{O}^{\mathsf{JL}}_{\mathsf{AO}}$ the oracles needed for the Aggregator obliviousness game of the JL protocol. We also assume that the aggregation system of the JL scheme involves n users $\mathbb{U} = \{\mathcal{U}_1, ..., \mathcal{U}_n\}$, each endowed with secret key sk_i.

Now to break the Aggregator obliviousness of the JL scheme, Aggregator $\mathcal{B}$ simulates the Aggregator obliviousness game of our scheme for Aggregator $\mathcal{A}$ as follows:

Learning phase:

- To simulate the oracle $\mathcal{O}_{\mathsf{setup},\mathcal{A}}$ for Aggregator $\mathcal{A}$, $\mathcal{B}$ first invokes the oracle $\mathcal{O}^{\mathsf{JL}}_{\mathsf{setup}}$ which returns the public parameters $\mathcal{P} = \{N, H\}$ (where N is the product of two safe primes, and $H : \mathbb{Z} \to \mathbb{Z}^*_{N^2}$ is a cryptographic hash function) and the Aggregator secret key sk_B. We recall that according to the description of the JL scheme $\mathsf{sk}_B = -\sum_{i=1}^{n} \mathsf{sk}_i$. Then, $\mathcal{B}$ supplies Aggregator $\mathcal{A}$ in our scheme with the public parameters $\mathcal{P} = \{N, H\}$. After receiving $\mathcal{P}$, Aggregator $\mathcal{A}$ selects a secret key $\mathsf{sk}_A \in \mathbb{Z}^*_{N^2}$ and for each time interval t it publishes $\mathsf{pk}_{A,t} = H(t)^{\mathsf{sk_A}}$.

- Whenever $\mathcal{A}$ submits a corruption query for some user $\mathcal{U}_i$ to the oracle $\mathcal{O}_{\mathsf{corrupt}}$, $\mathcal{B}$ relays this query to the corruption oracle $\mathcal{O}^{\mathsf{JL}}_{\mathsf{corrupt}}$ of the JL scheme which accordingly returns the secret key sk_i of user $\mathcal{U}_i$.

- Whenever $\mathcal{A}$ calls the encryption oracle $\mathcal{O}_{\mathsf{encrypt}}$ with an encryption query $(t, \mathsf{uid}_i, x_{i,t})$, $\mathcal{B}$ forwards this query to $\mathcal{O}^{\mathsf{JL}}_{\mathsf{encrypt}}$ which returns the matching ciphertext $c_{i,t} = (1 + x_{i,t}N)H(t)^{\mathsf{sk}_i}$ to $\mathcal{B}$. Next, $\mathcal{B}$ provides $\mathcal{A}$ with $c_{i,t}$.

- Whenever $\mathcal{A}$ queries the collection oracle $\mathcal{O}_{\mathsf{collect},\mathcal{A}}$ with time interval t, $\mathcal{B}$ computes $\mathsf{aux}_t = \mathsf{pk}_{A,t}^{-\mathsf{sk}_B}$ which it returns to $\mathcal{A}$. Note that $\mathsf{aux}_t = \mathsf{pk}_{A,t}^{-\mathsf{sk}_B} = H(t)^{-\mathsf{sk}_A\mathsf{sk}_B} = H(t)^{\mathsf{sk}_A \sum \mathsf{sk}_i}$ corresponds to the actual auxiliary information that a Collector in our scheme could have computed.

Challenge phase: In the challenge phase, $\mathcal{A}$ chooses a subset $\mathbb{S}^*$ of users that were not compromised and a challenge time interval t^* for which it did not make an encryption query during the learning phase. $\mathcal{A}$ publishes $\mathsf{pk}_{A,t^*} = H(t^*)^{\mathsf{sk}_A}$. $\mathcal{A}$ then submits two time-series

$\mathcal{X}^0_{t^*} = (\mathcal{U}_i, t^*, x^0_{i,t^*})_{\mathcal{U}_i \in \mathbb{S}^*}$ and $\mathcal{X}^1_{t^*} = (\mathcal{U}_i, t^*, x^1_{i,t^*})_{\mathcal{U}_i \in \mathbb{S}^*}$ such that $\sum x^0_{i,t^*} = \sum x^1_{i,t^*}$ to $\mathcal{B}$ which simulates oracle $\mathcal{O}_{\mathsf{AO}}$ as follows:

- It submits the time-series $\mathcal{X}^0_{t^*}$ and $\mathcal{X}^1_{t^*}$ to the oracle $\mathcal{O}^{\mathsf{JL}}_{\mathsf{AO}}$ which picks randomly $b \in \{0, 1\}$ and returns the encryption $(c^b_{i,t})_{\mathcal{U}_i \in \mathbb{S}^*}$ for the time-serie $\mathcal{X}^b_{t^*}$.
- Then it computes the auxiliary information $\mathsf{aux}^b_{t^*} = \mathsf{pk}^{-\mathsf{sk}_B}_{A,t^*} = H(t^*)^{-\mathsf{sk}_A \mathsf{sk}_B} = H(t^*)^{\mathsf{sk}_A \sum \mathsf{sk}_i}$ matching the time interval t^*.
- Finally, $\mathcal{B}$ returns $(c^b_{i,t^*})_{\mathcal{U}_i \in \mathbb{S}^*}$ and $\mathsf{aux}^b_{t^*}$ to $\mathcal{A}$.

It is important to notice here that Aggregator $\mathcal{A}$ cannot tell whether it is interacting with the actual oracles or with Aggregator $\mathcal{B}$ during this simulated game. As a matter of fact, the messages that $\mathcal{A}$ receives during this simulation are correctly computed.

At the end of the challenge phase, $\mathcal{A}$ outputs a guess b^* for the bit b. Note that if $\mathcal{A}$ has a non-negligible advantage ϵ in breaking the Aggregator obliviousness of our scheme, then this entails that it outputs a correct guess b^* for the bit b with a non-negligible advantage ϵ. Notably, if $\mathcal{A}$ outputs $b^* = 1$, then $(c^b_{i,t^*})_{\mathcal{U}_i \in \mathbb{S}}$ is an encryption of time-serie $\mathcal{X}^1_{t^*}$; otherwise it is an encryption of time-serie $\mathcal{X}^0_{t^*}$. Now to break the Aggregator obliviousness of the JL scheme, $\mathcal{B}$ outputs the bit b^*.

To conclude, if there is an Aggregator $\mathcal{A}$ which breaks the Aggregator obliviousness of our solution, then there exists an Aggregator $\mathcal{B}$ which breaks the Aggregator obliviousness of the JL scheme with the same non-negligible advantage ϵ. This leads to a contradiction under the decisional composite residuosity assumption in $\mathbb{Z}^*_{N^2}$. □

6.5.3.2 Collector Obliviousness

Theorem 6. *The proposed scheme assures Collector obliviousness in the random oracle model under the decisional composite residuosity (DCR) assumption in $\mathbb{Z}^*_{N^2}$, the quadratic residuosity (QR) assumption in $\mathbb{Z}^*_N$ and the decisional Diffie-Hellman (DDH) assumption in the subgroup of quadratic residues in $\mathbb{Z}^*_N$.*

Proof. Assume there is a Collector $\mathcal{C}$ that breaks the Collector obliviousness of our scheme with a non-negligible advantage ϵ. We show in what follows that there exists an Aggregator $\mathcal{B}$ that uses $\mathcal{C}$ to break the Aggregator obliviousness of the JL protocol (which is ensured under DCR) with a non-negligible advantage ϵ'.

To break the Aggregator obliviousness of the JL scheme, Aggregator $\mathcal{B}$ simulates the Collector obliviousness game of our scheme to Collector $\mathcal{C}$ as follows:

Learning phase:

- To simulate the oracle $\mathcal{O}_{\mathsf{setup},\mathcal{C}}$ for Collector $\mathcal{C}$, $\mathcal{B}$ first queries the oracle $\mathcal{O}^{\mathsf{JL}}_{\mathsf{setup}}$ which returns the Aggregator's secret key sk_B and the public parameters $\mathcal{P} = \{N, H\}$ (where N is the product of two safe primes and $H : \mathbb{Z} \to \mathbb{Z}^*_{N^2}$ is a cryptographic hash function). Then, $\mathcal{B}$ supplies Collector $\mathcal{C}$ with the public parameters $\mathcal{P} = \{N, H\}$. Finally, Aggregator $\mathcal{B}$ picks randomly $\mathsf{sk}_A \in \mathbb{Z}^*_{N^2}$ and for each time interval t, $\mathcal{B}$ simulates Aggregator $\mathcal{A}$ by publishing $\mathsf{pk}_{A,t} = H(t)^{\mathsf{sk}_A}$.

- Whenever $\mathcal{C}$ queries the oracle $\mathcal{O}_{\mathsf{corrupt}}$ for some user $\mathcal{U}_i$, $\mathcal{B}$ forwards the query to the corruption oracle of the JL scheme $\mathcal{O}^{\mathsf{JL}}_{\mathsf{corrupt}}$ which outputs the secret key sk_i of user $\mathcal{U}_i$.

- Whenever $\mathcal{C}$ submits an encryption query $(t, \mathsf{uid}_i, x_{i,t})$ to oracle $\mathcal{O}_{\mathsf{encrypt}}$, $\mathcal{B}$ sends this query to $\mathcal{O}^{\mathsf{JL}}_{\mathsf{encrypt}}$ which returns the matching ciphertext $c_{i,t} = (1 + x_{i,t}N)H(t)^{\mathsf{sk}_i}$ to $\mathcal{B}$. $\mathcal{B}$ then provides $\mathcal{C}$ with ciphertext $c_{i,t}$.

- Whenever $\mathcal{C}$ queries the collection oracle $\mathcal{O}_{\mathsf{collect},\mathcal{C}}$ with time interval t, user identifier uid and ciphertext $c_{i,t}$, $\mathcal{B}$ simulates $\mathcal{O}_{\mathsf{collect},\mathcal{C}}$ as follows:
  - It submits the encryption query $(t, \mathsf{uid}_i, 0)$ to $\mathcal{O}^{\mathsf{JL}}_{\mathsf{encrypt}}$ which returns accordingly $(1 + 0 \cdot N)H(t)^{\mathsf{sk}_i} = H(t)^{\mathsf{sk}_i}$.
  - Then using sk_A it computes $\mathsf{aux}_{i,t} = H(t)^{\mathsf{sk}_i \mathsf{sk}_A}$.

It is noteworthy that the messages that $\mathcal{C}$ received so far are correctly computed. This entails that $\mathcal{C}$ cannot detect during the learning phase that it is interacting with Aggregator $\mathcal{B}$.

Challenge phase: In the challenge phase, $\mathcal{C}$ chooses a subset $\mathbb{S}^*$ of users that were not compromised and a challenge time interval t^* for which it did not make an encryption query during the learning phase. Next, $\mathcal{C}$ submits two time-series $\mathcal{X}^0_{t^*} = (\mathcal{U}_i, t^*, x^0_{i,t^*})_{\mathcal{U}_i \in \mathbb{S}^*}$ and $\mathcal{X}^1_{t^*} = (\mathcal{U}_i, t^*, x^1_{i,t^*})_{\mathcal{U}_i \in \mathbb{S}^*}$ to $\mathcal{B}$ which simulates oracle $\mathcal{O}_{\mathsf{CO}}$ as follows:

- It picks time-serie $\mathcal{X}^0_{t^*}$ and generates a new time serie $\mathcal{X}'^1_{t^*} = (\mathcal{U}_i, t^*, x'^1_{i,t^*})_{\mathcal{U}_i \in \mathbb{S}^*}$ such that $\sum x^0_{i,t} = \sum x'^1_{i,t}$ and provides oracle $\mathcal{O}^{\mathsf{JL}}_{\mathsf{AO}}$ with the time series $\mathcal{X}^0_{t^*}$ and $\mathcal{X}'^1_{t^*}$. $\mathcal{O}^{\mathsf{JL}}_{\mathsf{AO}}$ consequently flips a coin $b \in \{0, 1\}$ and returns the tuple of ciphertexts $(c^b_{i,t^*})_{\mathcal{U}_i \in \mathbb{S}^*}$ such that $(c^b_{i,t^*})_{\mathcal{U}_i \in \mathbb{S}^*}$ is an encryption of the time-serie $\mathcal{X}^0_{t^*}$ if $b = 0$; otherwise, it is an encryption of the time-serie $\mathcal{X}'^1_{t^*}$.

- Upon receipt of $(c^b_{i,t^*})_{\mathcal{U}_i \in \mathbb{S}^*}$, $\mathcal{B}$ selects randomly $\mathsf{pk}_{A,t^*} \in \mathbb{Z}^*_{N^2}$, and computes aux^b_{i,t^*} of each user $\mathcal{U}_i \in \mathbb{S}$ by picking a random number $r^b_{i,t^*} \in \mathbb{Z}^*_{N^2}$ and setting $\mathsf{aux}^b_{i,t^*} = r^b_{i,t^*}$.

- Finally, $\mathcal{B}$ gives $(\langle c^b_{i,t^*}, \mathsf{aux}^b_{i,t^*} \rangle)_{\mathcal{U}_i \in \mathbb{S}^*}$ to Collector $\mathcal{C}$. It is important to indicate here that under the DDH assumption and the random oracle model, $\mathcal{C}$ cannot detect that pk_{A,t^*} and aux^b_{i,t^*} are generated randomly, instead of being computed as $\mathsf{pk}_{A,t^*} = H(t^*)^{\mathsf{sk}_A}$ and $\mathsf{aux}_{i,t^*} = H(t^*)^{\mathsf{sk}_i \mathsf{sk}_A}$ (cf. Lemma 7).

Lemma 7. *In the random oracle model, Collector $\mathcal{C}$ cannot detect that pk_{A,t^*} and $(\mathsf{aux}_{i,t^*})_{\mathcal{U}_i \in \mathbb{S}^*}$ are generated randomly under the decisional composite residuosity (DCR) assumption in $\mathbb{Z}^*_{N^2}$, the quadratic residuosity (QR) assumption in $\mathbb{Z}^*_N$ and the decisional Diffie-Hellman (DDH) assumption in the subgroup of quadratic residues in $\mathbb{Z}^*_N$.*

The proof of lemma 7 can be found in the Appendix Chapter. Now notice that if $b = 0$ and if $\mathcal{C}$ does not detect that $\langle (\mathsf{aux}_{i,t^*})_{\mathcal{U}_i \in \mathbb{S}^*}, \mathsf{pk}_{A,t^*} \rangle$ are generated randomly, then from the point of view of Collector $\mathcal{C}$ $(\langle c^b_{i,t^*}, \mathsf{aux}^b_{i,t^*} \rangle)_{\mathcal{U}_i \in \mathbb{S}^*}$ corresponds to a well formed tuple for the time-serie $\mathcal{X}^0_{t^*}$, and as a result, $\mathcal{C}$ will have a non-negligible advantage ϵ in breaking Collector obliviousness of our scheme. Notably, $\mathcal{C}$ will output the correct guess $b^* = 0$ for the bit b with a non-negligible advantage ϵ. In this case, if $\mathcal{B}$ outputs the bit $b^* = 0$ then it will break the Aggregator obliviousness of the JL scheme with a non-negligible advantage ϵ.

If $b = 1$, then the tuple $(\langle c^b_{i,t^*}, \mathsf{aux}^b_{i,t^*} \rangle)_{\mathcal{U}_i \in \mathbb{S}^*}$ is independent of the time-series $\mathcal{X}^0_{t^*}$ and $\mathcal{X}^1_{t^*}$ submitted by $\mathcal{C}$. Consequently, $\mathcal{C}$ will return with probability 1/2 either the bit $b^* = 1$ or the bit $b^* = 0$. Therefore, to break the Aggregator obliviousness of the JL scheme, all $\mathcal{B}$ needs to do is output b^*. $\square$

6.5.4 Dynamic Group Management

Suppose at time interval t a set of users $\mathbb{F}$ fail to participate in the protocol execution. This event does not affect the computation of the aggregate sum by the Aggregator $\mathcal{A}$. Indeed, each user $\mathcal{U}_i \notin \mathbb{F}$ computes: $\mathsf{aux}_{i,t} = \mathsf{pk}_{A,t}^{\mathsf{sk}_i}$ and encrypts its data by computing $c_{i,t} = (1 + x_{i,t}N)H(t)^{\mathsf{sk}_i}$. Upon receiving the auxiliary information $\mathsf{aux}_{i,t}$ from users $\mathcal{U}_i \notin \mathbb{F}$, Collector $\mathcal{C}$ computes $\mathsf{aux}_t = \prod_{\mathcal{U}_i \notin \mathbb{F}} \mathsf{aux}_{i,t} = \prod_{\mathcal{U}_i \notin \mathbb{F}} H(t)^{\mathsf{sk}_A \mathsf{sk}_i}$. When Aggregator $\mathcal{A}$ receives the ciphertexts $c_{i,t}$ from users $\mathcal{U}_i \notin \mathbb{F}$ and aux_t from Collector $\mathcal{C}$, it first computes the product $\prod_{\mathcal{U}_i \notin \mathbb{F}} c_{i,t}$ and computes as depicted above the value of $\sum_{\mathcal{U}_i \notin \mathbb{F}} x_{i,t}$. Thus, our solution will still function correctly even when an arbitrary number of users fail to submit their contributions to the protocol as long as Collector $\mathcal{C}$ operates properly.

Similarly, if a set of k new users $\mathbb{J} = \{\mathcal{U}_1^*, ..., \mathcal{U}_k^*\}$ join the protocol at time t, nothing changes from the point of view of Aggregator $\mathcal{A}$ and Collector $\mathcal{C}$. Notably, the new users $\mathcal{U}_i^*$ compute the auxiliary information $\mathsf{aux}_{i,t}^* = \mathsf{pk}_{A,t}^{\mathsf{sk}_i^*}$ corresponding to their ciphertexts $c_{i,t}^*$. The Collector $\mathcal{C}$ in turn evaluates the product $\mathsf{aux}_t = \prod_{\mathcal{U}_i \in \mathbb{U}} \mathsf{aux}_{i,t} \times \prod_{\mathcal{U}_i^* \in \mathbb{J}} \mathsf{aux}_{i,t}^*$, whereas the Aggregator $\mathcal{A}$ calculates the product $\prod_{\mathcal{U}_i \in \mathbb{U}} c_{i,t} \times \prod_{\mathcal{U}_i^* \in \mathbb{J}} c_{i,t}^*$. Now provided with aux_t and the secret key sk_A, Aggregator $\mathcal{A}$ can derive the sum $\sum_{\mathcal{U}_i \in \mathbb{U}} x_{i,t} + \sum_{\mathcal{U}_i^* \in \mathbb{J}} x_{i,t}^*$.

6.6 Evaluation

Table 7.1 depicts the theoretical computational and communication costs of our protocol. In each time interval t, Aggregator $\mathcal{A}$ first publishes $\mathsf{pk}_{A,t} = H(t)^{\mathsf{sk}_A}$, whereas each user $\mathcal{U}_i$ computes the ciphertext $c_{i,t} = (1 + x_{i,t}N)H(t)^{\mathsf{sk}_i}$ which consists of one exponentiation, one multiplication, one addition and one hash evaluation in $\mathbb{Z}_{N^2}^*$. User $\mathcal{U}_i$ also performs an additional exponentiation to compute the auxiliary information $\mathsf{aux}_{i,t} = \mathsf{pk}_{A,t}^{\mathsf{sk}_i} = H(t)^{\mathsf{sk}_A \mathsf{sk}_i} \in \mathbb{Z}_{N^2}^*$. Then, the Collector receives the auxiliary information $\mathsf{aux}_{i,t}$ $(1 \leqslant i \leqslant n)$ and computes the product $\mathsf{aux}_t = \prod_{i=1}^{n} \mathsf{aux}_{i,t}$ which calls for $n-1$ multiplications in $\mathbb{Z}^*_{N^2}$. Finally, the Aggregator computes the sum $\sum_{i=1}^{n} x_{i,t}$ by performing $n-1$ multiplications, one exponentiation, one division in $\mathbb{Z}^*_{N^2}$ and one division in $\mathbb{Z}$. Moreover, if l is the size in bits of N, then each user $\mathcal{U}_i$ sends $2l$ bits for ciphertext $c_{i,t}$ to Aggregator $\mathcal{A}$ and $2l$ bits for $\mathsf{aux}_{i,t}$ to Collector $\mathcal{C}$. As such, the overall communication cost per user is $4l$ per time interval.

Algorithm	Computation	Communication
User	$\mathbf{2}$ EXP $+\mathbf{1}$ MULT $+\mathbf{1}$ ADD $+\mathbf{1}$HASH	$\mathbf{4 \cdot l}$
Aggregator	$\mathbf{2}$ EXP $+\mathbf{2}$ DIV $+(\mathbf{n-1})$ MULT $+\mathbf{1}$HASH	$\mathbf{2 \cdot l}$
Collector	$(\mathbf{n-1})$ MULT	$\mathbf{2 \cdot l}$

Table 6.1: Performance analysis

6.6.1 Implementation

In order to show the feasibility of our protocol, we implemented a prototype in different hardware platforms. First we implemented our scheme in Python 3.2.3 using Charm [9, 10]. Charm is a Python library that provides cryptographic abstraction in order to build security protocols. The PC benchmarks run on an Intel Core i5 CPU M 2430 @ 2.40GHz $\times$ 4 with 6GB of memory machine, running Ubuntu 12.04 64bit. The implementation has been merged with the Charm library, which is maintained at John Hopkins University[1]. In order to show the feasibility and the efficiency of our protocol we measured the encryption time, which includes the computational overhead of the auxiliary information $\mathsf{aux}_{i,t}$ and the computation of the ciphertext by each user. Due to the simple mathematical operations involved in the computation of the ciphertext and the auxiliary information, the user side overhead is very low. In our benchmark analysis, we also included the running time of an Aggregator to decrypt the sum, and the cost of the Collector which aggregates auxiliary information and sends it to the Aggregator. We included different number of users in order to show the scalability of the protocol and different security parameters with respect to the bit-length of moduli N. As it was expected Aggregator's decryption time is proportional on and the size of the users.

[1] `https://github.com/JHUISI/charm/blob/dev/charm/schemes/lem_scheme.py`

We also put forward a comparison with Joye *et al.* [92] scheme in order to see the differences of the two schemes. The extra auxiliary information that is computed by each user increases the encryption time, and the total decryption overhead as well, since the Collector needs to aggregate all $\mathsf{aux}_{i,t}$. However, that extra overhead, allows efficient fault tolerance and dynamicity, since there is no further key distribution phase, in contrast with the work in [92].

In order to demonstrate the feasibility of our scheme in an ubiquitous environment we also performed a benchmark analysis on `cubieboard2`, which is a single board computer, with 1 GB RAM running on ARM Cortex-A7 Dual-Core. The operating system on top of `cubieboard2` platform is Ubuntu/Linaro 4.8.2. In order to boost efficiency we implemented our scheme in standard ANSI C using `libgmp` 6.0.0 and `openssl` 1.0.1. As with the benchmarks on a PC, we measured the encryption time per user and the overhead of the Collector and Aggregator in order for the latter to learn the sum.

Finally we deployed the Python code on a mobile device. We used for our implementation a SAMSUNG I9500 S4 smart phone with 2GB RAM and a quad-core 1.6 GHz Cortex A5 processor with Android 4.2.2. For this setting, we are envisioning mobile phones as end users that send their values encrypted to an Aggregator. As such we measured the cost on the user side only.

6.6.1.1 PCs

In this section we make a performance analysis on PCs. In table 6.2 we present our results that show the scalability for aggregation. Namely, for a different number of users we measured the time that the Aggregator needs to compute the sum and the time that the Collector needs to aggregate all the auxiliary information $\mathsf{aux}_{i,t}$. There is a growing overhead as the number of users increases: From $\approx$ 30 ms in order to decrypt the sum of 500 users, the computational overhead is increased to $\approx$ 1 minute for 1 million users and to $\approx$ 9 minutes for 10 million users with moduli bit-length 2048. Collector performs operations that are executed in less time than the Aggregator's operations, since the Collector does not need to perform costly mathematical operations as exponentiations or divisions but it only multiplies all the auxiliary information $\mathsf{aux}_{i,t}$, that is computed by each user. We also changed the moduli size of N to 4096 and as it was expected there is an increase at the cost for the Aggregator and the Collector due to the larger size of the group that the mathematical operations take place (table 6.3).

Entity \ #Users	500	1K	10K	100K	1M	10M
Collector	0.030	0.056	0.556	5.60	59.72	562.66
Aggregator	0.159	0.190	0.690	5.73	59.22	569.19

Table 6.2: Computational costs in seconds, for Collector and Aggregator when bit-length $|N| = 2048$ for PC benchmarks.

Table 6.4 presents the overhead on the user side due to the encryption process and the auxiliary information computation. We performed our analysis for different bit-lengths of N. Both encryption and auxiliary information computation are very efficient at the scale of ms. The computational cost for auxiliary information computation is higher than the one of computing the ciphertext, because the exponentiation performed by each user $H(t)^{\mathsf{sk}_i}$ is being reused during the ciphertext computation $c_{i,t} = (1 + x_{i,t}N)H(t)^{\mathsf{sk}_i}$ to optimize the encryption.

Finally, we make a comparison with the Joye-Libert scheme. In case of encryption, Joye-Libert scheme [92] outperforms our scheme since there is no need for auxiliary information computation at the user side (cf. table 6.5). However the extra computational cost is very low for both moduli N bit-lengths: 2048 and 4096. We also compared the decryption times. As it can be seen from figures 6.2a and 6.2b, for a population of users up to 1 million users, the difference in time for the decryption procedure is slightly higher for our scheme, since we included the cost of the Collector which aggregates the auxiliary information. The extra cost is almost doubled in case of 10 million users. These extra overhead of our scheme in comparison with Joye-Libert scheme [92] comes as a trade-off for supporting dynamicity and fault-tolerance with minimal extra computational and communication costs for existing users, since there is no need to perform a new key-distribution phase. In contrast in Joye-Libert scheme [92] at every dynamic leave or join of user the trusted key dealer has to distribute new secret keys to each user and to the Aggregator, which increases the communication cost of the scheme and the computational overhead of the key dealer.

Entity \ #Users	500	1K	10K	100K	1M	10M
Collector	0.103	0.206	2.06	20.32	205.74	2034.94
Aggregator	0.574	0.674	2.53	20.77	204.63	2257.87.19

Table 6.3: Computational costs in seconds, for Collector and Aggregator when bit-length $|N| = 4096$ for PC benchmarks.

Algorithm \ |N|	2048	4096
Encrypt	0.116	0.4
Aux	0.123	0.44
Total	0.239	0.84

Table 6.4: Computational overhead of users for encryption and auxiliary information in seconds for different security levels with respect to the bit-length of N implemented on a PC.

|N| \ Scheme	Our scheme	Joye-Libert
2048	0.239	0.156
4096	0.84	0.4

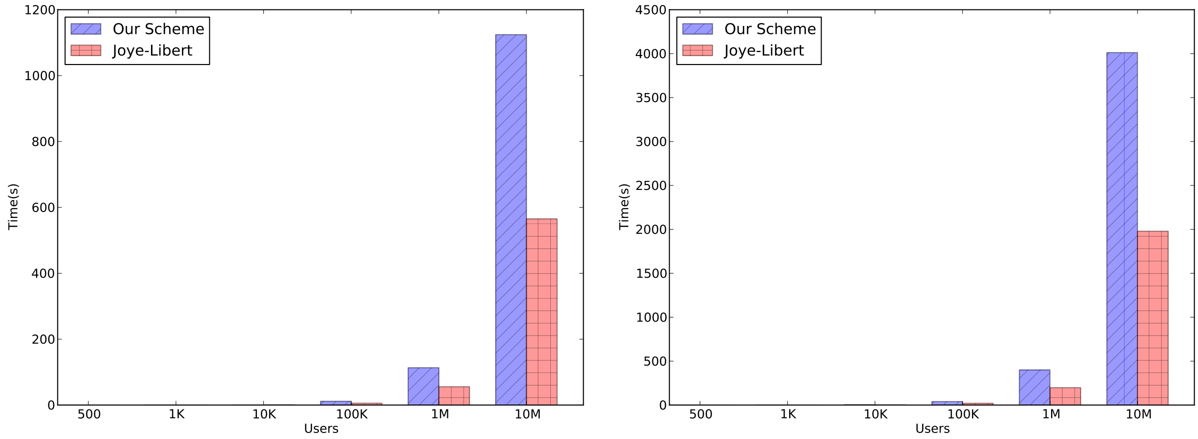
Table 6.5: Comparison in seconds for Encryption.

6.6.1.2 Cubieboard

We implemented our scheme on single board computers: `cubieboard2`. For this setting we simulated different users with a single `cubieboard2` and the Aggregator and the Collector at different `cubieboard2` platforms. We measured the decryption time for the Aggregator and the computational overhead of the Collector. In tables 6.6 and 6.7 we can see the computational overhead in seconds for different security parameters. In order to evaluate the performance on the user side we randomly selected a possible data range for users' data and we measured the total encryption time for different security parameters. We observed that the encryption time is proportional to the plaintext space. Due to the simplicity of the scheme the encryption time overall is a very fast process (cf. figure 6.3).

6.6.1.3 Mobile Device

Finally, we implemented our scheme on a mobile device running Android 4.2.2. For the deployment in Android we used the Py3KA package for the Python library for Android and the scripting layer SL4A. We computed the average encryption time per user with $|N| = 2048$ and



(a) |N|=2048 (b) |N|=4096

Figure 6.2: Decryption comparison with Joye-Libert scheme.

Entity \ #Users	500	1K	10K
Collector	0.098	0.202	2.015
Aggregator	0.100	0.202	2.014

Table 6.6: Computational costs in seconds, for Collector and Aggregator when bit-length $|N| = 2048$ for `cubieboard2` benchmarks.

Entity \ #Users	500	1K	10K
Collector	0.305	0.629	6.201
Aggregator	0.833	1.166	6.656

Table 6.7: Computational costs in seconds, for Collector and Aggregator when bit-length $|N| = 4096$ for `cubieboard2` benchmarks.

the average time to compute the auxiliary information per user. The average encryption time per user is 0.97 seconds the highest of all of our benchmark results on different platforms and the auxiliary information computational cost is 1.02 seconds. We also measured the energy consumption during encryption and is approximately 400mW (cf. Figure 6.4).

The benchmark results showed the practicality and scalability of our scheme on different platforms ranging from PCs and single board computers to mobile devices. Further code optimization can improve the computational overhead.

6.7 Summary

In this chapter, we presented a privacy preserving solution for time-series data aggregation which contrary to existing work supports arbitrary user failures and does not depend on trusted key dealers. The idea is to rely on a semi-trusted Collector which plays the role of an intermediary between the users and the Aggregator, and which enables the Aggregator to compute the aggre-

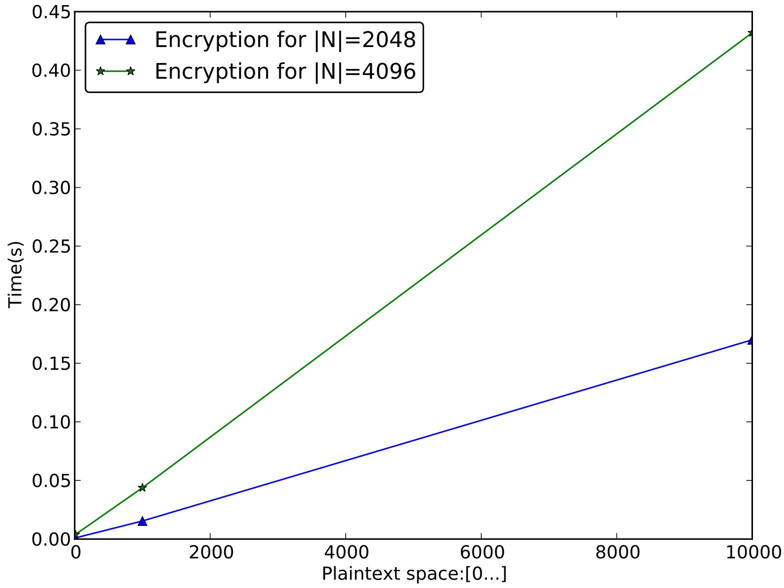


Figure 6.3: User overhead on `cubieboard2`.

gate sum of users' private data without undermining users' privacy. An interesting feature of the proposed scheme is that users' joins and leaves do not incur any additional computation or communication cost at either the users or the Aggregator. Furthermore, the scheme is provably privacy preserving against honest-but-curious Aggregators and Collectors. Finally, we evaluated the feasibility of our scheme on different platforms (PC, `cubieboard2`, Android) and we compared it with the Joye-Libert scheme [92]. The benchmark results demonstrate the practicality of the propounded solution.

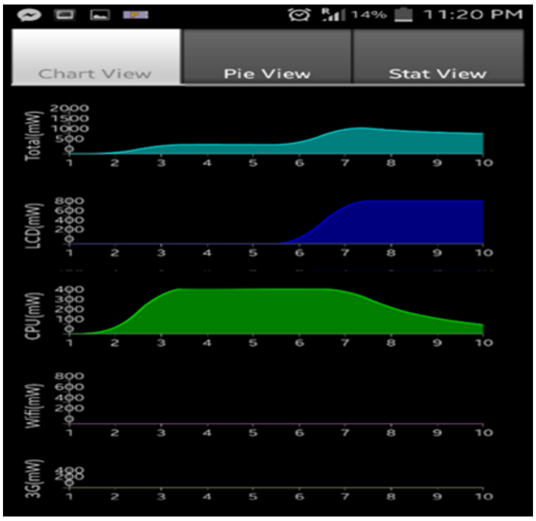


Figure 6.4: Energy consumption on SAMSUNG S4 - Android 4.2.2.

Chapter 7

PUDA - Privacy and Unforgeability for Aggregation

Contents

7.1 Introduction

In this Chapter we consider a scenario whereby an Aggregator collects individual data from multiple users which do not interact with each other and executes a function which outputs an aggregate value. In contrast with the protocol presented in Chapter 6, the result is further forwarded to a Data Analyzer which can finally extract useful information about the entire population. Existing **PPDCA** protocols only focus on the problem of data confidentiality and consider the Aggregator to be *honest-but-curious*: the Aggregator is curious in discovering the content of each individual data, but performs the aggregation operation correctly. Here we consider a more powerful security model, where the Aggregator is assumed to be malicious: The Aggregator may provide a bogus aggregate value to the Data Analyzer. In order to protect against such a malicious behavior, we propose that along with the aggregate value, the Aggregator provides a proof of the correctness of the computation of the aggregate result. We also require the Data Analyzer not to be able to communicate with each user and the result to be publicly verifiable. Moreover, similarly to the existing solutions, the proposed protocol assures obliviousness against the Aggregator and the Data Analyzer in the multi-user setting; meaning that neither the Data Analyzer nor the Aggregator learns individual data inputs.

The underlying idea of our solution is that each user encrypts its data according to Shi *et al.* [132] scheme using its own secret encryption key, and sends the resulting ciphertext to the untrusted Aggregator. Users, also homomorphically tag their data using two layers of randomness with two different keys and they forward the tags to the Aggregator. The latter computes the sum by applying operations on the ciphertexts and it also derives a proof for the correctness of the result from the tags. The Aggregator finally sends the result and the proof to the Data Analyzer. The latter verifies the correctness of the computation.

To the best of our knowledge we are the first to define a model for *Privacy and Unforgeability for Data Aggregation* (**PUDA**). We also instantiate a **PUDA** scheme which mainly pursues the following three objectives:

- Multi-user setting where multiple users produce personal sensitive data without interacting with each other.

- Public verifiability of the aggregate value.
- Privacy of individual data for all participants.

7.2 Related Work

In [37], authors proposed a solution which is based on homomorphic message authenticators in order to verify the computation of generic functions on outsourced data. Each data input is authenticated with an authentication tag. A composition of the tags is computed by the cloud in order to demonstrate the correctness of the output of a program P. Thanks to the homomorphic properties of the tags the user can verify the correctness of the program. The main drawback of the solution is that the verifier in order to verify the correctness of the computation has to be involved in computations that take exactly the same time as the computation of the function f. *Backes et al.* [12] proposed a generic solution for efficient verification of bounded degree polynomials in time less than the evaluation of f. The solution is based on *closed form efficient* pseudorandom function PRF. Contrary to our solution both solutions do not provide individual privacy and they are not designed for a multi-user scenario.

Catalano *et al.* [39] employed a nifty technique to allow single users to verify computations on encrypted data. The idea is to re-randomize the ciphertext and sign it with a homomorphic signature. Computations then are performed on the randomized ciphertext and the original one. However the aggregate value is not allowed to be learnt in cleartext by the untrusted aggregator since the protocols are geared for cloud based scenarios.

In the multi-user setting, Choi *et al.* [47] proposed a protocol in which multiple users are outsourcing their inputs to an untrusted server along with the definition of a functionality f. The server computes the result in a privacy preserving manner without learning the result and the computation is verified by a user that has contributed to the function input. The users are forced to operate in a *non-interactive* model, whereby they cannot communicate with each other. The underlying machinery entails a novel proxy based oblivious transfer protocol, which along with a fully homomorphic scheme and garbled circuits allows for verifiability and privacy. However, the need of fully homomorphic encryption and garbled circuits renders the solution impractical for a real world scenario.

7.3 Problem Statement

We are envisioning a scenario whereby a set of users $\mathbb{U} = \{\mathcal{U}_i\}_{i=1}^n$ are producing sensitive data inputs $x_{i,t}$ at each time interval t. These individual data are first encrypted into ciphertexts $c_{i,t}$ and further forwarded to an untrusted Aggregator $\mathcal{A}$. Aggregator $\mathcal{A}$ aggregates all the received ciphertexts, decrypts the aggregate and forwards the resulting plaintext to a Data Analyzer $\mathcal{DA}$ together with a cryptographic proof that assures the correctness of the aggregation operation, which in this Chapter corresponds to the *sum* of the users' individual data. An important criterion that we aim to fulfill is allowing Data Analyzer $\mathcal{DA}$ to verify the correctness of the Aggregator's output without compromising users' privacy. Namely, at the end of the verification operation, both Aggregator $\mathcal{A}$ and Data Analyzer $\mathcal{DA}$ learn nothing, but the value of the aggregation. While homomorphic signatures proposed in [29, 66] seem to answer the verifiability requirement, these solutions only consider scenarios where a single user generates data.

In the aim of assuring both individual user's privacy and unforgeable aggregation, we first come up with a generic model for privacy preserving and unforgeable aggregation that identifies the algorithms necessary to implement such functionalities and defines the corresponding privacy and security models. Furthermore, we propose a concrete solution which combines an already existing privacy preserving aggregation scheme [132] with an additively homomorphic tag designed for bilinear groups.

Notably, a scheme that allows a malicious Aggregator to compute the sum of users' data in privacy preserving manner and to produce a proof of correct aggregation will start by first running a setup phase. During setup, each user receives a secret key that will be used to encrypt the user's private input and to generate the corresponding authentication tag; the Aggregator $\mathcal{A}$ and the Data Analyzer $\mathcal{DA}$ on the other hand, are provided with a secret decryption key and a public verification key, respectively. After the key distribution, each user sends its data encrypted and authenticated to Aggregator $\mathcal{A}$, while making sure that the computed ciphertext and the matching authentication tag leak no information about its private input. On receiving users' data, Aggregator $\mathcal{A}$ first aggregates the received ciphertexts and decrypts the sum using its decryption key, then uses the received authentication tags to produce a proof that demonstrates the correctness of the decrypted sum. Finally, Data Analyzer $\mathcal{DA}$ verifies the correctness of the aggregation, thanks to the public verification key.

7.3.1 PUDA Model

A **PUDA** scheme consists of the following algorithms:

- $\textsf{\textbf{Setup}}(1^\lambda) \rightarrow (\mathcal{P}, \mathsf{sk_A}, \{\mathsf{sk}_i\}_{\mathcal{U}_i \in \mathbb{U}}, \mathsf{vk})$: It is a randomized algorithm run by a trusted dealer $\mathcal{KD}$, which on input of a security parameter λ outputs the public parameters $\mathcal{P}$ that will be used by subsequent algorithms, the Aggregator $\mathcal{A}$'s secret key $\mathsf{sk_A}$, the secret keys $\mathsf{sk_i}$ of users $\mathcal{U}_i$ and the public verification key vk.

- $\textsf{\textbf{EncTag}}(t, \mathsf{sk}_i, x_{i,t}) \rightarrow (c_{i,t}, \sigma_{i,t})$: It is a randomized algorithm which on inputs of time interval t, secret key sk_i of user $\mathcal{U}_i$ and data $x_{i,t}$, encrypts $x_{i,t}$ to get a ciphertext $c_{i,t}$ and computes a tag $\sigma_{i,t}$ that authenticates $x_{i,t}$.

- $\textsf{\textbf{Aggregate}}(\mathsf{sk_A}, \{c_{i,t}\}_{\mathcal{U}_i \in \mathbb{U}}, \{\sigma_{i,t}\}_{\mathcal{U}_i \in \mathbb{U}}) \rightarrow (\mathsf{sum_t}, \sigma_\mathsf{t})$: It is a deterministic algorithm run by the Aggregator $\mathcal{A}$. It takes as inputs Aggregator $\mathcal{A}$'s secret key $\mathsf{sk_A}$, ciphertexts $\{c_{i,t}\}_{\mathcal{U}_i \in \mathbb{U}}$ and authentication tags $\{\sigma_{i,t}\}_{\mathcal{U}_i \in \mathbb{U}}$, and outputs in cleartext the sum $\mathsf{sum_t}$ of the values $\{x_{i,t}\}_{\mathcal{U}_i \in \mathbb{U}}$. Moreover, it computes a proof σ_t attesting the correctness of $\mathsf{sum_t}$, using the authentication tags $\{\sigma_{i,t}\}_{\mathcal{U}_i \in \mathbb{U}}$.

- $\textsf{\textbf{Verify}}(\mathsf{vk}, \mathsf{sum_t}, \sigma_\mathsf{t}) \rightarrow \{0, 1\}$: It is a deterministic algorithm that is executed by the Data Analyzer $\mathcal{DA}$. It outputs 1 if Data Analyzer $\mathcal{DA}$ is convinced that the sum $\mathsf{sum_t} = \sum_{\mathcal{U}_i \in \mathbb{U}} \{x_{i,t}\}$; and 0 otherwise.

7.3.2 Security Model

We only focus on the adversarial behavior of Aggregator $\mathcal{A}$. The rationale behind this is that Aggregator $\mathcal{A}$ is the only party in the protocol that sees all the messages exchanged during the protocol execution: Namely, Aggregator $\mathcal{A}$ has access to users' ciphertexts and it is the party that interacts directly with the Data Analyzer. It follows that by ensuring security properties against the Aggregator, one by the same token, ensures these security properties against both Data Analyzer $\mathcal{DA}$ and external parties.

In accordance with previous work [92, 132], we formalize the property of *Aggregator obliviousness*, which ensures that at the end of a protocol execution, Aggregator $\mathcal{A}$ only learns the sum of users' inputs $x_{i,t}$ and nothing else. Also, we enhance the security definitions of data aggregation with the notion of *aggregate unforgeability.* As the name implies, aggregate unforgeability guarantees that Aggregator $\mathcal{A}$ cannot forge a valid proof σ_{t} for a sum $\mathsf{sum_t}$ that was not computed correctly from users' inputs (i.e. cannot generate a proof for $\mathsf{sum_t} \neq \sum x_{i,t}$).

7.3.2.1 Aggregator Obliviousness

Aggregator obliviousness ensures that when users $\mathcal{U}_i$ provide Aggregator $\mathcal{A}$ with ciphertexts $c_{i,t}$ and authentication tags $\sigma_{i,t}$, Aggregator $\mathcal{A}$ cannot reveal any information about individual inputs $x_{i,t}$, other than the sum value $\sum x_{i,t}$. We extend the existing definition of *Aggregator Obliviousness* (cf. [92, 103, 132]) so as to capture the fact that Aggregator $\mathcal{A}$ not only has access to ciphertexts $c_{i,t}$, but also has access to the authentication tags $\sigma_{i,t}$ that enable Aggregator $\mathcal{A}$ to generate proofs of correct aggregation.

Similarly to the work of [92, 132], we formalize *Aggregator obliviousness* using an indistinguishability-based game in which Aggregator $\mathcal{A}$ accesses the following oracles:

- $\mathcal{O}_{\mathsf{Setup}}$: When called by Aggregator $\mathcal{A}$, this oracle initializes the system parameters; it then gives the public parameters $\mathcal{P}$, the Aggregator 's secret key $\mathsf{sk_A}$ and public verification key vk to $\mathcal{A}$.

- $\mathcal{O}_{\mathsf{Corrupt}}$: When queried by Aggregator $\mathcal{A}$ with a user $\mathcal{U}_i$' s identifier uid_i, this oracle provides

Aggregator $\mathcal{A}$ with $\mathcal{U}_i$'s secret key denoted $\mathsf{sk_i}$.

- $\mathcal{O}_{\mathsf{EncTag}}$: When queried with time t, user $\mathcal{U}_i$'s identifier uid_i and a data point $x_{i,t}$, this oracle outputs the ciphertext $c_{i,t}$ and the authentication tag $\sigma_{i,t}$ of $x_{i,t}$ computed using $\mathcal{U}_i$'s secret key sk_i.

- $\mathcal{O}_{\mathsf{AO}}$: When called with a subset of users $\mathbb{S} \subset \mathbb{U}$ and with two time-series $(\mathcal{U}_i, t, x^0_{i,t})_{\mathcal{U}_i \in \mathbb{S}}$ and $(\mathcal{U}_i, t, x^1_{i,t})_{\mathcal{U}_i \in \mathbb{S}}$ such that $\sum x^0_{i,t} = \sum x^1_{i,t}$, this oracle flips a random coin $b \in \{0, 1\}$ and returns an encryption of the time-serie $(\mathcal{U}_i, t, x^b_{i,t})_{\mathcal{U}_i \in \mathbb{S}}$ (that is the tuple of ciphertexts $\{c^b_{i,t}\}_{\mathcal{U}_i \in \mathbb{S}}$) and the corresponding authentication tags $\{\sigma^b_{i,t}\}_{\mathcal{U}_i \in \mathbb{S}}$.

Aggregator $\mathcal{A}$ is accessing the aforementioned oracles during a learning phase (cf. Algorithm 5) and a challenge phase (cf. Algorithm 6). In the learning phase, $\mathcal{A}$ calls oracle $\mathcal{O}_{\mathsf{Setup}}$ which in turn returns the public parameters $\mathcal{P}$, the public verification key vk and the Aggregator 's secret key $\mathsf{sk_A}$. It also interacts with oracle $\mathcal{O}_{\mathsf{Corrupt}}$ to learn the secret keys sk_i of users $\mathcal{U}_i$, and oracle $\mathcal{O}_{\mathsf{EncTag}}$ to get a set of ciphertexts $c_{i,t}$ and authentication tags $\sigma_{i,t}$.

In the challenge phase, Aggregator $\mathcal{A}$ chooses a subset $\mathbb{S}^*$ of users that were not corrupted in the learning phase, and a challenge time interval t^* for which it did not make an encryption query. Oracle $\mathcal{O}_{\mathsf{AO}}$ then receives two time-series $\mathcal{X}^0_{t^*} = (\mathcal{U}_i, t^*, x^0_{i,t^*})_{\mathcal{U}_i \in \mathbb{S}^*}$ and $\mathcal{X}^1_{t^*} = (\mathcal{U}_i, t^*, x^1_{i,t^*})_{\mathcal{U}_i \in \mathbb{S}^*}$ from $\mathcal{A}$, such that $\sum x^0_{i,t^*} = \sum_{\mathcal{U}_i \in \mathbb{S}^*} x^1_{i,t^*}$. Then oracle $\mathcal{O}_{\mathsf{AO}}$ flips a random coin $b \xleftarrow{\$} \{0, 1\}$ and returns to $\mathcal{A}$ the ciphertexts $\{c^b_{i,t^*}\}_{\mathcal{U}_i \in \mathbb{S}^*}$ and the matching authentication tags $\{\sigma^b_{i,t^*}\}_{\mathcal{U}_i \in \mathbb{S}^*}$.

At the end of the challenge phase, Aggregator $\mathcal{A}$ outputs a guess b^* for the bit b.

We say that Aggregator $\mathcal{A}$ succeeds in the Aggregator obliviousness game, if its guess b^* equals b.

Algorithm 5 Learning phase of the obliviousness game

$(\mathcal{P}, \mathsf{sk_A}, \mathsf{vk}) \leftarrow \mathcal{O}_{\mathsf{Setup}}(1^\lambda)$; // $\mathcal{A}$ executes the following a polynomial number of times

$\mathsf{sk}_i \leftarrow \mathcal{O}_{\mathsf{Corrupt}}(\mathsf{uid}_i)$; // $\mathcal{A}$ is allowed to call $\mathcal{O}_{\mathsf{EncTag}}$ for all users $\mathcal{U}_\mathsf{i}$

$(c_{i,t}, \sigma_{i,t}) \leftarrow \mathcal{O}_{\mathsf{EncTag}}(t, \mathsf{uid}_i, x_{i,t})$;

Definition 43 (Aggregator Obliviousness)**.** *Let* $\Pr[\mathcal{A}^{\mathbf{AO}}]$ *denote the probability that Aggregator* $\mathcal{A}$ *outputs* $b^* = b$*. Then an aggregation protocol is said to ensure Aggregator obliviousness if*

Algorithm 6 Challenge phase of the obliviousness game

$\mathcal{A} \to t^*, \mathbb{S}^*$;
$\mathcal{A} \to \mathcal{X}^0_{t^*}, \mathcal{X}^1_{t^*}$;
$(c^b_{i,t^*}, \sigma^b_{i,t^*})_{\mathcal{U}_i \in \mathbb{S}^*} \leftarrow \mathcal{O}_{\mathsf{AO}}(\mathcal{X}^0_{t^*}, \mathcal{X}^1_{t^*})$;
$\mathcal{A} \to b^*$;

for any polynomially bounded Aggregator $\mathcal{A}$ the probability $\Pr[\mathcal{A}^{\mathbf{AO}}] \leqslant \frac{1}{2} + \epsilon(\lambda)$*, where ϵ is a negligible function and λ is the security parameter.*

7.3.2.2 Aggregate Unforgeability

We augment the security requirements of data aggregation with the requirement of *aggregate unforgeability*. More precisely, we assume that Aggregator $\mathcal{A}$ is not only interested in compromising the privacy of users participating in the data aggregation protocol, but also interested in tampering with the sum of users' inputs. That is, Aggregator $\mathcal{A}$ may sometimes have an incentive to feed Data Analyzer $\mathcal{DA}$ erroneous sums. Along these lines, we define *aggregate unforgeability* as the security feature that ensures that Aggregator $\mathcal{A}$ cannot convince Data Analyzer $\mathcal{DA}$ to accept a bogus sum, as long as users $\mathcal{U}_i$ in the system are honest (i.e. they always submit their correct input and do not collude with the Aggregator $\mathcal{A}$).

In compliance with previous work [38, 66] on homomorphic signatures, we formalize *aggregate unforgeability* via a game in which Aggregator $\mathcal{A}$ accesses oracles $\mathcal{O}_{\mathsf{Setup}}$ and $\mathcal{O}_{\mathsf{EncTag}}$. Furthermore, given the property that anyone holding the public verification key vk can execute the algorithm Verify, we assume that Aggregator $\mathcal{A}$ during the unforgeability game runs the algorithm Verify by itself.

As shown in Algorithm 7, Aggregator $\mathcal{A}$ enters the *aggregate unforgeability* game by querying the oracle $\mathcal{O}_{\mathsf{Setup}}$ with a security parameter λ. Oracle $\mathcal{O}_{\mathsf{Setup}}$ accordingly returns public parameters $\mathcal{P}$, verification key vk and the secret key sk_{A} of Aggregator $\mathcal{A}$. Moreover, Aggregator $\mathcal{A}$ calls oracle $\mathcal{O}_{\mathsf{EncTag}}$ with tuples $(t, \mathsf{uid}_i, x_{i,t})$ in order to receive the ciphertext $c_{i,t}$ encrypting $x_{i,t}$ and the matching authenticating tag $\sigma_{i,t}$, both computed using user $\mathcal{U}_i$'s secret key sk_i. Note that for each time interval t, Aggregator $\mathcal{A}$ is allowed to query oracle $\mathcal{O}_{\mathsf{EncTag}}$ for user $\mathcal{U}_i$ only once. In other words, Aggregator $\mathcal{A}$ cannot submit two distinct queries to oracle $\mathcal{O}_{\mathsf{EncTag}}$ with

Algorithm 7 Learning phase of the aggregate unforgeability game

$\mathcal{P}, \mathsf{vk} \leftarrow \mathcal{O}_{\mathsf{Setup}}(1^\lambda)$; // $\mathcal{A}$ executes the following a polynomial number of times

$(c_{i,t}, \sigma_{i,t}) \leftarrow \mathcal{O}_{\mathsf{EncTag}}(t, \mathsf{uid}_i, x_{i,t})$;

Algorithm 8 Challenge phase of the aggregate unforgeability game

$(t^*, \mathsf{sum}_{t^*}, \sigma_{t^*}) \leftarrow \mathcal{A}$;

the same time interval t and the same user identifier uid_i. Without loss of generality, we suppose that for each time interval t, Aggregator $\mathcal{A}$ invokes oracle $\mathcal{O}_{\mathsf{EncTag}}$ for all users $\mathcal{U}_i$ in the system.

At the end of the *aggregate unforgeability* game (see Algorithm 8), Aggregator $\mathcal{A}$ outputs a tuple $(t^*, \mathsf{sum}_{t^*}, \sigma_{t^*})$. We say that Aggregator $\mathcal{A}$ wins the *aggregate unforgeability* game if one of the following statements holds:

1. $\mathsf{Verify}(\mathsf{vk}, \mathsf{sum}_{t^*}, \sigma_{t^*}) \rightarrow 1$ and Aggregator $\mathcal{A}$ never made a query to oracle $\mathcal{O}_{\mathsf{EncTag}}$ that comprises time interval t^*. In the remainder of this Chapter, we denote this type of forgery **Type I Forgery**.

2. $\mathsf{Verify}(\mathsf{vk}, \mathsf{sum}_{t^*}, \sigma_{t^*}) \rightarrow 1$ and Aggregator $\mathcal{A}$ has made a query to oracle $\mathcal{O}_{\mathsf{EncTag}}$ for time t^*, however the sum $\mathsf{sum}_{t^*} \neq \sum_{\mathcal{U}_i} x_{i,t^*}$. In what follows, we call this type of forgery **Type II Forgery**.

Definition 44 (*Aggregate Unforgeability*)**.** *Let* $\Pr[\mathcal{A}^{\mathbf{AU}}]$ *denote the probability that Aggregator* $\mathcal{A}$ *wins the* aggregate unforgeability *game, that is, the probability that Aggregator* $\mathcal{A}$ *outputs a* **Type I Forgery** *or* **Type II Forgery** *that will be accepted by algorithm* Verify.

An aggregation protocol is said to ensure aggregate unforgeability if for any polynomially bounded adversary $\mathcal{A}$, $\Pr[\mathcal{A}^{\mathbf{AU}}] \leqslant \epsilon(\lambda)$, *where* ϵ *is a negligible function in the security parameter* λ.

7.4 Idea of our PUDA protocol

In an extended model with an untrusted Aggregator, it is of utmost importance to design a solution in which the untrusted Aggregator cannot provide bogus results to the Data Analyzer.

Such a solution will use a proof system that enables the Data Analyzer to verify the correctness of the computation. Yet verifiability should be achieved without sacrificing privacy. Towards this goal, we propose a protocol that relies on the following techniques:

- A *homomorphic encryption* algorithm that allows the Aggregator to compute the sum without divulging individual data.
- A *homomorphic tag* that allows each user to authenticate the data input $x_{i,t}$, in such a way that the Aggregator can use the collected tags to construct a proof that demonstrates to the Data Analyzer $\mathcal{DA}$ the correctness of the aggregate sum.

Concisely, a set of non-interacting users are connected to personal services and devices that produce personal data. Without any coordination, each user chooses a random tag key $\mathsf{tk_i}$ and sends an encoding thereof, $\overline{\mathsf{tk_i}}$ to the key dealer. After collecting all encoded keys $\overline{\mathsf{tk_i}}$ by users, the key dealer publishes the public verification key vk of this group of users. This verification key is computed as a function of the encodings $\overline{\mathsf{tk_i}}$. Later, the key dealer gives to each user in the system an encryption key $\mathsf{ek_i}$ that will be used to compute the user's ciphertexts. Accordingly, the secret key of each user sk_i is defined as the pair of tag key tk_i and encryption key ek_i. Finally, the key dealer provides the Aggregator with secret key sk_A computed as the sum of encryption keys ek_i and goes off-line.

Now at each time interval t, each user employs its secret key $\mathsf{sk_i}$ to compute a ciphertext based on the encryption algorithm of Shi *et al.* [132] and a homomorphic tag on its sensitive data input. When the Aggregator collects the ciphertexts and the tags from all users, it computes the sum $\mathsf{sum_t}$ of users' data and a proof σ_t for the sum, and forwards the sum and the proof to the Data Analyzer. At the final step of the protocol, the Data Analyzer verifies with the verification key vk and proof σ_t the validity of the result $\mathsf{sum_t}$. Although the modification seems straightforward, the proof for **Type II Forgery** turns out to be challenging.

Thanks to the homomorphic encryption algorithm of Shi *et al.* [132] and the way in which we construct our homomorphic tags, we show that our protocol ensures *Aggregator obliviousness.* Moreover, we show that the Aggregator cannot forge bogus results. Finally, we note that the

Data Analyzer $\mathcal{DA}$ does not keep any state with respect to users' transcripts be they ciphertexts or tags, but it only holds the public verification key, the sum $\mathsf{sum_t}$ and the proof σ_t.

7.5 PUDA Instantiation

For encryption and sum computation we employ the *discrete logarithm* based encryption scheme of Shi *et al.* [132]:

7.5.1 Shi-Chan-Rieffel-Chow-Song Scheme

- **Setup**(1^κ): Let $\mathbb{G}_1$ be a group of large prime order p. A trusted key dealer $\mathcal{KD}$ selects a hash function $H : \{0,1\}^* \rightarrow \mathbb{G}_1$. Furthermore, $\mathcal{KD}$ selects secret encryption keys $\mathsf{ek}_i \in \mathbb{Z}_p$, uniformly at random. $\mathcal{KD}$ distributes to each user $\mathcal{U}_i$ the secret key ek_i and it also sends the secret key $\mathsf{sk_A} = -\sum_{i=1}^{n} \mathsf{ek}_i$ to the Aggregator.
- **Encrypt**$(\mathsf{ek}_i, x_{i,t})$: Each user $\mathcal{U}_i$ encrypts the value $x_{i,t}$ by using its secret encryption key ek_i and outputs the corresponding ciphertext $c_{i,t} = H(t)^{\mathsf{ek}_i} g_1^{x_{i,t}} \in \mathbb{G}_1$.
- **Aggregate**$(\{c_{i,t}\}_{\mathcal{U}_i \in \mathbb{U}}, \{\sigma_{i,t}\}_{\mathcal{U}_i \in \mathbb{U}}, \mathsf{sk_A})$: Upon receiving all the ciphertexts $\{c_{i,t}\}_{i=1}^{n}$, the Aggregator computes: $V_t = (\prod_{i=1}^{n} c_{i,t}) H(t)^{\mathsf{sk}_A} = H(t)^{\sum_{i=1}^{n} \mathsf{ek}_i} g_1^{\sum_{i=1}^{n} x_{i,t}} H(t)^{-\sum_{i=1}^{n} \mathsf{ek}_i} = g_1^{\sum_{i=1}^{n} x_{i,t}} \in \mathbb{G}_1$. Finally $\mathcal{A}$ learns the sum $\mathsf{sum_t} = \sum_{i=1}^{n} x_{i,t} \in \mathbb{Z}_p$ by computing the discrete logarithm of V_t on the base g_1. The sum computation is correct as long as $\sum_{i=1}^{n} x_{i,t} < p$.

7.5.2 PUDA Scheme

In what follows we describe our **PUDA** protocol:

- **Setup**(1^κ): $\mathcal{KD}$ outputs $(p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ for an efficient computable bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, where g_1 and g_2 are two random generators for the multiplicative groups $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively and p is a prime number that denotes the order of all the groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$. Moreover a secret key a is selected by $\mathcal{KD}$. Each $\mathcal{U}_i$ selects a random tag key $\mathsf{tk}_i \in \mathbb{Z}_p$ independently and forwards $g_2^{\mathsf{tk_i}}$ to $\mathcal{KD}$. $\mathcal{KD}$ publishes the verification key $\mathsf{vk} = (\mathsf{vk}_1, \mathsf{vk}_2) = (g_2^{\sum_{i=1}^{n} \mathsf{tk}_i}, g_2^{a})$ and distributes to each user $\mathcal{U}_i \in \mathbb{U}$ the secret key $g_1^a \in \mathbb{G}_1$ through a secure channel. Thus the secret keys of the scheme are $\mathsf{sk}_i = (\mathsf{ek}_i, \mathsf{tk}_i, g_1^a)$. After publishing the public parameters $\mathcal{P} = (H, p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ and the verification key vk, $\mathcal{KD}$ goes off-line and it does not further participate in any protocol phase.

- **EncTag**$(t, \mathsf{sk}_i = (\mathsf{ek}_i, \mathsf{tk}_i, g_1^a), x_{i,t})$: At each time interval t each user $\mathcal{U}_i$ encrypts the data value $x_{i,t}$ with its secret encryption key ek_i, using the encryption algorithm, described in section 7.5.1, which results in a ciphertext

$$c_{i,t} = H(t)^{\mathsf{ek}_i} g_1^{x_{i,t}} \in \mathbb{G}_1$$

  $\mathcal{U}_i$ also constructs a tag $\sigma_{i,t} \in \mathbb{G}_1$ with its secret tag key (tk_i, g_1^a):

$$\sigma_{i,t} = H(t)^{\mathsf{tk}_i} (g_1^a)^{x_{i,t}} \in \mathbb{G}_1$$

  Finally $\mathcal{U}_i$ sends $(c_{i,t}, \sigma_{i,t})$ to $\mathcal{A}$.

- **Aggregate**$(\mathsf{sk_A}, \{c_{i,t}\}_{\mathcal{U}_i \in \mathbb{U}}, \{\sigma_{i,t}\}_{\mathcal{U}_i \in \mathbb{U}})$: Aggregator $\mathcal{A}$ computes the sum $\mathsf{sum_t} = \sum_{i=1}^{n} x_{i,t}$ by using the **Aggregate** algorithm presented in section 7.5.1.

  Moreover, $\mathcal{A}$ aggregates the corresponding tags as follows:

$$\sigma_\mathsf{t} = \prod_{i=1}^{n} \sigma_{i,t} = \prod_{i=1}^{n} H(t)^{\mathsf{tk}_i} (g_1^a)^{x_{i,t}} = H(t)^{\sum \mathsf{tk}_i} (g_1^a)^{\sum x_{i,t}}$$

  $\mathcal{A}$ finally forwards $\mathsf{sum_t}$ and σ_t to data analyzer $\mathcal{DA}$.

- **Verify**$(\mathsf{vk}, \mathsf{sum_t}, \sigma_\mathsf{t})$: During the verification phase $\mathcal{DA}$ verifies the correctness of the computation with the verification key $\mathsf{vk} = (\mathsf{vk}_1 = g_2^{\sum \mathsf{tk}_i}, \mathsf{vk}_2 = g_2^a)$, by checking the following equality:

$$e(\sigma_\mathsf{t}, g_2) \stackrel{?}{=} e(H(t), \mathsf{vk}_1) e(g_1^{\mathsf{sum_t}}, \mathsf{vk}_2)$$

Verification correctness follows from bilinear pairing properties:

$$
\begin{aligned}
e(\sigma_{\mathsf{t}}, g_2) &= e(\prod_{i=1}^{n} \sigma_{i,t}, g_2) = e(\prod_{i=1}^{n} H(t)^{\mathsf{tk}_i} g_1^{a x_{i,t}}, g_2) = e(H(t)^{\sum_{i=1}^{n} \mathsf{tk}_i} g_1^{a \sum_{i=1}^{n} x_{i,t}}, g_2) \\
&= e(H(t)^{\sum_{i=1}^{n} \mathsf{tk}_i}, g_2) e(g_1^{a \sum_{i=1}^{n} x_{i,t}}, g_2) = e(H(t), g_2^{\sum_{i=1}^{n} \mathsf{tk}_i}) e(g_1^{\mathsf{sum_t}}, g_2^{a}) \\
&= e(H(t), \mathsf{vk}_1) e(g_1^{\mathsf{sum_t}}, \mathsf{vk}_2)
\end{aligned}
$$

7.6 Analysis

7.6.1 Aggregator Obliviousness

Theorem 7. *The proposed solution achieves Aggregator obliviousness in the random oracle model under the decisional Diffie-Hellman (DDH) assumption in* $\mathbb{G}_1$.

Proof. Assume there is an Aggregator $\mathcal{A}$ which breaks the obliviousness of the **PUDA** scheme with a non-negligible advantage ϵ. We build in what follows an adversary $\mathcal{B}$, which uses $\mathcal{A}$ as a subroutine to break the Aggregator obliviousness of the private streaming aggregation (PSA) protocol presented in [132], which is guaranteed under DDH. Without loss of generality we call the oracles that the adversary $\mathcal{B}$ has access to from the PSA scheme as follows: $\mathcal{O}^{\mathsf{PSA}}_{\mathsf{Setup}}$, $\mathcal{O}^{\mathsf{PSA}}_{\mathsf{Corrupt}}$, $\mathcal{O}^{\mathsf{PSA}}_{\mathsf{Encrypt}}$, and $\mathcal{O}^{\mathsf{PSA}}_{\mathsf{AO}}$.

We consider in PSA as in **PUDA** that there are n users $\mathcal{U}_i$ and each one of these users possesses a secret encryption key $\mathsf{ek_i}$. In the following, we show how an adversary $\mathcal{B}$ simulates the Aggregator obliviousness game presented in Algorithms 5 and 6 to Aggregator $\mathcal{A}$ and how therewith breaks the Aggregator obliviousness of PSA.

Learning phase: In the learning phase, adversary $\mathcal{B}$ proceeds as following: Whenever $\mathcal{A}$ calls oracle $\mathcal{O}_{\mathsf{Setup}}$ with a security parameter κ, $\mathcal{B}$ queries oracle $\mathcal{O}^{\mathsf{PSA}}_{\mathsf{Setup}}$ with the same security parameter. Oracle $\mathcal{O}^{\mathsf{PSA}}_{\mathsf{Setup}}$ in turn outputs the public parameters that are composed of a hash function $H : \{0,1\}^* \rightarrow \mathbb{G}_1$, a generator g_1 of the group $\mathbb{G}_1$ of large prime order p, and the Aggregator's secret key $\mathsf{sk_A} = -\sum_{\mathsf{i=1}}^{\mathsf{n}} \mathsf{ek_i}$. $\mathcal{B}$ then selects the parameters of a bilinear pairing $(e, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$. $\mathcal{B}$ chooses uniformly at random $a, \{r_i\}_{\mathcal{U}_i \in \mathcal{U}}$ and defines the verification key vk as follows:

$$\mathsf{vk} = (g_2^{a\mathsf{sk_A}+\sum_{i=1}^{n} r_i}, g_2^a) = (g_2^{a\sum_{i=1}^{n}\mathsf{ek_i}+\sum_{i=1}^{n} r_i}, g_2^a) = (g_2^{\sum_{i=1}^{n} a\mathsf{ek}_i + r_i}, g_2^a)$$

This entails that tk_i is defined as: $a\mathsf{ek}_i + r_i$. Finally $\mathcal{B}$ forwards to $\mathcal{A}$ the public parameters: $\mathcal{P} = (H, p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$, the verification keys $\mathsf{vk} = (g_2^{\sum_{i=1}^{n}\mathsf{tk_i}}, g_2^a)$ and the secret key of the Aggregator $\mathsf{sk_A}$.

Whenever $\mathcal{A}$ calls oracle $\mathcal{O}_{\mathsf{Corrupt}}$ with a user's identifier uid_i, $\mathcal{B}$ relays the query uid_i to $\mathcal{O}_{\mathsf{Corrupt}}^{\mathsf{PSA}}$ of the PSA scheme which in turns outputs the secret encryption key $\mathsf{ek_i}$ of user $\mathcal{U}_i$. $\mathcal{B}$ then returns secret key $\mathsf{sk}_i = (\mathsf{ek}_i, \mathsf{tk}_i) = (\mathsf{ek}_i, a\mathsf{ek}_i + r_i)$.

Whenever $\mathcal{A}$ calls oracle $\mathcal{O}_{\mathsf{EncTag}}$ with query $(t, \mathsf{uid}_i, x_{i,t})$, $\mathcal{B}$ forwards the query to the $\mathcal{O}_{\mathsf{Encrypt}}^{\mathsf{PSA}}$ oracle which returns the appropriate ciphertext $c_{i,t} = H(t)^{\mathsf{ek_i}} g_1^{x_{i,t}}$. $\mathcal{B}$ computes then the tag associated with ciphertext $c_{i,t}$ as $\sigma_{i,t} = (c_{i,t})^a H(t)^{r_i} = H(t)^{a\mathsf{ek_i}+\mathsf{r_i}} g_1^{ax_{i,t}} = H(t)^{\mathsf{tk}_i} g_1^{ax_{i,t}}$ and transmits to $\mathcal{A}$ ciphertext $c_{i,t}$ and tag $\sigma_{i,t}$.

Challenge phase: In the challenge phase $\mathcal{A}$ chooses a set of users $\mathbb{S}^*$ that have not been corrupted during the learning phase and a time interval t^* for which $\mathcal{A}$ did not make a query to oracle $\mathcal{O}_{\mathsf{EncTag}}$. $\mathcal{A}$ then submits two time-series $\mathcal{X}_0^* = (\mathcal{U}_i, t^*, x_{i,t^*}^0)_{\mathcal{U}_i \in \mathbb{S}^*}$ and $\mathcal{X}_1^* = (\mathcal{U}_i, t^*, x_{i,t^*}^1)_{\mathcal{U}_i \in \mathbb{S}^*}$ to $\mathcal{O}_{\mathsf{AO}}$, such that $\sum x_{i,t^*}^0 = \sum x_{i,t^*}^1$. $\mathcal{B}$ simulates this oracle as follows:

It forwards the series $\mathcal{X}_0^*$ and $\mathcal{X}_1^*$ to $\mathcal{O}_{\mathsf{AO}}^{\mathsf{PSA}}$ which chooses uniformly at random a bit $b \xleftarrow{\$} \{0,1\}$ and returns to $\mathcal{B}$ the ciphertexts $\{c_{i,t^*}^b\}_{\mathcal{U}_i \in \mathbb{S}^*}$ encrypting $\mathcal{X}_b^*$.

Next, $\mathcal{B}$ constructs for all $\mathcal{U}_i$ in $\mathbb{S}^*$ the tag σ_{i,t^*}^b corresponding to ciphertext c_{i,t^*}^b by computing:

$$\sigma_{i,t^*}^b = (c_{i,t}^b)^a H(t^*)^{r_i} = (H(t^*)^{\mathsf{ek}_i} g_1^{x_{i,t^*}^b})^a H(t^*)^{r_i} = H(t^*)^{a\mathsf{ek}_i + r_i} g_1^{ax_{i,t^*}^b} = H(t^*)^{\mathsf{tk}_i} g_1^{ax_{i,t^*}^b}$$

Note that σ_{i,t^*}^b corresponds to a correctly computed tag for input x_{i,t^*}^b. Finally, $\mathcal{B}$ forwards $\{c_{i,t^*}^b, \sigma_{i,t^*}^b\}_{\mathcal{U}_i \in \mathbb{S}^*}$ to $\mathcal{A}$. At this point, the simulated view of Aggregator $\mathcal{A}$ is computationally indistinguishable from its view in an actual *Aggregator obliviousness* game as defined in Algorithms 5 and 6. This leads to correct verification of the sum computed by $\mathcal{A}$, more precisely:

$$
\begin{aligned}
e(\prod_{i \in \mathbb{S}^*} \sigma_{i,t^*}^b, g_2) &= e(\prod_{i=1}^{n} H(t^*)^{\mathsf{tk}_i} g_1^{a x_{i,t^*}^b}, g_2) = e(H(t^*), g_2^{a \sum_{i=1}^n \mathsf{ek}_i + \sum_{i=1}^n r_i}) e(g_1^{\sum_{i=1}^n x_{i,t^*}^b}, g_2^a) \\
&= e(H(t^*), \mathsf{vk}_1) e(g_1^{\sum_{i=1}^n x_{i,t^*}^b}, \mathsf{vk}_2)
\end{aligned}
$$

It follows that if Aggregator $\mathcal{A}$ is able to output a correct guess b^* for the bit b with a non-negligible advantage ϵ: (i.e. is able to break the Aggregator obliviousness of our scheme), then $\mathcal{B}$ will break the Aggregator obliviousness of the PSA scheme with the same non-negligible advantage ϵ by outputting the guess b^*. $\square$

As such PSA scheme ensures Aggregator obliviousness under the DDH assumption in $\mathbb{G}_1$, we can conclude that our scheme also ensures Aggregator obliviousness: $\Pr[\mathcal{A}^{\mathbf{AO}}] \leqslant \frac{1}{2} + \epsilon(\kappa)$ as long as DDH holds in $\mathbb{G}_1$.

7.6.2 Aggregate Unforgeability

Theorem 8. *Our scheme achieves* Aggregate Unforgeability *for a* **Type I Forgery** *under* **BCDH** *assumption in the random oracle model.*

Proof. We show how to build an adversary $\mathcal{B}$ that solves **BCDH** in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$. Let g_1 and g_2 be two generators for $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. $\mathcal{B}$ receives the challenge $(g_1, g_2, g_1^a, g_1^b, g_1^c, g_2^a, g_2^b)$ from the **BCDH** oracle $\mathcal{O}_{\mathbf{BCDH}}$ and is asked to output $e(g_1, g_2)^{abc} \in \mathbb{G}_T$. $\mathcal{B}$ simulates the interaction with $\mathcal{A}$ in the two phases (**Setup, Learning**) as follows:

Setup:

- To simulate the $\mathcal{O}_{\mathsf{Setup}}^{\mathcal{A}}$ oracle $\mathcal{B}$ selects uniformly at random $2n$ keys $\{\mathsf{k}_i\}_{i=1}^n$, $\{\mathsf{y}_i\}_{i=1}^n \in \mathbb{Z}_p$ and outputs the public parameters $\mathcal{P} = (\kappa, p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2)$ the verification key $\mathsf{vk} = (\mathsf{vk}_1, \mathsf{vk}_2) = (g_2^{b \sum_{i=1}^n \mathsf{k}_i}, g_2^a)$ and the secret key of the Aggregator $\mathsf{sk}_A = -\sum_{i=1}^n \mathsf{y}_i$.

Learning phase

- $\mathcal{A}$ is allowed to query the random oracle H for any time interval . $\mathcal{B}$ constructs a $\mathtt{H-list}$ and responds to $\mathcal{A}$ query as follows:
  1. If query (t) already appears in a tuple H-tuple$\langle t : r_t, \mathsf{coin}(t), H(t)\rangle$ of the $\mathtt{H-list}$ it responds to $\mathcal{A}$ with $H(t)$.
  2. Otherwise it selects a random number $r_t \in \mathbb{Z}_p$ and flips a random $\mathsf{coin} \xleftarrow{\$} \{0, 1\}$. With probability p, $\mathsf{coin(t)} = \mathsf{0}$ and $\mathcal{B}$ answers with $H(t) = g_1^{r_t}$. Otherwise if $\mathsf{coin(t)} = \mathsf{1}$ then $\mathcal{B}$ responds with $H(t) = g_1^{cr_t}$ and updates the $\mathtt{H-list}$ with the new tuple H-tuple$\langle t : r_t, \mathsf{coin}(t), H(t)\rangle$.
- Whenever $\mathcal{A}$ submits a query $(t, \mathsf{uid}_i, x_{i,t})$ to the $\mathcal{O}^{\mathcal{A}}_{\mathsf{EncTag}}$, $\mathcal{B}$ constructs a $\mathtt{T-list}$ and responds as follows:
  1. If at time interval t $\mathcal{A}$ has never queried before the $\mathcal{O}^{\mathcal{A}}_{\mathsf{EncTag}}$ oracle then:
     - 1.1. $\mathcal{B}$ initializes a variable $\Sigma_t = 0$.
     - 1.2. $\mathcal{B}$ calls the simulated random oracle, receives the result for H(t) and appends the tuple H-tuple$\langle t : r_t, \mathsf{coin}(t), H(t)\rangle$ to the $\mathtt{H-list}$.
     - 1.3. If $\mathsf{coin}(t) = 1$ then $\mathcal{B}$ stops the simulation.
     - 1.4. Otherwise it chooses the secret tag key k_i where $i = \mathsf{uid_i}$ to be used as secret tag key from the set of $\{\mathsf{k}_i\}$ keys, chosen by $\mathcal{B}$ in the **Setup** phase.
     - 1.5. $\mathcal{B}$ sends to $\mathcal{A}$ the tag $\sigma_{i,t} = g_1^{r_t bk_i} g_1^{ax_{i,t}} = H(t)^{bk_i} g_1^{ax_{i,t}}$, which is a valid tag for the value $x_{i,t}$. Notice that $\mathcal{B}$ can correctly compute the tag without knowing a and b from the **BCDH** problem parameters g_1^a, g_1^b.
     - 1.6. $\mathcal{B}$ chooses also a secret encryption key $\mathsf{y}_i \in \{\mathsf{y}_i\}_{i=1}^n \in \mathbb{Z}_p$ and computes the ciphertext as $c_{i,t} = H(t)^{\mathsf{y}_i} g_1^{x_{i,t}}$. The simulation is correct since $\mathcal{A}$ can check that the sum $\sum_{i=1}^n x_{i,t}$ corresponds to the ciphertexts given by $\mathcal{B}$ with its decryption key $\mathsf{sk}_A = -\sum_{i=1}^n \mathsf{y_i}$, considering the adversary has made distinct encryption queries for all the n users in the scheme at a time interval t.

1.7. $\mathcal{B}$ sets $\Sigma_t = \Sigma_t + x_{i,t}$ and updates the $\mathtt{T-list}$ with the tuple: $\langle t, \mathsf{uid}_i, x_{i,t}, \sigma_{i,t}\rangle$

2. Else if $\mathtt{T-list}$ contains $i' = \mathsf{uid_i}$ and $x_{i,t} = x'_{i,t}$ then $\mathcal{B}$ fetches the corresponding $\sigma_{i,t}$ from the list and forwards it to $\mathcal{A}$.

3. Else if $\mathtt{T-list}$ contains $i' = \mathsf{uid_i}$ and $x_{i,t} \neq x'_{i,t}$ then $\mathcal{B}$ aborts.

4. Otherwise, $\mathcal{B}$ looks to the $\mathtt{H-list}$ list for the tuple indexed by t in order to get $\langle t : r_t, \mathsf{coin}(t), H(t)\rangle$. If the tuple does not exist then $\mathcal{B}$ tosses a random coin and if $\mathsf{coin}(t) = 1$ then $\mathcal{B}$ aborts. If $\mathsf{coin}(t) = 0$ then $\mathcal{B}$ computes the tag identically as in 1(d)(e)(f)(g) steps: It chooses a key k_i where $i = \mathsf{uid_i}$ from the selected keys $\{\mathsf{k}_i\}$. It constructs the tag as $\sigma_{i,t} = g_1^{r_t b k_i} g_1^{a x_{i,t}} = H(t)^{b k_i} g_1^{a x_{i,t}}$ and the ciphertext as $c_{i,t} = H(t)^{\mathsf{y}_i} g_1^{x_{i,t}}$. Finally $\mathcal{B}$ sets $\Sigma_t = \Sigma_t + x_{i,t}$, updates the $\mathtt{T-list}$ with the tuple: $\langle t, \mathsf{uid}_i, x_{i,t}, \sigma_{i,t}\rangle$.

Now, when $\mathcal{B}$ receives the forgery $(\mathsf{sum_t}^*, \sigma_{\mathsf{t}}{}^*)$ at time interval $t = t^*$, it continues if $\mathsf{sum_t}^* \neq \Sigma_t$. $\mathcal{B}$ first queries the H-tuple for time t^* in order to fetch the appropriate tuple.

- If $\mathsf{coin}(t^*) = 0$ then $\mathcal{B}$ aborts.
- If $\mathsf{coin}(t^*) = 1$ then since $\mathcal{A}$ outputs a valid forged $\sigma_{\mathsf{t}}{}^*$ at t^*, it is true that the following equation should hold:

$$e(\sigma_{\mathsf{t}}{}^*, g_2) = e(H(t^*), \mathsf{vk}_1) e(g_1^{\mathsf{sum_t}^*}, \mathsf{vk}_2)$$

which is true when $\mathcal{A}$ makes n queries for time interval t^* for distinct users to the $\mathcal{O}^{\mathcal{A}}_{\mathsf{EncTag}}$ oracle during the **Learning** phase. As such $\sigma_{\mathsf{t}}{}^* = g_1^{c r_t b \sum \mathsf{k}_i} g_1^{a \mathsf{sum_t}^*}$

Finally $\mathcal{B}$ outputs:

$$e\left(\left(\frac{\sigma_{\mathsf{t}}{}^*}{g_1^{a\mathsf{sum_t}^*}}\right)^{\frac{1}{r_t \sum \mathsf{k}_i}}, g_2^a\right) = e\left(\left(\frac{g_1^{c r_t b \sum \mathsf{k}_i} g_1^{a\mathsf{sum_t}^*}}{g_1^{a\mathsf{sum_t}^*}}\right)^{\frac{1}{r_t \sum \mathsf{k}_i}}, g_2^a\right) =$$
$$e\left(\left(g_1^{c r_t b \sum \mathsf{k}_i}\right)^{\frac{1}{r_t \sum \mathsf{k}_i}}, g_2^a\right) = e(g_1^{bc}, g_2^a) = e(g_1, g_2)^{abc}$$

Let $\mathcal{A}^{\mathbf{AU1}}$ the event when $\mathcal{A}$ successfully forges a **Type I forgery** σ_t for our **PUDA** protocol that happens with some non-negligible probability ϵ'. Then $\Pr[\mathcal{B}^{\mathbf{BCDH}}] = \Pr[\mathtt{event}_0]\Pr[\mathtt{event}_1]\Pr[\mathcal{A}^{\mathbf{AU2}}] = p(1-p)^{\mathtt{q_H}-1}\epsilon'$, for $\mathtt{q_H}$ random oracle queries with the probability $\Pr[\mathsf{coin}(t)=0]=p$. As such we ended up in a contradiction assuming the hardness of the **BCDH** assumption and finally $\Pr[\mathcal{A}^{\mathbf{AU1}}] \leq \epsilon_1$, where ϵ_1 is a negligible function.

□

For the security proof against an adversary $\mathcal{A}$ for **Type-II Forgery** we first introduce a new assumption that is used during the security analysis of our **PUDA** instantiation. Our new assumption named hereafter LEOM is a variant of the LRSW assumption [110] which is proven secure in the generic model [134] and used in the construction of the CL signatures [32].

The oracle $\mathcal{O}_{\mathsf{LEOM}}$ first chooses a and k_i, $1 \leqslant i \leqslant n$ in $\mathbb{Z}_p^*$. Then it publishes the tuple $(g_1, g_2^{\sum_{i=1}^n k_i}, g_2^a)$. Thereafter, the adversary picks $h_t \in \mathbb{G}_1$ and makes queries $(h_t, i, x_{i,t})$ for $1 \leqslant i \leqslant n$ to the $\mathcal{O}_{\mathsf{LEOM}}$ oracle which in turn replies with $h_t^{k_i} g_1^{ax_{i,t}}$ for $1 \leqslant i \leqslant n$.

The adversary is allowed to query the oracle $\mathcal{O}_{\mathsf{LEOM}}$ for different h_t with the restriction that it cannot issue two queries for the same pair (h_t, i).

We say that the adversary breaks the LEOM assumption, if it outputs a tuple $(z, h_t, h_t^{\sum_{i=1}^n k_i} g_1^{az})$ for a previously queried t and $z \neq \sum_{i=1}^n x_{i,t}$.

Theorem 9. *(LEOM Assumption) Given the security parameter κ, the public parameters $(p, e, \mathbb{G}_1, \mathbb{G}_2, g_1, g_2)$, the public key $(g_2^a, g_2^{\sum_{i=1}^n k_i})$ and the oracle $\mathcal{O}_{\mathsf{LEOM}}$, we say that the LEOM assumption holds iff:*

For all probabilistic polynomial time adversaries $\mathcal{A}$, the following holds:

$$\Pr[(z, h_t, \sigma_t) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{LEOM}}(.)} : z \neq \sum_{i=1}^n x_{i,t} \wedge \sigma_t = h_t^{\sum_{i=1}^n k_i} g_1^{az}] \leq \epsilon_2(\kappa)$$

Where ϵ_2 is a negligible function.

The security evidence of the assumption is referred to the Appendix Chapter.

Theorem 10. *Our scheme guarantees aggregate unforgeability against a* **Type II Forgery** *under the* LEOM *assumption in the random oracle model.*

Proof. (Sketch) Here we show how an adversary $\mathcal{B}$ breaks the LEOM assumption by using an Aggregator $\mathcal{A}$ that provides a **Type II Forgery** with a non-negligible probability. Notably, adversary $\mathcal{B}$ simulates oracle $\mathcal{O}_{\mathsf{Setup}}$ as follows: It first picks secret encryptions keys $\{\mathsf{ek}_i\}_{i=1}^n$ and sets the corresponding decryption key $\mathsf{SK}_A = -\sum_{i=1}^n \mathsf{ek}_i$. Then, it forwards to $\mathcal{A}$ the public parameters $\mathcal{P} = (p, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2)$, the public key $(\mathsf{vk}_1, \mathsf{vk}_2) = (g_2^{\sum_{i=1}^n k_i}, g_2^a)$ of the $\mathcal{O}_{\mathsf{LEOM}}$ oracle and the secret key $\mathsf{SK}_A = -\sum_{i=1}^n \mathsf{ek}_i$.

Afterwards, when adversary $\mathcal{B}$ receives a query $(t, \mathsf{uid}_i, x_{i,t})$ for oracle $\mathcal{O}_{\mathsf{EncTag}}$, adversary $\mathcal{B}$ calls oracle $\mathcal{O}_{\mathsf{LEOM}}$ with the pair $(h_t = H(t), i, x_{i,t})$. Oracle $\mathcal{O}_{\mathsf{LEOM}}$ accordingly returns $h_t^{k_i} g_1^{ax_{i,t}}$ and adversary $\mathcal{B}$ outputs $\sigma_{i,t} = h_t^{k_i} g_1^{ax_{i,t}}$. Note that if we define the tag key tk_i of user $\mathcal{U}_i$ as k_i, then the tag $\sigma_{i,t} = h_t^{k_i} g_1^{ax_{i,t}}$ is computed correctly.

Eventually with a non-negligible advantage, Aggregator $\mathcal{A}$ outputs a **Type II Forgery** $(t^*, \mathsf{sum}_{t^*}, \sigma_{t^*})$ that verifies:

$$e(\sigma_{t^*}, g_2) = e(H(t^*), \mathsf{vk}_1) e(g_1^{\mathsf{sum}_{t^*}}, \mathsf{vk}_2)$$

where t^* is previously queried by Aggregator $\mathcal{A}$ and $\mathsf{sum}_{t^*} \neq \sum_{i=1}^n x_{(i,t^*)}$.

It follows that $\mathcal{B}$ breaks the LEOM assumption with a non-negligible probability by outputting the tuple $(H(t^*), \mathsf{sum}_{t^*}, \sigma_{t^*})$. This leads to a contradiction under the LEOM assumption. We conclude that our scheme guarantees *aggregate unforgeability* for a **Type II Forgery** under the LEOM assumption. □

To conclude with the analysis the success probabilities for the *aggregate unforgeability* game $\Pr[\mathcal{A}^{\mathbf{AU}}]$, are taken over the union of the success probabilities for the two type of forgeries. As such

$$\Pr[\mathcal{A}^{\mathbf{AU}}] = \Pr[\mathcal{A}^{\mathbf{AU1}}] + \Pr[\mathcal{A}^{\mathbf{AU2}}] \leq \epsilon_1(\kappa) + \epsilon_2(\kappa)$$

where ϵ_1 and ϵ_2 are negligible functions.

Participant	Computation	Communication
User	$\mathbf{2}\,\mathsf{EXP} + \mathbf{1}\,\mathsf{MULT}$	$\mathbf{2 \cdot l}$
Aggregator	$(\mathbf{n-1})\,\mathsf{MULT}$	$\mathbf{2 \cdot l}$
Data Analyzer	$\mathbf{3}\,\mathsf{PAIR} + \mathbf{1}\,\mathsf{EXP} + \mathbf{1}\,\mathsf{MULT} + \mathbf{1}\mathsf{H}ASH$	-

Table 7.1: Performance of tag computation, proof construction and verification operations. l denotes the bit-size of the prime number p.

7.6.3 Performance Evaluation

In this section we analyze the extra overhead of ensuring the *aggregate unforgeability* property in our **PUDA** instantiation scheme. First, we consider a theoretical evaluation with respect to the mathematical operations a participant of the protocol be it user, Aggregator or Data Analyzer has to perform with respect to the verifiability transcripts. That is, the computation of the tag by each user, the proof by the Aggregator and the verification of the proof by the Data Analyzer. We also present an experimental evaluation that shows the practicality of out scheme.

To allow the Data analyzer to verify the correctness of computations performed by an untrusted Aggregator each user selects uniformly and at random a secret key $\mathsf{tk}_i \in \mathbb{Z}_p$. The key dealer distributes to each user $g_1^a \in \mathbb{G}_1$ and publishes $g_2^a \in \mathbb{G}_2$, which calls for two exponentiations: one in $\mathbb{G}_1$ and one in $\mathbb{G}_2$. At each time interval t each user computes $\sigma_{i,t} = H(t)^{\mathsf{tk}_i}(g_1^a)^{x_{i,t}} \in \mathbb{G}_1$, which entails two exponentiations and one multiplication in $\mathbb{G}_1$. For the computation of the σ_t the Aggregator is involved in $n-1$ multiplications in $\mathbb{G}_1 : \prod_{i=1}^n \sigma_{\mathsf{i,t}}$. Finally the data analyzer verifies by checking the equality: $e(\sigma_\mathsf{t}, g_2) \stackrel{?}{=} e(H(t), \mathsf{vk}_1)e(g_1^{\mathsf{sum_t}}, \mathsf{vk}_2)$, which asks for three pairing evaluations, one hash in $\mathbb{G}_1$, one exponentiation in $\mathbb{G}_1$ and one multiplication in $\mathbb{G}_T$ (see table 7.1). The efficiency of **PUDA** stems from the constant time verification with respect to the size of the users. This is of crucial importance since the Data Analyzer may not own computational power.

We implemented the verification functionalities of **PUDA** with the `Charm` cryptographic framework [9, 10]. For pairing computations, it inherits the `PBC` [109] library which is also written in `C`. All of our benchmarks are executed on Intel Core i5 CPU M 560 @ 2.67GHz × 4 with 8GB of memory, running Ubuntu 12.04 32bit. `Charm` uses 3 types of asymmetric pairings:

Operation \ Pairings	MNT159	MNT201	MNT224
Tag	**1.2** μs	**1.8** μs	**2.2** μs
Verify	**28.3** μs	**42.7** μs	**53.5** μs

Table 7.2: Computational cost of **PUDA** operations with respect to different pairings.

Op. \ Curve	MNT159	MNT201	MNT224
HASH in $\mathbb{G}_1$	**0.139** μs	**0.346** μs	**0.296** μs
HASH in $\mathbb{G}_2$	**25.667** μs	**41.628** μs	**48.305** μs
MULT in $\mathbb{G}_1$	**0.004** μs	**0.0006** μs	**0.006** μs
MULT in $\mathbb{G}_2$	**0.040** μs	**0.051** μs	**0.054** μs
MULT in $\mathbb{G}_T$	**0.012** μs	**0.015** μs	**0.016** μs
EXP in $\mathbb{G}_1$	**0.072** μs	**0.092** μs	**0.099** μs
EXP in $\mathbb{G}_2$	**0.615** μs	**0.757** μs	**0.784** μs
PAIR	**7.077** μs	**10.674** μs	**13.105** μs

Table 7.3: Average computation overhead of the underlying mathematical group operations for different type of curves.

`MNT159`, `MNT201`, `MNT224`. We run our benchmarks with these three different types of asymmetric pairings. The timings for all the underlying mathematical group operations are summarized in table 7.3. There is a vast difference on the computation time of operations between $\mathbb{G}_1$ and $\mathbb{G}_2$ for all the different curves. The reason is the fact that the bit-length of elements in $\mathbb{G}_2$ is much larger than in $\mathbb{G}_1$.

As shown in table 7.2, the computation of tags $\sigma_{i,t}$ implies a computation overhead at a scale of milliseconds with a gradual increase as the bit size of the underlying elliptic curve increases. The data analyzer is involved in pairing evaluations and computations at the target group independent of the size of the data-users.

7.7 Summary

In this Chapter, we designed and analyzed a protocol for private and unforgeable aggregation. First we modeled the security and privacy requirements. In this setting, a set of trustworthy users submits data coupled with unforgeable tags. The purpose of the protocol is to allow a data analyzer to verify the correctness of computation performed by a malicious Aggregator, without discovering the underlying data. The challenge of the verification in aggregation protocols that we tackled with the PUDA protocol is the fact that the privacy from the authentication tags is guaranteed against a malicious Aggregator. Our PUDA instantiation allows for public verification in constant time and is provably secure under the DDH, BCDH and the new LEOM assumption in bilinear pairing groups in the random oracle model.

Chapter 8

Concluding Remarks and Future Research

Contents

8.1 Summary

Privacy Preserving Data Collection and Analysis protocols significantly contribute in decision making. The aggregation of data allows Aggregators to infer useful statistical information that contributes to the social welfare. However, due to the nature of the personal sensitive information that each user outsources to an untrusted party, users are reluctant to reveal their data values in cleartext. Current solutions propose different privacy preserving mechanisms such that users' privacy is not compromised but at the same time an untrusted party learns a statistical function f over the entire population of users.

In this dissertation, we first defined what a Privacy Preserving Data Collection and Analysis protocol (**PPDCA**) is and we presented the state-of-the-art of **PPDCA** protocols. We started our analysis with noise-based techniques, in which each user adds noise to the data value such that an untrusted Aggregator can infer noisy-statistics for the entire population of the users. Noise-based techniques are restricted to provide noisy statistics and therefore, they are not suitable for use case scenarios in which there is necessity for precision in the final result of the statistical function f. Cryptographic protocols aim to address the need for precision in the computation of f. Users encrypt their data appropriately, so as to allow partial access control over an aggregate value. After presenting current cryptographic protocols for **PPDCA**, we proceeded into a detailed taxonomy of the cryptographic protocols in the existing literature based on different characteristics thereof. With our analysis, we identified a gap in the following directions:

- Existing protocols are focused on a restricted family of functions f an Aggregator can learn, such as the sum, inner product and boolean operations.
- The majority of current cryptographic solutions assume the existence of a fully trusted key dealer, which distributes secret keys to the users and to the Aggregator. The consequences of a fully trusted key dealer hinder the deployment of the protocols in a dynamic environment. Namely, within a dynamic environment users are joining and leaving at every single execution of the protocol, thus forcing existing users to receive new secret keys by

the trusted key dealer. Moreover, this single point of trust, renders the protocols fault intolerant, since the trusted key dealer needs to distribute new keys to all existing users in case of a fault.

- There are no solutions supporting a stronger security model in Privacy Preserving Data Collection and Analysis protocols. Current protocols either assume the existence of a fully trusted Aggregator or they base their security on a *honest-but-curious* model in which the Aggregator is trusted to execute the steps of the protocol correctly, but is curious in learning any exchanged messages.

The aforementioned observations led us to the following results of this dissertation:

In Chapter 4, we presented our solution for privacy preserving clustering. Namely, in this scenario an untrusted party wants to learn the degree of similarity of two users' data in order to perform clustering. Our solution entails a novel approach in which data are first transformed to a set of bi-vectors and users encrypt the set of bi-vectors such that an untrusted Aggregator can compare two vectors with the cosine similarity metric. Out technique assures:

- **Oblivious similarity detection**: An untrusted party can perform clustering on encrypted data without learning individual data inputs.
- **Provably secure**: Our protocol is provably secure in the standard model.

In Chapter 5, we analyzed a protocol for a smart grid scenario. Users encrypt their energy consumptions and they send them to an energy supplier who acts as an Aggregator. The latter is interested in learning the time intervals in which users consumed the maximum, for energy utility awareness and energy forecasting. Our technique is based on an order preserving encryption scheme, combined with a delta encoding function in order to eliminate short spontaneous spikes in the energy consumption graph which are not continuous. Our protocol guarantees:

- **Aggregator obliviousness**: An untrusted Aggregator does not learn any individual inputs but only the final result, which is the time interval in which a user consumed the maximum.

- **Continuous maximum consumption**: A user in a home may switch on and switch off immediately a high energy appliance. This will contribute to faulty results for the Aggregator, since a spontaneous usage of an energy appliance for a very short period will wrongfully dominate the aggregate statistics. We were able to capture this information with a delta encoding function that allows an Aggregator to discern if the differences of plaintext energy consumptions around a time window, converge to zero, which is interpreted as a continuous maximum energy consumption.

- **Provably secure**: Our protocol is provably secure.

In Chapter 6, we presented a protocol which is suitable for a dynamic population of users. Namely, dynamic leaves and joins of users do not increase the computational and communication overhead of existing users. Moreover, we relax the requirement for a fully trusted dealer. The core idea of the scheme is that each user independently and without any coordination, chooses its secret encryption key. Then, users obfuscate the secret keys with some public information provided by the untrusted Aggregator, and they send the obfuscated keys to a semi-trusted Collector. Finally, the Aggregator learns the sum of the values without compromising individual privacy. In this way our protocol assures:

- **Aggregator & Collector obliviousness**: The protocol provides Aggregator and Collector obliviousness, which assures that individual privacy is not compromised neither by the Aggregator nor by the Collector, which helps the Aggregator in the computation of the sum.

- **Dynamicity & Fault tolerance**: The Aggregator learns the sum even in case of a fault due to a communication error. Moreover the protocol is dynamic in the sense that dynamic leaves and joins do not affect the existing users of the protocol, since there is no need to proceed in a new key distribution phase.

- **Provably secure**: Out protocol is provably secure in the random oracle model under the intractability of well known mathematical problems against *honest-but-curious* Aggregators.

We then considered the problem of verifiability of aggregation in Chapter 7. In this case we strengthen the security requirements of the protocol with a malicious Aggregator, which can deviate from the protocol rules and provide faulty results. The Aggregator along with the result of f which is the sum of the values, computes a proof that allows anyone to verify the correctness of computations. Throughout our analysis we made the following contributions:

- **Aggregate unforgeability**: A malicious Aggregator cannot convince a Data Analyzer with erroneous aggregation with non negligible probability.
- **Obliviousness**: Individual privacy is preserved against untrusted parties of the protocol, while the Aggregator is able to learn the sum of the data inputs.
- **Constant time public verification**: The running time of the verification algorithm is constant and independent on the number of the users. Moreover, the construction allows for public verification of the correctness of the result with a public verification key.
- **Provably secure**: Our protocol is provably secure under a new mathematical assumption whose security evidence is shown in the generic group model.

8.2 Future Work

In this section we present possible directions for future research that stems from the results of this dissertation.

- **Multi-User Order Preserving Encryption**: The protocol presented in Chapter 5 is suitable for single user statistics. A possible line of extension would be to design and analyze an order preserving encryption scheme that takes as input encrypted data by different users and outputs ciphertexts which preserve their plaintext space ordering. The design of such a scheme is challenging since any public key order preserving encryption scheme would be insecure: An adversary which is able to encrypt with the public key of an order preserving encryption scheme can mount an attack based on the result of the encryption algorithm: Adversary $\mathcal{A}$ seeks to learn the plaintext from a ciphertext c. $\mathcal{A}$ chooses a random value r and encrypts it with an OPE scheme, that results in ciphertext c'_r. If the $c'_r > c$ then $\mathcal{A}$ chooses $r' < r$ and encrypts again. Else $\mathcal{A}$ chooses $r' > r$ and continues until it finds a match $c'_r = c$. The search is not exhaustive on the plaintext space but logarithmic. As such a multi-user order preserving encryption scheme needs further investigation.

- **Homomorphic Group Signatures**: The protocol for verification of the aggregate computation presented in Chapter 7 assumes the existence of trustworthy users. That is, users will not try to forge others' tags. In a setting with untrusted users, whereby users have incentives to falsely accuse other users, our protocol falls short to provide the necessary security guarantees. While group signatures can trace users' signatures and provide sender's anonymity, in case of **PPDCA** protocols–in which there must be some homomorphism on the signatures, such that an Aggregator can compute a signature for a function over the received group signatures–group signatures fail to provide a solution. Therefore, a further investigation on how homomorphic group signatures can be realized is a well worth endeavor.

- **Selective Aggregation**: The protocols in Chapters 6 and 7 compute the sum of the users obliviously. However, in real world scenarios, statistics often need to be computed over a sample of a population in order to extract the outliers. A trivial solution would be a Data Analyzer to publish a predicate for the outliers. Then, each user based on the satisfiability of the predicate on input their value, contributes with its encrypted value during the collection phase or excludes itself from sending data. This motivates mistrustful users to lie about their data values. Moreover, the publication of the predicate increases the leakage information a malicious Aggregator learns. Studying the design of a protocol with a dual privacy guarantee, which assures that the predicate is not learnt by the users and the Aggregator aggregates only the values that satisfy the predicate, seems a particularly interesting challenge.

- **Indistinguishability obfuscation**: A long standing problem in computer science has been put forward recently [69]. Namely, the primitive of *indistinguishability obfuscation* guarantees that by obfuscating a program f in f_1 and f_2, an adversary with full power over a machine which executes the circuits that realize the functionality f with f_1 and f_2, cannot distinguish the execution of f_1 from f_2. An adversary can correctly execute an indistinguishable obfuscated program that takes as input the secret key of the user without learning the secret key. This observation has led to the transformation of public key protocols in symmetric protocols, while keeping the secret key private to an adversary [46,81]. The new primitive can lend more security in **PPDCA** protocols. The assumption of non-collusion between an Aggregator and a Collector for the protocol in Chapter 6 can be mitigated as follows: Instead of $\mathcal{A}$ asking for the aggregation of auxiliary information from the Collector, each user along with the ciphertext generates an indistinguishable obfuscated program that takes as input the auxiliary information $\mathsf{aux}_{i,t}$ and outputs the aggregation of $\mathsf{aux}_{i,t}$ only when all indistinguishable obfuscated programs are combined. Otherwise it outputs gibberish data. The construction of joint indistinguishable obfuscated programs that output a correct value only when they are combined appropriately will make great strides towards stronger security models for **PPDCA** protocols.

Appendices

A Lemma 7

We provide the full proof of the following lemma that has been used in Chapter 6.5.3.2:

Lemma 7. *In the random oracle model, collector $\mathcal{C}$ cannot detect that pk_{A,t^*} and $(\mathsf{aux}_{i,t^*})_{\mathcal{U}_i \in \mathbb{S}^*}$ are generated randomly under the decisional composite residuosity (DCR) assumption in $\mathbb{Z}^*_{N^2}$, the quadratic residuosity (QR) assumption in $\mathbb{Z}^*_N$ and the decisional Diffie-Hellman (DDH) assumption in the subgroup of quadratic residues in $\mathbb{Z}^*_N$.*

Let $\mathcal{O}_{\mathsf{DDH}}$ be an oracle which upon a DDH query, first selects randomly g in the subgroup of quadratic residues in $\mathbb{Z}^*_N$ and the pair $(a, b) \in \mathbb{Z}^*_{\phi(N)/4}$ ($\phi(N)$ is the Euler totient of N), then flips a random coin $\beta \in \{0, 1\}$. If $\beta = 0$, then $\mathcal{O}_{\mathsf{DDH}}$ selects c randomly from $\mathbb{Z}^*_{\phi(N)/4}$; otherwise, it sets $c = ab$. Finally, $\mathcal{O}_{\mathsf{DDH}}$ returns the tuple (g, g^a, g^b, g^c).

We say that an adversary $\mathcal{B}$ breaks the DDH assumption in the subgroup of quadratic residues, if it can tell whether $g^c = g^{ab}$ or not.

Proof. (Sketch) Assume there is a collector $\mathcal{C}$ that detects pk_{A,t^*} and $(\mathsf{aux}_{i,t^*})_{\mathcal{U}_i \in \mathbb{S}^*}$ are generated randomly. We show in the random oracle model that there exists an adversary $\mathcal{B}$ that uses collector $\mathcal{C}$ to break DDH in the subgroup of quadratic residues in $\mathbb{Z}^*_N$ under DCR in $\mathbb{Z}^*_{N^2}$ and QR in $\mathbb{Z}^*_N$.

- Let (g, g^a, g^b, g^c) be the DDH tuple provided by $\mathcal{O}_{\mathsf{DDH}}$ to adversary $\mathcal{B}$.
- Let $\mathcal{R}_N$ denote the subgroup of $\mathbb{Z}^*_{N^2}$ defined as $\mathcal{R}_N = \{h^N, h \in \mathbb{Z}^*_{N^2}\}$. We recall that $\mathcal{R}_N$ is of order $\phi(N) = (p-1)(q-1)$ and thus there exists an isomorphism $\rho : \mathbb{Z}^*_N \to \mathcal{R}_N$. Notably, ρ could be defined as: $\forall\ g \in \mathbb{Z}^*_N$, $\rho(g) = g^N \mod N^2$.
- Let $\mathcal{QR}_N$ denote the subgroup of $\mathcal{R}_N$ defined as $\mathcal{QR}_N = \{\tilde{h}^2, \tilde{h} \in \mathcal{R}_N\}$.

Game 0. This game is the collector obliviousness game: Adversary $\mathcal{B}$ executes the setup algorithm by generating the users' secret keys sk_i and the Aggregator $\mathcal{A}$'s secret key sk_A and by publishing the public parameters $\mathcal{P} = (N, H)$, where H is a cryptographic hash function $H : \mathbb{Z} \to \mathbb{Z}^*_{N^2}$. By having the user's secret keys sk_i and $\mathcal{A}$'s secret key sk_A, adversary $\mathcal{B}$ can simulate successfully the collector obliviousness game to collector $\mathcal{C}$.

Game 1. This game is identical to the above game except for the following:

- For each time interval t, $\mathcal{B}$ publishes $\mathsf{pk}_{A,t} = H(t)^{\mathsf{sk}_A N} \in \mathcal{R}_N$ instead of $\mathsf{pk}_{A,t} = H(t)^{\mathsf{sk}_A} \in \mathbb{Z}^*_{N^2}$ (i.e., the Aggregator's secret key is actually $\mathsf{sk}_A N$ instead of sk_A).

- For each time interval t, $\mathcal{B}$ computes $\mathsf{aux}_{i,t} = (H(t)^{\mathsf{sk}_i})^{\mathsf{sk}_A N} \in \mathcal{R}_N$ instead of $\mathsf{aux}_{i,t} = (H(t)^{\mathsf{sk}_i})^{\mathsf{sk}_A} \in \mathbb{Z}^*_{N^2}$.

Note that under the DCR assumption, $\mathcal{C}$ cannot tell whether $\mathsf{pk}_{A,t}$ and $\mathsf{aux}_{i,t}$ are in $\mathcal{R}_N$ or not, and accordingly, Game 0 and Game 1 are computationally indistinguishable.

Game 2. In this game, we compute $\mathsf{pk}_{A,t}$ as $H(t)^{2\mathsf{sk}_A N} \mod N^2$ and $\mathsf{aux}_{i,t}$ as $(H(t)^{\mathsf{sk}_i})^{2\mathsf{sk}_A N} \mod N^2$. Note that under the quadratic residuosity assumption in $\mathbb{Z}^*_N$, Game 1 and Game 2 are computationally indistinguishable. Indeed, if there is a distinguisher $\mathcal{D}$ that is able to tell for instance whether $\mathsf{pk}_{A,t}$ is an element of $\mathcal{QR}_N$ or not, then $\mathcal{D}$ can be used to break the quadratic assumption in $\mathbb{Z}^*_N$ by employing the isomorphism $\rho : \mathbb{Z}^*_N \to \mathcal{R}_N$. Namely, given an element $g \in \mathbb{Z}^*_N$, one computes $\hat{h} = \rho(g)$ and submits $\hat{h}$ to $\mathcal{D}$. If $\mathcal{D}$ outputs that $\hat{h}$ is of the form $\tilde{h}^2$ ($\tilde{h} \in \mathcal{R}_N$), then one outputs that g is quadratic residue in $\mathbb{Z}^*_N$.

Game 3. This game is identical to Game 2 except that this time adversary $\mathcal{B}$ controls a random oracle $\mathcal{H}$ and instead of generating the secret key sk_A randomly in $\mathbb{Z}^*_{N^2}$, it sets $\mathsf{pk}_A = g^a \mod N$ and uses the random oracle to simulate that it possesses the secret key $\mathsf{sk}_A = a$. We recall that (g, g^a, g^b, g^c) is the DDH tuple that adversary $\mathcal{B}$ received from $\mathcal{O}_{\mathsf{DDH}}$.

Without loss of generality, we assume that collector $\mathcal{C}$ makes q hash queries to the random oracle $\mathcal{H}$.

Random Oracle Simulation. To answer the queries of the random oracle $\mathcal{H}$, adversary $\mathcal{B}$ keeps a table T_H of tuples $(t_i, r_i, \mathsf{coin}i(t), H(t_i))$ as explained next. On a query $H(t)$ to $\mathcal{H}$, adversary $\mathcal{B}$ replies as follows:

- If there is a tuple $(t, r, \mathsf{coin}(t), H(t))$ that corresponds to t in table T_H, then $\mathcal{B}$ returns $H(t)$.

- If t has never been queried before, then $\mathcal{B}$ picks a random number $r \in [0, N/4]$, and flips a random coin $\mathsf{coin}(t) \in \{0, 1\}$ such that: $\mathsf{coin}(t) = 1$ with probability p, and it is equal to 0 with probability $1-p$. If $\mathsf{coin}(t)=0$, then $\mathcal{B}$ answers with $H(t)=g^{rN} \mod N^2$. Otherwise, it answers with $H(t)=g^{rbN} \mod N^2$. Finally, adversary $\mathcal{B}$ stores the tuple $(t, r, \mathsf{coin}(t), H(t))$ in table T_H.

Notice that $H(t) \in \mathcal{QR}_N$ instead of being in $\mathbb{Z}^*_{N^2}$, nonetheless collector $\mathcal{C}$ cannot detect this fact thanks to the QR assumption in $\mathbb{Z}^*_N$ and the DCR assumption in $\mathbb{Z}^*_{N^2}$.

Now suppose that $\mathsf{coin}(t^*)=1$, then $H(t^*)$ is of the form g^{r^*bN}. Accordingly, $\mathcal{B}$ simulates the oracle $\mathcal{O}_{\mathsf{CO}}$, by computing $\mathsf{pk}_{A,t^*}=g^{r^*cN} \mod N^2$ and $\mathsf{aux}_{i,t^*}=g^{r^*\mathsf{sk}_i cN} \mod N^2$. Note that in the case where $c=ab$, then $\mathsf{pk}_{A,t^*}=H(t^*)^a=H(t^*)^{\mathsf{sk}_A}$ and $\mathsf{aux}_{i,t^*}=H(t^*)^{\mathsf{sk}_i\mathsf{sk}_A}$, and as a result, collector $\mathcal{C}$ continues the collector obliviousness game. However, if $c \neq ab$ and if $\mathcal{C}$ has a non-negligible advantage ϵ in detecting that pk_{A,t^*} and aux_{i,t^*} are randomly generated, then $\mathcal{C}$ aborts the game with non-negligible advantage ϵ. Therefore, to break the DDH assumption, $\mathcal{B}$ outputs 1 when collector $\mathcal{C}$ continues the collector obliviousness game; and outputs 0 otherwise.

We remark here that the event $\mathsf{coin}(t^*)=1$ occurs with probability $\Pi=p(1-p)^{q-1}$, where q is the number of hash queries that $\mathcal{C}$ issues during the collector obliviousness game. The probability Π is maximal when $p=1/q$ and it equals to $\Pi_{\mathsf{max}} \simeq \frac{1}{eq}$. Therefore, the advantage ϵ' of adversary $\mathcal{B}$ in breaking DDH is equal to $\frac{\epsilon}{eq}$.

□

B LEOM Assumption

In this section we provide security evidence for the hardness of the new LEOM assumption by presenting bounds on the success probabilities of an adversary $\mathcal{A}$, which presumably breaks the assumption. We follow the theoretical *generic group model* (GGM) as presented in [134]. Namely under the GGM framework an adversary $\mathcal{A}$ has access to a black box that conceptualizes the underlying mathematical group $\mathbb{G}$ in which the assumption takes place. $\mathcal{A}$ without knowing any details about the underlying group apart from its order p is asking for encodings of its choice and the black box replies through a random encoding function ξ that maps elements from $\mathbb{G} \rightarrow \Xi$ as random bit strings of size $\lceil \log_2 p \rceil$. Since our construction operates on asymmetric bilinear pairing groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ we make use of three random encoding functions $\xi_c, c \in [1, 2, T]$ where

$\xi_c : \mathbb{G}_c \to \{0,1\}^{\lceil \log_2 p \rceil}$.

Theorem 11. *Suppose $\mathcal{A}$ is a polynomial probabilistic time adversary that solves the* LEOM *assumption, making at most q_G oracle queries for the underlying group operations on $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ and the $\mathcal{O}_{\mathsf{LEOM}}$ oracle, all counted together. All the encodings $\xi_c, c \in [1,2,T]$ and $\delta, \{\gamma_u\}_{u=1}^n \in \mathbb{Z}_p$ are chosen at random. Then the probability ϵ_2 that $\mathcal{A}$ on input $(p, \xi_1(1), \xi_2(1), \xi_1(a), \xi_1(b), \xi_1(c), \xi_2(\delta), \xi_2(\sum_{i=1}^n \gamma_i))$ to output a tuple $(\xi_1(a), \xi_1(b), \xi_1(c_f = \xi_1(\beta_t \sum_{u=1}^n \gamma_u + \alpha\delta \sum_{u=1}^n x_{u,t})))$ for which neither $x_{u',t'} \neq x_{u,t}$ nor $\mathcal{A}$ has made more than n distinct queries for a fixed time interval t, is bounded as:*

$$\epsilon_2 \leq \frac{(q_G + 16)^2}{p}$$

.

Proof. We assume a polynomial time simulator $\mathcal{B}$ that interacts with adversary $\mathcal{A}$ and simulates the black box for the underlying groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$. $\mathcal{B}$ maintains 3 lists of tuples:

- $\mathcal{L}_1 = \{(F_{1,i}, \xi_{1,i}) : i = 1, \cdots, \tau_1\}$
- $\mathcal{L}_2 = \{(F_{2,i}, \xi_{2,i}) : i = 1, \cdots, \tau_2\}$
- $\mathcal{L}_T = \{(F_{T,i}, \xi_{T,i}) : i = 1, \cdots, \tau_T\}$

where $F_{1,i} \in \mathbb{Z}_p[A, B, \{\Gamma_u\}_{u=1}^n, \Delta, X], F_{2,i}, \mathbb{Z}_p[\Delta, E]$ and $F_{T,i} \in \mathbb{Z}_p[A, B, \{\Gamma_u\}_{u=1}^n, \Delta, E, X]$ are multivariate polynomial on the indeterminants $A, B, \{\Gamma_u\}_{u=1}^n, \Delta, E, X$. Hereafter we will denote inteterminants for polynomials with capital letters and coefficients with lowercase. The random encodings $\xi_{c,i}, c \in [1,2,T]$ of each list $\mathcal{L}_c, c \in [1,2,T]$ are provided to the adversary $\mathcal{A}$ at each step τ, where $\tau = \tau_1 + \tau_2 + \tau_T + 4$. The lists are initialized at step $\tau = 0$ by setting $\tau_1 = 1, \tau_2 = 3, \tau_T = 0$ and assigning $F_{1,1} = 1, F_{2,1} = 1, F_{2,2} = \sum_{u=1}^n \Gamma_u, F_{2,3} = \Delta$, that corresponds to the generators g_1, g_2 and the public information $g_2^{\sum_{u=1}^n \gamma_u}, g_2^\delta$. $\mathcal{A}$ has access to the random encodings $\xi_{1,1}, \xi_{2,1}, \xi_{2,2}, \xi_{2,3}$ respectively.

In what follows we describe how $\mathcal{B}$ simulates the groups operations in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ and the oracle responses to $\mathcal{O}_{\mathsf{LEOM}}$. We first assume that before $\mathcal{A}$ queries the oracle or asks for group operations it has already asked for the random encodings of the elements involved in the operations. Consequently when $\mathcal{A}$ asks for operations in $\mathbb{G}_c, c \in [1, 2, T]$ for some operands $\xi_c, c \in [1, 2, T]$, $\mathcal{B}$ checks if $\xi_c, c \in [1, 2, T]$ already exists in $\mathcal{L}_c, c \in [1, 2, T]$ and aborts if this happens.

- **Group operations:** $\mathcal{A}$ provides $\mathcal{B}$ two operands $\xi_{c,1}, \xi_{c,2}, c \in [1, 2, T]$ and a bit defining multiplication or division. $\mathcal{B}$ starts by incrementing $\tau_c + = 1, c \in [1, 2, T]$. It the computes $F_{1,\tau_c} = F_{1,i} + F_{1,j}$, where $1 \leq i, j \leq \tau_c$ if the operation bit is for multiplication or $F_{c,\tau_c} = F_{1,i} - F_{1,j}$, where $1 \leq i, j \leq \tau_c$ if it is for division. If the new polynomial F_{c,τ_c} is equal to another polynomial $F_{c,l}$ for some $l \leq \tau_c, c \in [1, 2, T]$ in list $\mathcal{L}_c, c \in [1, 2, T]$ then $\mathcal{B}$ fetches the corresponding $\xi_{c,l}$ and forwards it to $\mathcal{A}$, otherwise it chooses a fresh random $\xi_{c,\tau_c} \in \{0, 1\}^{\log_2 p}$ and gives it to $\mathcal{A}$. $\mathcal{B}$ finally appends to $\mathcal{L}_c, c \in [1, 2, T]$ the pair $(F_{c,\tau_c}, \xi_{c,\tau_c}), c \in [1, 2, T]$.

- **Pairing:** A pairing operation in $\mathbb{G}_T$ consists of two random encodings $\xi_{1,i}, \xi_{2,j}$ with $1 \leq i \leq \tau_1$ and $1 \leq j \leq \tau_2$. $\mathcal{B}$ first increments the counter $\tau_T + = 1$. Afterwards it computes the pairing as the multiplication of the appropriate polynomials: $F_{T,\tau_T} = F_{1,\tau_1} \cdot F_{2,\tau_2}$. If the same polynomial already exists in $\mathcal{L}_T$: $F_{T,\tau_T} = F_{T,l}, 1 \leq l \leq \tau_T$ then $\mathcal{B}$ clones the random string $\xi_{T,l}$, otherwise it chooses a fresh random $\xi_{T,\tau_T} \in \{0, 1\}^{\log_2 p}$ and gives it to $\mathcal{A}$. $\mathcal{B}$ finally appends to $\mathcal{L}_T$ the pair $(F_{T,\tau_T}, \xi_{T,\tau_T})$.

- $\mathcal{O}_{\mathsf{LEOM}}$**:** $\mathcal{B}$ increments a counter $\tau_{\mathcal{O}}$ by 1 and sets $\tau_1 + = 3$. $\mathcal{A}$ inputs $(u, t, x_{u,t})$. $\mathcal{B}$ computes the polynomials $F_{1,\tau_1 - 2} = A_t, F_{1,\tau_1 - 1} = A_t(Y), F_{1,\tau_1} = (B\Gamma_u + A\Delta X)$ for the indeterminants $B, \Gamma_u, A, \Delta, X$. If any of the $F_{1,\tau_1 - 2}, F_{1,\tau_1 - 1}, F_{1,\tau_1}$ already exists in $\mathcal{L}_1$ then $\mathcal{B}$ clones the associated random encodings $\xi_{1,l}$ for some $l \in [1, \cdots, \tau_1]$. Otherwise it creates three random encodings $\xi_{1,\tau_1 - 2}, \xi_{1,\tau_1 - 1}, \xi_{1,\tau_1} \in \{0, 1\}^{\log_2 p}$ and forwards them to $\mathcal{A}$. It also stores the pairs $(F_{1,\tau_1 - 2}, \xi_{1,\tau_1 - 2}), (F_{1,\tau_1 - 1}, \xi_{1,\tau_1 - 1}), (F_{1,\tau_1}, \xi_{1,\tau_1})$ in $\mathcal{L}_1$ list.

Eventually $\mathcal{A}$ outputs a forgery $(m_f, \xi_{1,fa}, \xi_{1,fy}, \xi_{1,fxy})$. If $\mathcal{A}$'s forgery is valid then it must hold:

$$\frac{e(\prod c_t, g_2)}{e(\beta_t, g_2^{\sum_{u=1}^{n} \gamma_u}) e(a^{\sum_{u=1}^{n} m_u}, g_2^{\delta})} = 1 \in \mathbb{G}_T \tag{1}$$

We show now that this does not happen always. Indeed w.l.o.g we have the following form for each polynomial in the three lists:

- $F_{1,i} = z_{0,i} + z_{1,i}hB\Gamma_{u,i} + z_{2,i}A\Delta X$, for coefficients $z_{0,i}, z_{1,i}, h, z_{2,i}$.
- $F_{2,i} = w_{0,i} + w_{1,i}\Delta + w_{2,i}E$, for coefficients $w_{0,i}, w_{1,i}, w_{2,i}$.
- $F_{T,i} = y_{0,i} + \eta_{1,i}\Delta hB\Gamma_{u,i} + \eta_{2,i}EhB\Gamma_{u,i} + \rho_{1,i}A\Delta^2X + \rho_{2,i}A\Delta XE$, for coefficients $y_{0,i}, \eta_{1,i}, h, \eta_{2,i}, \rho_{1,i}, \rho_{2,i}$.

Equation (1) following the aforementioned presentation of each polynomial can be rewritten as

$$F_f = F_{T,k} - F_{T,l}F_{T,o} \tag{2}$$

for indexes k, l, o. Simplifying the equation, since it is equal to 0, then the second part consists of a polynomial with determinants $(\Delta\Gamma)^2, (E\Gamma)^2, A\Delta^4X^2, (A\Delta XE)^2$ and the first part with determinants $(\Delta\Gamma, E\Gamma, A\Delta^2X, A\Delta XE)$. Since there are no common terms, then all are canceled out and we are left with $y_{0,k} = y_{0,l}y_{0,o}$. As such $F_f = 0$ only when $y_{0,k} = y_{0,l}y_{0,o}$.

$\mathcal{B}$ assigns random values $(\alpha, \beta, \gamma, \delta, \epsilon, x)$ for the indeterminants $A, B, \Gamma, \Delta, E, X$ and in order for $\mathcal{A}$ to win in the game, it should find two identical polynomial in any of the lists $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_T$ or $F_f = 0$. As such the success probability of $\mathcal{A}$ is bounded by the probability that one at least of the following equations holds:

1. $F_{1,i}(\alpha, \beta, \gamma, \delta, x) - F_{1,j}(\alpha, \beta, \gamma, \delta, x) = 0 : i \neq j$
2. $F_{2,i}(\delta, \epsilon) - F_{2,j}(\delta, \epsilon) = 0 : i \neq j$
3. $F_{T,i}(\alpha, \beta, \gamma, \delta, x) - F_{T,j}(\alpha, \beta, \gamma, \delta, x) = 0 : i \neq j$
4. $F_{f,i}(\alpha, \beta, \gamma, \delta, \epsilon, x) - F_{f,j}(\alpha, \beta, \gamma, \delta, \epsilon, x) = 0 : i \neq j$

$F_{1,i}$ degree is at most 3, $F_{2,i}$ at most 1, and $F_{T,i}$ at most 4. . As such they vanish with probability $\frac{3}{p}, \frac{1}{p}, \frac{4}{p}$ respectively, from the Schwartz-Zippel theorem. As such summing for all possible pairs i, j for each of the aforementioned polynomials the success probability of $\mathcal{A}$ is bounded by:

$$\epsilon_2 \leq \binom{\tau_1}{2}\frac{3}{p} + \binom{\tau_2}{2}\frac{1}{p} + \binom{\tau_T}{2}\frac{4}{p} + \frac{4}{p} \leq \frac{(\tau_1 + \tau_2 + \tau_T + 12)^2}{p}$$

As $\tau_1 + \tau_2 + \tau_T \leq q_G + 4$ then $\epsilon_2 \leq \frac{(q_G+16)^2}{p}$ $\square$

C Resume

La motivation pour cette thèse provient de notre engagement dans un projet sur contrôle d'usage, qui a commencé comme un problème de côté et nous a emmenés progressivement au sujet principal de cette thèse. A savoir, la fonction de ces protocoles de contrôle d'usage est de contrôler la manière dont les données sont utilisées tout au long de leur vie, car les systèmes ordinaires de contrôle d'accès ne peuvent pas assurer la confidentialité pour l'utilisation des données. Un système ordinaire de contrôle d'accès permet à un adversaire de copier et stocker des données supprimées, les dupliquer ou les utiliser avec malveillance et de manière non autorisée. Un élément essentiel d'un régime d'application de contrôle d'usage s'avère être une fonction de la détection de similarité, qui est utilisé pour détecter l'utilisation de données malveillantes par les parties non fiables. Notre étude d'une catégorie spécifique des opérations d'analyse de données, telles que la détection de similarité à des fins de sécurité, nous a inciter à s'intéresser à une catégorie plus vaste des protocoles, dans laquelle d'autres fonctions que la similarité sont évalués par des parties non fiables pour des raisons différentes.

Dans ce type des protocoles, tout comme avec le problème de contrôle d'usage, il existe une exigence contradictoire entre la sécurité et l'utilité, dans ce type de protocoles. Un tiers non fiables cherche à apprendre certaines informations statistiques utiles grâce à un recensement des données représentant une population d'usagers. Les parties non fiables recueillent des données provenant d'individus pendant la phase de collecte. Après avoir recueilli toutes les données, les agrégateurs, qui ne sont pas autorisées à avoir accès aux données individuelles, vont essayer de les analyser, afin d'en tirer une valeur globale. Les données individuelles contiennent des informations personnelles et sensibles, c'est pourquoi les utilisateurs qui les fournissent cherchent à protéger leur vie privée. Grâce à l'analyse des données recueillies auprès des utilisateurs, des informations statistiques utiles peuvent être calculées en texte clair qui aidera aux agrégateurs à prendre une décision. En tant que tel, le problème devient un défi, lorsque les entrées individuelles à la fonction sont obscurcis, afin d'assurer la confidentialité. Désormais, des solutions différentes répondant à ce problème sont appelées PPDCA, en elles utilisent deux classes de techniques. La première classe repose sur l'ajout de bruit aux échantillons de données afin d'assurer la confidentialité. Le bruit ajouté permet à l'agrégateur de calculer une fonction statistique avec une erreur. L'autre variante de solutions est basée sur la cryptographie. Grâce à des techniques non-conventionnelles de cryptage et de gestion de clés, l'agrégateur non fiable apprend le résultat d'une fonction statistique sans aucun bruit. En dépit des progrès réalisés par ces solutions cryptographiques concernant la vie privée et l'efficacité, le modèle de sécurité n'implique pas un agrégateur entièrement malveillant ou suppose l'existence d'un distributeur de clés complètement fiable qui distribuant des clés à chaque partie du protocole. Cette thèse porte sur les techniques cryptographiques pour les protocoles PPDCA avec un modèle de menace plus fort, tout en assouplissant les hypothèses de confiance existantes, afin de mieux répondre au déploiement de monde réel. Avant de présenter les défis et les objectifs que nous abordons, nous présentons des scénarios de cas d'utilisation qui nous motivent.

C.1 Applications

Dans cette partie, nous fournissons, à travers des scénarios du monde réel, des preuves pour les applications diverses des protocoles PPDCA. La confluence des serveurs puissants, des appareils informatiques omniprésents et d'informatique intelligente, permet la collection d'une énorme quantité des données provenant des utilisateurs finaux. L'existence d'une grande quantité d'informations, permet à l'agrégateur de déduire, grâce à ses opérations, des informations statistiques sur la population sous-jacente, qui améliorent la protection sociale: imaginez un scénario de services médicaux, selon lequel les patients d'un hôpital reçoivent des renseignements personnels sur leur santé en forme électronique. Cette information représente leur historique médicale et elle est considéré comme une information personnelle. Les scientifiques médicaux, d'autre part, cherchent à travailler sur les données afin d'en tirer des informations statistiques, notamment les modèles prédictifs sophistiqués pour découvrir la prédisposition aux maladies (cancer, crise cardiaque, des anomalies génétiques) ou pour connaitre, grâce à des données génomiques, à quel point il est probable que les descendants héritent certaines maladies. Les données médicales, produites par une population de patients, sont recueillies par les scientifiques médicaux qui se comportent comme des agrégateurs. La mauvaise usage de données privées des patients est possible, ce qui peut avoir une conséquence négative dans la vie des patients: par exemple, quand une compagnie d'assurance décide si un patient va devenir un client potentiel ou pas, l'élicitation des données médicales à cette compagnie d'assurance peut avoir une influence négative sur sa décision. Le problème centrale de ce scénario, qui se passe entre les fournisseurs des données médicales (utilisateurs) et les scientifiques médicaux, qui peuvent agir avec malveillance, est d'assurer la confidentialité des données individuelles, tout en permettant aux scientifiques médicaux d'effectuer certaines opérations dessus.

Dans un autre contexte, grâce à la chute des coûts des appareils informatiques, les compteurs intelligents sont beaucoup déployés dans les foyers afin de signaler la consommation d'énergie dans un environnement de réseau électrique intelligent. Comme la consommation d'énergie, surveillé par les compteurs intelligents, peut révéler des informations sensibles d'un foyer, tels que le nombre de personnes, les appareils et les activités personnelles, les usagers n'ont pas trop envie de dévoiler leurs habitudes de consommation d'énergie. De l'autre côté du système de réseau intelligent, les fournisseurs d'énergie, considérés comme des consommateurs des données, collecte et analyse des échantillons de la consommation d'énergie de compteurs intelligents afin de réaliser différents types d'optimisation. A partir de l'analyse de ces échantillons, ils sont en mesure de prévoir précisément la demande d'électricité, afin de répartir l'énergie à l'avance, en fonction des besoins de toute une population. C'est une opposition typique qui se pose donc entre les deux bout du système de réseau intelligent, la vie privée d'un côté et l'utilité de l'autre côté. Le défi du scénario de comptage intelligent est, donc, de préserver la confidentialité des données individuelles, tout en permettant aux tiers non fiables d'avoir accès à certaines informations globales sur les comptages.

C.2 PPDCA

Dans les scénarios mentionnés ci-dessus un tiers non fiable collecte les données provenant de plusieurs utilisateurs. Les utilisateurs veulent protéger la confidentialité de leurs données et ils hésitent à révéler leurs informations personnelles. D'autre part, la partie non fiable cherche à obtenir en texte clair une fonction sur l'ensemble des données, sans apprendre des entrées individuelles. Pendant une phase de collecte, un agrégateur recueille des données obscurcies. Ensuite, au cours de la phase d'analyse, l'agrégateur effectue certaines opérations sur les données qui lui permettent de révéler en texte clair des informations statistiques utiles sur les informations recueillies. Les faits de préserver la confidentialité des données et de permettre l'accès spécifique à l'information globale ont les exigences contradictoires, ce qui rend la conception de protocoles PPDCA difficile. Examinons maintenant quelques solutions possibles de ce problème. Cryptage homomorphique permet des opérations sur les données cryptées, mais ne résout pas le problème de la dérivation d'une valeur globale en texte clair. Dans un contexte standard, basé sur le cryptage homomorphique, l'agrégateur non fiable aurait besoin de la clé de cryptage secrète pour décrypter le résultat global cryptées, ce qui compromettrait la vie privée des utilisateurs. Suite à une direction différente, le problème pourrait être atténué avec les protocoles calculs multi-partis (MPC). Cependant, MPC implique une surcharge importante de communication, car les utilisateurs sont obligés d'échanger des multiples messages secrets pour que le calcul d'une fonction sur les données puisse être effectué. Le paradigme de cryptage fonctionnel peut être utilisé pour concevoir des protocoles PPDCA, mais en cas des entrées multiples les modèles de cryptage fonctionnels proposés seraient complexes et d'un coût prohibitif. Nous nous tournons vers des approches plus adaptées qui portent spécifiquement sur le problème PPDCA. L'idée des solutions liées au bruit est d'ajouter un peu de bruit à chaque valeur de données avant de l'envoyer à l'agrégateur. Le bruit empêche l'agrégateur de compromettre la confidentialité individuelle, mais il est adapté de manière appropriée pour que certaines statistiques portant sur toutes les entrées de données puissent être inférées. Il existe un autre approche qui utilise des protocoles cryptographiques avec un ensemble restreint d'opérations qu'un agrégateur peut effectuer sur l'ensemble des données.

C.3 Objectifs

Bien que l'approche le plus approprié pour PPDCA semble être celui qui utilise des protocoles cryptographiques, des solutions existantes, basées sur des protocoles cryptographiques, ont toujours quelques limitations. Tout d'abord, les protocoles existants ne fonctionnent qu'avec un ensemble basique de fonctions d'agrégation: l'extension de ceux-ci semble être un très bon défi pour la recherche. De plus, des protocoles cryptographiques existants pour PPDCA souffrent d'exigences irréalistes concernant la gestion des clés, car il existe une dépendance d'un distributeur des clés entièrement fiable ainsi que la nécessité de mettre à jour le matériel de clé pour la population entière des utilisateurs. Les utilisateurs existants du modèle sont également touchés en cas de faute, parce que les utilisateurs qui participent déjà dans le protocole, ont besoin de recevoir de nouvelles clés. Dans le cas des appareils à faibles ressources, tel que les compteurs intelligents, en raison des contraintes de ressources de l'appareil, il est d'une grande importance de soutenir la dynamicité et la résilience aux pannes avec faibles coûts de communication. Enfin, les protocoles cryptographiques suivent le modèle de menace honnête, mais curieux, dans lequel l'agrégateur est semi-approuvé de suivre les règles du protocole. Nous concluons qu'il est important d'introduire un modèle de sécurité plus fort en prenant en compte adversaires plus puissants qui cherchent à s'écarter des règles de protocole, afin d'altérer de façon malicieuse les résultats globaux. Les objectifs de cette thèse peuvent être résumés en quelques points:

1. Fournir de nouvelles fonctionnalités qu'un agrégateur peut effectuer sur les données pour la collecte de données sécurisées et sur l'analyse, qui ne sont pas disponibles à partir des protocoles cryptographiques existants. Nous soulignons que les fonctionnalités étendues devraient venir avec une récompense idéal pour la confidentialité sans la compromettre en grande partie: l'apprentissage des statistiques globales sur l'ensemble de la population d'utilisateurs est faisable et acceptable, mais l'apprentissage de comportement en ligne de chaque utilisateur de la population est considéré comme la violation de la vie privée des utilisateurs.

2. Conception d'un protocole cryptographique qui serait approprié pour une population dynamique des utilisateurs avec la résilience aux pannes. Nous soulignons que le soutien de dynamicité et de tolérance aux pannes ne devrait pas compromettre la vie privée des utilisateurs.

3. La formalisation de nouvelles définitions de sécurité qui ne existent pas dans la littérature actuelle. En quelques mots, nous renforçons les définitions de confidentialité existantes

concernant inconscience de l'Aggregator, telles qu'elles ont déjà été proposées, en réduisant la quantité de confiance qui doit être placé dans une seule entité. De plus, nous proposons une définition intégrée de la sécurité qui garantit à la fois la confidentialité et la vérification des calculs. D'après l'analyse effectuée, nous présentons brièvement les résultats de cette thèse:

C.4 Clustering aux profils privées

Les tiers non fiables ont tendance à exploiter, de plus en plus, les informations des usagers pour assurer une meilleure diffusion de contenu. Les systèmes recueillent des données sur les utilisateurs et leurs interactions avec leur environnement afin de fournir le contenu le plus adéquat et personnalisé. Les informations utilisées, comprenant les relations sociales des usagers et leurs intérêts personnels, sont constituées des données très sensibles, ce qui, donc, pose le problème dans le domaine de la vie privée. Une solution naïve du problème mentionné ci-dessus pourrait être le cryptage des données avant de les analyser. Cela ne résoudra pas le problème puisque les opérations ne sont pas envisageables après le cryptage. Une solution plus adéquate pourrait être le cryptage des données de façon homomorphique, pour que les propriétés statistiques des données après le cryptage puissent être calculées. Bien que cette solution semble abordable, les systèmes actuels de cryptage homomorphiques ne réussissent pas à donner une solution pour un système d'analyse globale appliqué aux certaines grandes échelles d'ensemble de données. L'un des éléments constitutifs de base dans la grande majorité des scénarios d'analyse de données est la détection de similarité. En analysant l'ensemble de données d'utilisateurs, un moteur de recommandation peut découvrir des profils similaires et ainsi recommander à un nouvel utilisateur un contenu qui a déjà été consommé par d'autres utilisateurs similaires déjà existants. Les annonceurs en ligne ont cherché à augmenter leur chiffre d'affaires en examinant le comportement en ligne des usagers. Cela implique que les détaillants en ligne externalisent des informations personnelles sensibles aux annonceurs. Les utilisations mentionnées ci-dessus impliquent un risque de violation de confidentialité. Etant donné que les opérations d'analyse de données sont effectuées sur des informations personnelles confidentielles et sensibles, il est possible que la vie privée des individus ne soit protégés. Ainsi, les utilisateurs et les entreprises, soit ont tendance à ne pas soumettre leurs données à une analyse plus approfondie des tiers non fiables, soit ils leur donnent un accès limité en raison du risque de violation de confidentialité [86, 107, 112, 127]. Des solutions radicales comprennent une limitation relative aux opérations d'analyse de données disponibles qu'un agrégateur peut effectuer à partir de la perspective de l'analyseur, ce qui dégrade la précision de l'analyse des données. Dans ce chapitre, nous présentons un protocole de la préservation de la vie privée pour la détection de similarité. Similarité cosinus est capable de reconnaître des vecteurs similaires basés sur l'angle formé entre ces deux vecteurs. Notre mécanisme de préservation de la vie privée, trace d'abord les données des utilisateurs sous la forme des vecteurs et ensuite chaque utilisateur crypte individuellement ses données, de sorte que la représentation géométrique des données vectorisées soit préservée. La sécurité de cette solution est confirmée sous la sécurité des générateurs pseudo-aléatoires. L'exactitude de la solution proposée est ensuite évaluée avec l'aide d'étude sur les caractéristiques de la personnalité des utilisateurs.

L'idée de la solution est d'appliquer des transformations sur les vecteurs originaux qui d'une part préservent l'angle entre n'importe quel paire d'entre eux et d'autre part assurent la vie privée. Étant donné que la rotation dans un espace à deux dimensions conserve les angles, on applique cette transformation aux vecteurs à deux-dimensions, nommés des sous-vecteurs qui sont originaires du vecteur de données initial. En outre, ces sous-vecteurs sont encore dilatés au hasard et ainsi obscurcis, mais sans encore avoir eu un impact sur l'angle. Nous avons observé des fuites de sécurité lorsque le mécanisme de cryptage ne comporte pas tous les deux éléments, les dilatations aléatoires et les rotations. Si chaque utilisateur sélectionne uniquement la dilatation aléatoire comme le mécanisme de cryptage, un adversaire, après avoir obtenu une bonne estimation d'un coefficient de vecteur d'un utilisateur, peut récupérer les vecteurs bidimensionnels spécifiques en calculant l'inverse de l'élément estimé et en le multipliant par le coefficient crypté. Le problème susmentionné est atténué grâce à des rotations. Cependant, un autre apparaît lorsque les rotations de vecteur aléatoires sont utilisées: si deux utilisateurs avec des vecteurs secrets D_i, D_j ont respectivement la même valeur en même position de leurs vecteurs, des vecteurs cryptés correspondants auraient la même valeur à cette position, uniquement si le cryptage est fait avec une matrice de rotation R_θ de l'angle θ. Ceci s'oppose à la définition de la sécurité 36. Ainsi, pour bien préserver la similarité cosinus après le cryptage des vecteurs, à la fois la dilatation aléatoire et la rotation sont appliquées. Par conséquent, grâce à la rotation, l'adversaire ne peut pas découvrir des similarités entre les coordonnées d'un vecteur. Le processus de tracer les vecteurs en sous-vecteurs diminue également la probabilité de découvrir le vecteur d'origine, puisque chaque sous-vecteur a un facteur de dilatation différent. Contributions Notre technique assure :

- Calculation de similarité sans avoir : Un tiers non fiable peut effectuer l'agrégation des données cryptées sans apprendre des entrées de données individuelles.

- La sécurité confirmée : La sécurité de notre protocole est confirmée dans le modèle standard.

C.5 Preservation de la vie privée par les statistiques dans le réseau intelligent

Les compteurs intelligents sont des appareils déployés dans les foyers qui ont pour l'objet de mesurer la consommation d'énergie dans des intervalles de temps spécifiques. Ils ne mesurent pas uniquement la consommation d'électricité mais aussi celle du gaz et de l'eau. Les raisons de ce déploiement important des compteurs intelligents sont nombreuses. D'une part, les fournisseurs peuvent, ainsi, apprendre plus précisément les intervalles de temps où chaque foyer consomme plus d'énergie et donc ajuster de manière appropriée la facturation de chaque client et de prévoir la demande potentielle d'énergie. D'autre part, les habitants d'un foyer peuvent recevoir des conseils et ainsi changer leurs habitudes de consommation d'énergie. Notamment, après avoir appris quelle est la période de consommation la plus élevée, un client peut préférer de consommer l'énergie d'une manière plus efficace. Dans ce chapitre, nous abordons le problème consistant en calcul de la consommation maximale continue de l'énergie au cours des relevés, envoyées par les compteurs intelligents individuels tout en préservant la vie privée des usagers. Suite à l'analyse que nous avons faite dans le chapitre 3, ce type de statistiques n'existe pas dans les ouvrages scientifiques actuels. Nous supposons que tout les deux, le fournisseur et les compteurs intelligents individuels, cherchent à déterminer l'intervalle où le compteur intelligent consomme le plus. Une telle opération ne peut être effectuée que par un compteur intelligent, car il lui manque de ressources et en particulier de mémoire. Le compteur intelligent aurait besoin d'un nombre important de valeurs afin de trouver la valeur maximale correspondant à une consommation continue. Cependant, la conséquence de l'externalisation de ces calculs au fournisseur sera naturellement la fuite des informations concernant les consommations périodiques qui sont certainement des informations très sensibles. Nous proposons donc une solution, dans laquelle les compteurs intelligents envoient leur mesure périodique au fournisseur de la façon que la confidentialité soit préservée, tout en permettant à cette organisme de calculer l'intervalle de temps de la consommation maximale. La solution proposée est fondé sur un cryptage de maintien de l'ordre (OPE), qui par définition conserve l'ordre des valeurs en texte clair après leur cryptage sans révéler aucune information supplémentaire. De plus, afin de filtrer les maximums spontanés (qui apparaissent, car parfois des appareils domestiques par exemple, s'allument et s'éteignent du façon erroné), le compteur intelligent envoie également les différences de valeurs de consommation consécutive au fur et à mesure, en utilisant une approche "sur la volée" de sorte que le compteur intelligent n'a pas besoin de stocker de l'information auxiliaire. Grâce aux différences le fournisseur est capable de déterminer la période de consommation maximale qui est continue. La sécurité de la solution proposée est confirmée avec l'aide d'une preuve réductionniste de l'hypothèse POPF-CPA [23], ce qui correspond à la notion de sécurité qui caractérise la sécurité de l'OPE. Idée Dans cette partie, nous donnons une courte description de notre solution. Notre modèle de PPSGS réussie à faire oublier les tiers grâce à un système de cryptage de maintien de l'ordre dans lequel l'ordre des éléments numériques dans l'espace de texte clair est également conservé dans l'espace de texte codé. Chaque compteur intelligent est équipé d'un module matériel inviolable dans lequel une clé secrète est intégrée. Cette clé secrète est utilisée pour crypter les mesurages à chaque intervalle de temps. Grâce à la primitive cryptographique des fonctions de maintien de l'ordre, des fonctions préservant un ordre et utilisant la clé, choisies de manière uniforme et au hasard, ne se distinguent pas des fonctions idéales. Ainsi, ce n'est que l'ordre qui est révélé au fournisseur qui agit en tant qu'entité d'analyse de données.

Pour l'exactitude de l'analyse, une fois que le fournisseur a identifié l'intervalle de temps dans lequel un compteur intelligent a consommé le maximum, grâce à l'information supplémentaire composée par les différences entre chaque consommation, il peut vérifier qu'en effet il y a une consommation valable d'énergie continue maximale autour de cet intervalle de temps. Si les différences sont convergées à 0, cela veut dire qu'il a une forte indication que les mesurages autour de cet intervalle particulier font partie d'une consommation continue maximale. Bien que l'objectif de la publication des différences est de permettre aux fournisseurs d'énergie de détecter les consommations continues d'énergie maximale, les chercheurs ont exprimé l'intérêt pour concevoir des protocoles préservant la confidentialité pour les détections de crête, pour que les opérateurs énergétiques puissent identifier les lignes électriques surchargées [53]. En tant que tel, notre solution est également convenable pour le cas mentionné ci-dessus. L'avantage de notre approche est que les compteurs intelligents n'ont pas besoin de stocker les différences ou les textes chiffrés afin d'effectuer l'analyse, mais ceux-ci sont calculées et envoyées immédiatement au fur et à mesure. Du point de vue des fournisseurs, la vérification d'un intervalle de consommation maximale continue est effectuée par lot avec une seule opération. De plus, comme il sera établi dans la section 5.2.3, les différences ne compromettent pas les exigences de la confidentialité du modèle. L'information provenant de l'identification d'une consommation d'énergie continue améliorerra les prévisions de consommation d'énergie mais également permettra une meilleure allocation de l'énergie à l'avance des producteurs d'énergie. En outre, les informations sur l'intervalle de la consommation d'énergie maximale peuvent être renvoyées aux usagers pour qu'ils puissent passer rapidement de leur utilisation habituelle d'énergie pendant les périodes de tarif haut à celle des périodes de tarif bas. Il est impossible d'effectuer cette opération individuellement à chaque compteur intelligent, car leurs ressources ne sont pas suffisantes pour l'exécution des grandes opérations d'analyse de données. D'autre part, un mécanisme de vérification de l'intégrité est indispensable pour que le fournisseur soit assuré que les mesurages sont envoyés à partir des compteurs intelligents existants et authentifiés.

Contributions

Notre protocole assure:

- L'inconscience d'agrégateur: Un agrégateur non fiable n'apprend aucune des entrées individuelles. Il n'apprend que le résultat final, ce qui est en effet l'intervalle de temps dans lequel un utilisateur consomme le maximum.

- La consommation continue maximale: Un utilisateur dans un foyer peut allumer et éteindre un appareil de haute énergie, immédiatement et dans n'importe quel moment. Cela résultera en résultats erronés pour l'agrégateur, car un usage spontané d'un appareil de l'énergie pour une période très courte donnera les statistiques globales fausses. Nous étions en mesure de capter cette information grâce à une fonction de codage delta, qui permet à l'agrégateur de discerner si les différences de consommations d'énergie en texte clair autour d'un intervalle de temps, convergent à zéro, ce qui est interprété comme une consommation maximale d'énergie continue.

- La sécurité confirmée: La sécurité de notre protocole est bien confirmée.

C.6 Protection de la vie privee pour l'agrégation de séries temporelles dans le manière dynamique

Nous proposons un protocole PPDCA, qui élimine le besoin d'une redistribution des clés après une connexion et une déconnexion d'un utilisateur, ainsi que la nécessité d'un distributeur des clés complètement fiable. Ainsi, nous renforçons le modèle de menace des protocoles PPDCA actuels avec des fonctionnalités améliorées concernant la dynamicité et la tolérance aux pannes. Les caractéristiques du protocole amélioré peuvent être résumées comme suit:

- Pas de distributeur de clés. Contrairement à la plupart des protocoles PPDCA, dans notre modèle il n'y a pas distributeur des clés fiable. Par contre, nous introduisons une partie semi-fiable, appelée Collecteur, qui rassemble des informations clés partielles prises des utilisateurs via un canal sécurisé.

- Populations dynamiques des utilisateurs soutenues. Aucune coordination n'est indispensable pour gérer les changements qui concernent la population des usagers. Ceci est possible grâce à un mécanisme de clé auto-générée, selon lequel aucun accord de clé entre utilisateurs n'est exigé.

- Confidentialité. En ce qui concerne la confidentialité, le modèle assure l'inconscience d'agrégateur, tel qu'introduit par Elaine Shi et al. [131]. Autrement dit, l'agrégateur non fiable n'apprend que la somme et la moyenne sur les données privées des utilisateurs à la fin de l'exécution du protocole. Par ailleurs, nous montrons que le collecteur ne relève aucune information sur les données privées des utilisateurs.

- Efficacité. Notre modèle reprend certaines fonctionnalités de Joye et al. [92], telles que la possibilité de calculer la somme et la moyenne sur un grand nombre d'utilisateurs sans restrictions concernant la diversité des valeurs des utilisateurs. Il est également extensible dans le sens que les décryptages effectués par l'agrégateur ne dépendent pas du nombre d'utilisateurs.

Aperçu

Afin d'éliminer le besoin d'un distributeur entièrement fiable et pour soutenir la gestion de groupe dynamique sans induire de communication ou de calcul supplémentaires au-dessus, nous utilisons deux techniques:

- Le mécanisme de division de responsabilité: Chaque utilisateur, $\mathcal{U}_i$, envoie en même temps, un cryptage de son échantillon de données privées à l'agrégateur $\mathcal{A}$ et une autre version obscurcie de son clé secrète, sk_i, au collecteur demi-fiable $\mathcal{C}$, de façon que ni l'agrégateur, ni le collecteur, ne peuvent violer la confidentialité des échantillons individuels fournis par les utilisateurs.

- Auto-génération de clés secrètes: Les clés secrètes, utilisées pour le cryptage des échantillons de données, sont générées indépendamment par les utilisateurs, sans coordination par un distributeur des clés fiable.

Une présentation de notre solution est illustrée dans la figure 6.1. Chaque utilisateur $\mathcal{U}_i$ choisit indépendamment sa clé secrète sk_i, alors que l'Aggregator non fiable génère une clé aléatoire sk_A. Pour chaque intervalle de temps t, agrégateur, $\mathcal{A}$, publie une version obscurcie $\mathsf{pk}_{A,t}$ de la clé secrète pk_A. Les utilisateurs $\mathcal{U}_i$, d'autre part, cryptent leurs échantillons de données privée $x_{i,t}$, avec leurs clés secrètes, sk_i, en utilisant le système de cryptage Joye-Libert, et envoient les textes chiffrés correspondants $c_{i,t}$, à l'agrégateur $\mathcal{A}$. Ils, également, occultent leurs clés secrètes sk_i par $\mathsf{pk}_{A,t}$, et envoient l'information auxiliaire obtenue $\mathsf{aux}_{i,t}$ au Collector C via un canal sécurisé. Collecteur $\mathcal{C}$ calcule une fonction $g(t)$ à partir de l'information auxiliaire $\mathsf{aux}_{i,t}$ qu'il a reçu et transmet la sortie auxt à l'agrégateur A. Après avoir reçu les textes chiffrés $c_{i,t}$ et l'information auxiliaire aux_t, l'agrégateur, $\mathcal{A}$, utilise sa clé secrète sk_A et apprend la somme $\sum x_{i,t}$, pour l'intervalle de temps t. De cette façon, nous éliminons le besoin d'un distributeur de clés fiable qui connaît de clés privées des usagers, tout en assurant que ni l'agrégateur, ni le collecteur ne peuvent déduire des informations sur les données individuelles des usagers. Egalement, la gestion efficace de groupe dynamique qui ne nécessite pas de mécanisme d'aucune mise à jour de clé est atteinte.

De cette manière notre protocole assure :

- L'inconscience d'agrégateur et de collecteur : Le protocole assure l'inconscience d'agrégateur et de Collector. Grâce à cela la confidentialité n'est pas compromise ni par l'agrégateur, ni par le collecteur, ce qui aide l'agrégateur de calculer la somme.

- Dinamicité et la tolérance aux pannes : L'agrégateur apprend la somme, même en cas d'une panne causée par une erreur de communication. De plus, le protocole est dynamique dans le sens que les connections et déconnections dynamiques n'ont aucune influence sur les utilisateurs existants de ce protocole, car il n'y a pas besoin de procéder à une nouvelle phase de distribution de clé.

- La sécurité confirmée : La sécurité de notre protocole est confirmée dans le modèle de l'oracle aléatoire sous le caractère insoluble des problèmes mathématiques bien connus contre agrégateurs honnêtes, mais curieux.

C.7 PUDA – Confidentialité et infalsifiable pour l'agrégation

Nous considérons un scénario dans lequel un agrégateur recueille des données individuelles de plusieurs utilisateurs, qui n'interagissent pas entre eux, et ensuite exécute une fonction qui délivre une valeur globale. Contrairement au protocole présenté dans le chapitre 6, le résultat est ensuite transmis à un analyseur de données qui peut enfin extraire des informations utiles relatives à l'ensemble de la population des usagers. Protocoles PPDCA existants sont concentrés uniquement sur le problème de la confidentialité des données et ils considèrent que l'agrégateur est honnête, mais curieux: l'agrégateur est curieux de découvrir le contenu de chacune des données individuelles, mais effectue l'opération d'agrégation correctement. Ici, nous considérons un modèle de sécurité plus puissant, où l'agrégateur est supposée être malveillant: L'agrégateur peut fournir une valeur globale fausse à l'analyseur de données. Pour protéger contre ce comportement malveillant, nous proposons qu'en plus des la valeur globale, l'agrégateur fournit une preuve de l'exactitude du calcul du résultat global. Nous exigeons également que l'analyseur de données ne puisse pas communiquer avec chaque utilisateur et que le résultat soit publiquement vérifiable. De plus, analogue aux solutions existantes, le protocole proposé assure inconscience l'agrégateur et l'analyseur de données dans le contexte multi-utilisateur; ce qui signifie que ni l'analyseur de données, ni l'agrégateur apprend des entrées de données individuelles. L'idée sous-jacente de notre solution est la suivante: chaque utilisateur crypte ses données selon le modèle Shi et al. [131], en utilisant sa propre clé de codage secrète, et envoie le texte chiffré obtenu à l'agrégateur non fiable. Les utilisateurs, également étiquètent leurs données de façon homomorphique, tout en utilisant deux couches d'aléatoire avec deux clés différentes et ils transmettent les étiquètes à l'agrégateur. Ce dernier calcule la somme en appliquant des opérations sur les textes chiffrés et également il dérive de ces étiquètes une preuve de l'exactitude du résultat. L'agrégateur envoie finalement le résultat et la preuve à l'analyseur de données qui vérifie l'exactitude du calcul. A notre connaissance, nous sommes les premiers à définir un modèle PUDA. Nous instancions également un modèle PUDA qui poursuit principalement les trois objectifs suivants :

- Milieu multi-utilisateurs où plusieurs utilisateurs produisent des données personnelles sensibles sans aucune interaction entre eux.

- Vérifiabilité publique de la valeur globale.
- Confidentialité des données individuelles pour tous les participants.

Aperçu de PUDA

Quand il s'agit d'un modèle étendu avec un agrégateur non fiable, il est extrêmement important de concevoir une solution dans laquelle l'agrégateur non fiable ne peut pas fournir des résultats faux à l'analyseur de données. Une telle solution utilisera un système de preuve qui permet à l'analyseur de données de vérifier l'exactitude du calcul. Pourtant la vérifiabilité devrait se réaliser sans sacrifier la confidentialité. Pour atteindre cet objectif, nous proposons un protocole qui s'appuie sur les techniques suivantes:

- Un algorithme de cryptage homomorphique qui permet à l'agrégateur de calculer la somme, sans dévoiler les données individuelles
- Une étiquette homomorphique qui permet à chaque utilisateur d'authentifier l'entrée de données xi t, de sorte que l'agrégateur puisse utiliser les étiquettes recueillies, pour constituer une preuve démontrant à l'analyseur de données DA l'exactitude de la somme globale.

L'explication la plus concise est la suivante: un ensemble d'utilisateurs sans interférence sont connectés à des services personnelles et appareils qui produisent des données personnelles. Sans aucune coordination, chaque utilisateur choisit une clé d'étiquette aléatoire, tk_i , et envoie un codage de celle-ci, tk_i, au distributeur de clés. Après avoir recueilli toutes les clés codées, tk_i, des utilisateurs, le distributeur des clé publie la clé de vérification publique, VK, de ce groupe d'utilisateurs. Cette clé de vérification est calculé comme une fonction du codage tk_i. Ensuite, le distributeur de clés donne à chaque utilisateur dans le système une clé de cryptage, ek_i, qui sera utilisée pour calculer les textes chiffrés des utilisateur. Par conséquent, la clé secrète de chaque utilisateur, ski, est définie comme la paire constituée de la clé d'étiquette, tk_i, et la clé de cryptage, ek_i. Finalement, le distributeur de clés fournit l'agrégateur avec une clé secrète, sk_A, calculée comme la somme des clés codées, ek_i, et ensuite se met hors ligne. Or, à chaque intervalle de temps t, chaque utilisateur utilise sa clé secrète, sk_i, pour calculer un texte chiffré basé sur l'algorithme de cryptage de Shi et al. [131] et une étiquette homomorphique sur son entrée de données sensibles. Lorsque l'agrégateur collecte les textes chiffrés et les étiquettes de tous les utilisateurs, il calcule la somme, sum_t, des données des utilisateurs et une preuve de la somme, σ_t, et ensuite les transmet à l'analyseur de données. Lors de l'étape finale du protocole, l'analyseur de données vérifie avec la clé de vérification, VK, et la preuve σ_t, la validité du résultat, sum_t. Bien que la modification semble simple et évidente, la preuve de Falsification - type II s'avère être complexe. Grâce à l'algorithme de cryptage homomorphique de Shi et al. [131] et à la manière dont nous construisons nos étiquettes homomorphiques, nous montrons que notre protocole assure inconscience de l'agrégateur. De plus, nous montrons que l'agrégateur ne peut pas forger des résultats faux. Enfin, nous constatons que l'Analyseur de données DA ne conserve aucune forme des transcriptions des utilisateurs textes chiffrés, mais il ne retient que la clé de vérification publique, la somme, sum_t et la preuve, σ_t.

Contributions

En réalisant notre analyse, nous avons fait les contributions suivantes:

- Agrégation infalsifiable: Un agrégateur malveillant ne peut pas convaincre un analyseur de données avec une agrégation erronée avec une probabilité non négligeable.

- L'inconscience: La confidentialité individuelle est préservée contre les parties non fiables du protocole, tandis que l'agrégateur peut apprendre la somme des entrées de données.

- Vérification publique du temps constant: Le temps d'exécution de l'algorithme de vérification est constant et ne dépend pas du nombre d'utilisateurs. De plus, la construction permet la vérification publique de l'exactitude du resultat avec une clé de vérification publique.

- La sécurité confirmée: La sécurité de notre protocole est confirmé grâce à une nouvelle hypothèse mathématique dont la preuve de la sécurité est montré dans le modèle de groupe générique

C.8 Conclusion

PPDCA contribuent considérablement à la prise de décision. L'agrégation des données permet aux agrégateurs de déduire des informations statistiques utiles, contribuant à la protection sociale. Cependant, les usagers sont réticents à révéler leurs valeurs de données en clairtext, à cause de la nature des informations personnelles sensibles que chaque usager confie à un tiers non fiable. Les solutions actuelles proposent des différents mécanismes de la protection de la vie privée des usagers. Ils préservent la vie privée des usagers, mais en même temps ils donnent la possibilité à un tiers non fiable d'apprendre une fonction statistique f sur l'ensemble de la population d'usagers. Dans cette thèse, nous avons d'abord défini ce qu'est un protocole PPDCA et ensuite nous avons présenté l'état de l'art des protocoles PPDCA. Nous avons commencé notre analyse avec des techniques basées sur le bruit. D'après ces techniques, chaque usager ajoute du bruit à la valeur de données, de façon qu'un agrégateur non fiable puisse déduire des statistiques bruyantes pour l'ensemble de la population des usagers. Les techniques basées sur le bruit sont limitées à fournir des statistiques bruyantes et donc, elles ne conviennent pas aux scénarios de cas d'utilisation dans lesquels la précision dans le résultat final de la fonction statistique f est indispensable. Les protocoles cryptographiques ont pour l'objectif de répondre au besoin de précision dans le calcul de f. Les usagers cryptent leurs données de façon appropriée, afin de permettre le contrôle d'accès partiel sur une valeur globale. Après avoir présenté les protocoles cryptographiques actuels pour PPDCA, nous avons procédé à une taxonomie détaillée des protocoles cryptographiques dans la littérature déjà existante basées sur des caractéristiques différentes de ceux-ci. En faisant notre analyse, nous avons identifié une lacune dans les directions suivantes:

- Les protocoles existants sont concentrés sur une famille restreinte des fonctions f, qu'un agrégateur peut apprendre, telle que la somme, le produit intérieur et des opérations booléennes.
- La majorité des solutions cryptographiques actuelles supposent qu'il existe un revendeur de clé complètement fiable, qui distribue des clés secrètes aux usagers et à l'agrégateur. Les

conséquences d'un revendeur de clé entièrement fiable entravent le déploiement des protocoles dans un environnement dynamique. A savoir, dans un environnement dynamique les usagers se joinder et se laisser à chaque exécution du protocole, obligeant ainsi les usagers existants d'obtenir de nouvelles clés secrètes par le revendeur de clé fiable. De plus, ce point de confiance, rend la faute des protocoles intolérante, puisque en cas de faute, le revendeur de clé fiable doit distribuer de nouvelles clés à tous les usagers existants.

- Il n'y a pas de solutions soutenant un modèle de sécurité plus forte dans les protocoles PPDCA. Les protocoles actuels ont deux fonctionnalités. Soit ils supposent qu'il existe un agrégateur entièrement fiable, soit ils appuient leur sécurité sur un modèle honnête mais curieux, dans lequel l'agrégateur est fiable pour exécuter correctement les étapes du protocole, mais aussi il est curieux d'apprendre toute messages échangés.

Ainsi, on a introduit quatre protocoles pour PPDCA:

- Un protocol permettant de calculer la similarité entree deux profils privées.
- Un protocol qui facilite le calcul de l'intervalle de temps dans laquelle la consommation énergétique d'un utilisateur a été maximale, sans divulguer les valeurs individuelles de cette consommation.
- Un protocol dynamique pour l'aggregation des séries temporelles sans distributeur des clés.
- Un protocol PPDCA avec verification de calcul.

Bibliography

[1] http://amturing.acm.org/award_winners/rivest_1403005.cfm.

[2] Cubieboard. http://cubieboard.org/.

[3] Recognition european project, 2011.

[4] G. Ács and C. Castelluccia. I have a dream! (differentially private smart metering). In *Information Hiding*, pages 118–132, 2011.

[5] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '01, pages 247–255, New York, NY, USA, 2001. ACM.

[6] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order-preserving encryption for numeric data. In *SIGMOD Conference*, pages 563–574, 2004.

[7] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *ACM Sigmod Record*, volume 29, pages 439–450. ACM, 2000.

[8] R. Agrawal and R. Srikant. Privacy-preserving data mining, 2000.

[9] J. A. Akinyele and M. G. an Aviel D. Rubin. Charm: A tool for rapid cryptographic prototyping. http://www.charm-crypto.com/Main.html.

[10] J. A. Akinyele, M. Green, and A. D. Rubin. Charm: A framework for rapidly prototyping cryptosystems. *IACR Cryptology ePrint Archive*, 2011:617, 2011. `http://eprint.iacr.org/2011/617.pdf`.

[11] G. Ateniese, J. Camenisch, S. Hohenberger, and B. de Medeiros. Practical group signatures without random oracles. *IACR Cryptology ePrint Archive*, 2005:385, 2005.

[12] M. Backes, D. Fiore, and R. M. Reischuk. Verifiable delegation of computation on outsourced data. In *ACM Conference on Computer and Communications Security*, pages 863–874, 2013.

[13] S. Barker, A. Mishra, D. Irwin, E. Cecchet, P. Shenoy, and J. Albrecht. Smart*: An open data set and tools for enabling research in sustainable homes. In *1st KDD Workshop on Data Mining Applications In Sustainability*, 2011.

[14] P. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In B. Preneel and S. Tavares, editors, *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer Berlin Heidelberg, 2006.

[15] G. Barthe, G. Danezis, B. Grégoire, C. Kunz, and S. Z. Béguelin. Verified computational differential privacy with applications to smart metering. In *CSF*, pages 287–301, 2013.

[16] M. Bellare. Practice-oriented provable security. In *Lectures on Data Security, Modern Cryptology in Theory and Practice, Summer School, Aarhus, Denmark, July 1998*, pages 1–15, London, UK, UK, 1999. Springer-Verlag.

[17] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73. ACM, 1993.

[18] F. Benhamouda, M. Joye, and B. Libert. A new framework for privacy-preserving aggregation of time-series data. `https://hal.inria.fr/hal-01181321`, 2015.

[19] E. Bertino, I. N. Fovino, and L. P. Provenza. A framework for evaluating privacy preserving data mining algorithms*. *Data Min. Knowl. Discov.*, 11(2):121–154, Sept. 2005.

[20] E. Bertino, D. Lin, and W. Jiang. A survey of quantification of privacy preserving data mining algorithms. In *Privacy-preserving data mining*, pages 183–205. Springer US, 2008.

[21] I. Bilogrevic, J. Freudiger, E. De Cristofaro, and E. Uzun. What's the gist? privacy-preserving aggregation of user profiles. In *Computer Security-ESORICS 2014*, pages 128–145. Springer International Publishing, 2014.

[22] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: The sulq framework. In *Proceedings of the Twenty-fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '05, pages 128–138, New York, NY, USA, 2005. ACM.

[23] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill. Order-preserving symmetric encryption. In *EUROCRYPT*, pages 224–241, 2009.

[24] A. Boldyreva, N. Chenette, and A. O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *CRYPTO*, pages 578–595, 2011.

[25] J. Bolot, N. Fawaz, S. Muthukrishnan, A. Nikolov, and N. Taft. Private decayed predicate sums on streams. In *Proceedings of the 16th International Conference on Database Theory*, ICDT '13, pages 284–295, New York, NY, USA, 2013. ACM.

[26] D. Boneh and X. Boyen. Short signatures without random oracles and the sdh assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, 2008.

[27] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Advances in Cryptology–CRYPTO 2004*, pages 41–55. Springer Berlin Heidelberg, 2004.

[28] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology-CRYPTO 2001*, pages 213–229. Springer Berlin Heidelberg, 2001.

[29] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *EUROCRYPT*, pages 416–432, 2003.

[30] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In *Advances in Cryptology-ASIACRYPT 2001*, pages 514–532. Springer Berlin Heidelberg, 2001.

[31] D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In Y. Ishai, editor, *Theory of Cryptography*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273. Springer Berlin Heidelberg, 2011.

[32] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, pages 56–72, 2004.

[33] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[34] J. Cao, Q. Xiao, G. Ghinita, N. Li, E. Bertino, and K.-L. Tan. Efficient and accurate strategies for differentially-private sliding window queries. In *Proceedings of the 16th International Conference on Extending Database Technology*, EDBT '13, pages 191–202, New York, NY, USA, 2013. ACM.

[35] C. Castelluccia, A. C.-F. Chan, E. Mykletun, and G. Tsudik. Efficient and provably secure aggregation of encrypted data in wireless sensor networks. *ACM Trans. Sen. Netw.*, 5(3):20:1–20:36, June 2009.

[36] C. Castelluccia, E. Mykletun, and G. Tsudik. Efficient aggregation of encrypted data in wireless sensor networks. In *Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on*, pages 109–117, July 2005.

[37] D. Catalano and D. Fiore. Practical homomorphic macs for arithmetic circuits. In *EUROCRYPT*, pages 336–352, 2013.

[38] D. Catalano, D. Fiore, and B. Warinschi. Homomorphic signatures with efficient verification for polynomial functions. In *Advances in Cryptology–CRYPTO 2014*, pages 371–389. Springer Berlin Heidelberg, 2014.

[39] D. Catalano, A. Marcedone, and O. Puglisi. Authenticating computation on groups: New homomorphic primitives and applications. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, pages 193–212, 2014.

[40] T.-H. H. Chan, M. Li, E. Shi, and W. Xu. Differentially private continual monitoring of heavy hitters from distributed streams. In *Proceedings of the 12th International Conference on Privacy Enhancing Technologies*, PETS'12, pages 140–159, Berlin, Heidelberg, 2012. Springer-Verlag.

[41] T.-H. H. Chan, E. Shi, and D. Song. Private and continual release of statistics. *ACM Trans. Inf. Syst. Secur.*, 14(3):26:1–26:24, Nov. 2011.

[42] T.-H. H. Chan, E. Shi, and D. Song. Privacy-preserving stream aggregation with fault tolerance. In *Financial Cryptography*, pages 200–214, 2012.

[43] S. Chawla, C. Dwork, F. McSherry, A. Smith, and H. Wee. Toward privacy in public databases. In *Theory of Cryptography*, pages 363–385. Springer, 2005.

[44] S. Chawla, C. Dwork, F. McSherry, and K. Talwar. On the utility of privacy-preserving histograms. *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, 2005.

[45] R. Chen, I. E. Akkus, and P. Francis. Splitx: High-performance private analytics. *SIGCOMM Comput. Commun. Rev.*, 43(4):315–326, Aug. 2013.

[46] R. Cheng, J. Yan, C. Guan, F. Zhang, and K. Ren. Verifiable searchable symmetric encryption from indistinguishability obfuscation. In *Proceedings of the 10th ACM Symposium*

on Information, Computer and Communications Security, ASIA CCS '15, pages 621–626, New York, NY, USA, 2015. ACM.

[47] S. G. Choi, J. Katz, R. Kumaresan, and C. Cid. Multi-client non-interactive verifiable computation. In *Proceedings of the 10th Theory of Cryptography Conference on Theory of Cryptography*, TCC'13, pages 499–518, Berlin, Heidelberg, 2013. Springer-Verlag.

[48] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for privacy preserving distributed data mining. *ACM SIGKDD Explorations*, 4:2003, 2003.

[49] F. Cohen. Introductory information protection, 1995.

[50] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (3. ed.)*. MIT Press, 2009.

[51] T. Dalenius. Finding a needle in a haystack - or identifying anonymous census record. *Journal of Official Statistics*, 2:329–336, 1986.

[52] G. Danezis, C. Fournet, M. Kohlweiss, and S. Zanella-Béguelin. Smart meter aggregation via secret-sharing. In *Proceedings of the First ACM Workshop on Smart Energy Grid Security*, SEGS '13, pages 75–80, New York, NY, USA, 2013. ACM.

[53] B. Defend and K. Kursawe. Implementation of privacy-friendly aggregation for the smart grid. In *Proceedings of the First ACM Workshop on Smart Energy Grid Security*, SEGS '13, pages 65–74, New York, NY, USA, 2013. ACM.

[54] A. W. Dent. A note on game-hopping proofs. *IACR Cryptology ePrint Archive*, 2006:260, 2006.

[55] C. Dwork. Differential privacy. In *Automata, languages and programming*, pages 1–12. Springer, 2006.

[56] C. Dwork. Differential privacy: A survey of results. In *Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.

[57] C. Dwork. Differential privacy in new settings. In *SODA*, pages 174–183. SIAM, 2010.

[58] C. Dwork. A firm foundation for private data analysis. *Commun. ACM*, 54(1):86–95, Jan. 2011.

[59] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *Proceedings of the 24th Annual International Conference on The Theory and Applications of Cryptographic Techniques*, EUROCRYPT'06, pages 486–503, Berlin, Heidelberg, 2006. Springer-Verlag.

[60] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography*, TCC'06, pages 265–284, Berlin, Heidelberg, 2006. Springer-Verlag.

[61] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum. Differential privacy under continual observation. In *Proceedings of the 42nd ACM symposium on Theory of computing*, pages 715–724. ACM, 2010.

[62] C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases. In *Advances in Cryptology–CRYPTO 2004*, pages 528–544. Springer, 2004.

[63] Z. Erkin and G. Tsudik. Private computation of spatial and temporal power consumption with smart meters. In *ACNS*, pages 561–577, 2012.

[64] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the Twenty-second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '03, pages 211–222, New York, NY, USA, 2003. ACM.

[65] L. Fan and L. Xiong. Real-time aggregate monitoring with differential privacy. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2169–2173. ACM, 2012.

[66] D. M. Freeman. Improved security for linearly homomorphic signatures: A generic framework. In *Public Key Cryptography - PKC 2012 - 15th International Conference on Practice*

and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012. Proceedings, pages 697–714, 2012.

[67] G. Frey, M. Muller, and H. G. Ruck. The tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. *IEEE Trans. Inf. Theor.*, 45(5):1717–1719, Sept. 2006.

[68] S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. *Discrete Appl. Math.*, 156(16):3113–3121, Sept. 2008.

[69] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 40–49, Oct 2013.

[70] D. Genkin, A. Shamir, and E. Tromer. RSA key extraction via low-bandwidth acoustic cryptanalysis. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 444–461, 2014.

[71] A. Ghosh, T. Roughgarden, and M. Sundararajan. Universally utility-maximizing privacy mechanisms. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 351–360, New York, NY, USA, 2009. ACM.

[72] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikäinen. On private scalar product computation for privacy-preserving data mining. In *Proceedings of the 7th international conference on Information Security and Cryptology*, ICISC'04, pages 104–120, Berlin, Heidelberg, 2005. Springer-Verlag.

[73] L. R. Goldberg. An Alternative "Description of Personality": the Big-Five Factor Structure. *Journal of Personality and Social Psychology*, 59(6):1216–29, Dec. 1990.

[74] O. Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications.* Cambridge University Press, 2004.

[75] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807, Aug. 1986.

[76] S. Goldwasser, S. D. Gordon, V. Goyal, A. Jain, J. Katz, F. Liu, A. Sahai, E. Shi, and H. Zhou. Multi-input functional encryption. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 578–602, 2014.

[77] S. Goldwasser and S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pages 365–377, 1982.

[78] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[79] S. Goryczka, L. Xiong, and V. Sunderam. Secure multiparty aggregation with differential privacy: A comparative study. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, EDBT '13, pages 155–163, New York, NY, USA, 2013. ACM.

[80] C. P. L. Gouvêa and J. López. High speed implementation of authenticated encryption for the msp430x microcontroller. In *Proceedings of the 2nd international conference on Cryptology and Information Security in Latin America*, LATINCRYPT'12, pages 288–304, Berlin, Heidelberg, 2012. Springer-Verlag.

[81] C. Guan, K. Ren, F. Zhang, F. Kerschbaum, and J. Yu. A symmetric-key based proofs of retrievability supporting public verification. In *ESORICS 2015*.

[82] F. Günther, M. Manulis, and A. Peter. Privacy-enhanced participatory sensing with collusion resistance and data aggregation. In *Cryptology and Network Security - 13th International Conference, CANS 2014, Heraklion, Crete, Greece, October 22-24, 2014. Proceedings*, pages 321–336, 2014.

[83] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten. Lest we remember: Cold-boot attacks on encryption keys. *Commun. ACM*, 52(5):91–98, May 2009.

[84] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proc. VLDB Endow.*, 3(1-2):1021–1032, Sept. 2010.

[85] V. S. Iyengar. Transforming data to satisfy privacy constraints. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 279–288, New York, NY, USA, 2002. ACM.

[86] M. Jawurek, M. Johns, and K. Rieck. Smart metering de-pseudonymization. In *Proceedings of the 27th Annual Computer Security Applications Conference*, ACSAC '11, pages 227–236, New York, NY, USA, 2011. ACM.

[87] M. Jawurek and F. Kerschbaum. Fault-tolerant privacy-preserving statistics. In *Privacy Enhancing Technologies*, pages 221–238, 2012.

[88] O. P. John, E. M. Donahue, and R. L. Kentle. The big five inventory versions 4a and 54. *Berkeley: University of California, Berkeley, Institute of Personality and Social Research*, 1991.

[89] O. P. John, L. P. Naumann, and C. J. Soto. Paradigm shift to the integrative big five trait taxonomy. *Handbook of personality: Theory and research*, 3:114–158, 2008.

[90] W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1–1, 1984.

[91] A. Joux. A one round protocol for tripartite diffie–hellman. *Journal of Cryptology*, 17(4):263–276, 2004.

[92] M. Joye and B. Libert. A scalable scheme for privacy-preserving aggregation of time-series data. In *Financial Cryptography*, 2013.

[93] S. Kamara, P. Mohassel, and M. Raykova. Outsourcing multi-party computation. Cryptology ePrint Archive, Report 2011/272, 2011. `http://eprint.iacr.org/`.

[94] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Trans. on Knowl. and Data Eng.*, 16(9):1026–1037, Sept. 2004.

[95] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 99–106, Nov 2003.

[96] G. Kellaris and S. Papadopoulos. Practical differential privacy via grouping and smoothing. In *Proceedings of the 39th international conference on Very Large Data Bases*, pages 301–312. VLDB Endowment, 2013.

[97] G. Kellaris, S. Papadopoulos, X. Xiao, and D. Papadias. Differentially private event sequences over infinite streams. *Proceedings of the VLDB Endowment*, 7(12), 2014.

[98] A. Kiayias. History of the scytale, 2013.

[99] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[100] P. C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, pages 104–113, 1996.

[101] P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, pages 388–397, 1999.

[102] K. Kursawe, G. Danezis, and M. Kohlweiss. Privacy-friendly aggregation for the smart-grid. In *PETS*, pages 175–191, 2011.

[103] I. Leontiadis, K. Elkhyaoui, and R. Molva. Private and dynamic time-series data aggregation with trust relaxation. In *CANS*, Lecture Notes in Computer Science. Springer, 2014.

[104] Q. Li and G. Cao. Efficient privacy-preserving stream aggregation in mobile sensing with low aggregation error. In *PETS*, pages 60–81, 2013.

[105] J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37:145–151, 1991.

[106] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *CRYPTO*, pages 36–54, 2000.

[107] M. Lisovich, D. Mulligan, and S. Wicker. Inferring personal information from demand-response systems. *Security Privacy, IEEE*, 8(1):11–20, Jan.-Feb.

[108] K. Liu, H. Kargupta, and J. Ryan. Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *Knowledge and Data Engineering, IEEE Transactions on*, 18(1):92–106, Jan 2006.

[109] B. Lynn. The stanford pairing based crypto library. `http://crypto.stanford.edu/pbc`.

[110] A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In H. Heys and C. Adams, editors, *Selected Areas in Cryptography*, volume 1758 of *Lecture Notes in Computer Science*, pages 184–199. Springer Berlin Heidelberg, 2000.

[111] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval.* Cambridge University Press, New York, NY, USA, 2008.

[112] S. McLaughlin, P. McDaniel, and W. Aiello. Protecting consumer privacy from electric load monitoring. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, pages 87–98, New York, NY, USA, 2011. ACM.

[113] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '07, pages 94–103, Washington, DC, USA, 2007. IEEE Computer Society.

[114] V. Miller. Use of elliptic curves in cryptography. In H. Williams, editor, *Advances in Cryptology-CRYPTO 1985 Proceedings*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer Berlin Heidelberg, 1986.

[115] D. Mir, S. Muthukrishnan, A. Nikolov, and R. N. Wright. Pan-private algorithms via statistics on sketches. In *Proceedings of the Thirtieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '11, pages 37–48, New York, NY, USA, 2011. ACM.

[116] I. Mironov, O. Pandey, O. Reingold, and S. P. Vadhan. Computational differential privacy. In *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, pages 126–142, 2009.

[117] A. Miyaji, M. Nakabayashi, and S. TAKANO. New explicit conditions of elliptic curve traces for fr-reduction, 2001.

[118] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, SP '08, pages 111–125, Washington, DC, USA, 2008. IEEE Computer Society.

[119] S. R. M. Oliveira and et al. Privacy-preserving clustering by object similarity-based representation and dimensionality reduction transformation. In *ICDM 2004*, pages 21–30, 2004.

[120] S. R. M. Oliveira and O. R. Zaïane. Privacy preserving clustering by data transformation. *JIDM*, 1(1):37–52, 2010.

[121] M. Önen and R. Molva. Secure data aggregation with multiple encryption. In *Proceedings of the 4th European Conference on Wireless Sensor Networks*, EWSN'07, pages 117–132, Berlin, Heidelberg, 2007. Springer-Verlag.

[122] D. Page, N. Smart, and F. Vercauteren. A comparison of mnt curves and supersingular curves. *Applicable Algebra in Engineering, Communication and Computing*, 17(5):379–392, 2006.

[123] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, pages 223–238, 1999.

[124] V. Rastogi and S. Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, SIGMOD '10, pages 735–746, New York, NY, USA, 2010. ACM.

[125] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[126] P. Rogaway and T. Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In B. Roy and W. Meier, editors, *Fast Software Encryption*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer Berlin Heidelberg, 2004.

[127] I. Rouf, H. Mustafa, M. Xu, W. Xu, R. Miller, and M. Gruteser. Neighborhood watch: security and privacy analysis of automatic meter reading systems. In *Proceedings of the 2012 ACM conference on Computer and communications security*, CCS '12, pages 462–473, New York, NY, USA, 2012. ACM.

[128] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Trans. on Knowl. and Data Eng.*, 13(6):1010–1027, Nov. 2001.

[129] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, 1998.

[130] K. P. Samuel Kotz, Tomasz J. Kozubowski. The laplace distribution and generalizations: A revisit with applications to communications, economics, engineering and finance. 2001.

[131] R. Sarathy and K. Muralidhar. Evaluating laplace noise addition to satisfy differential privacy for numeric data. *Trans. Data Privacy*, 4(1):1–17, Apr. 2011.

[132] E. Shi, T.-H. H. Chan, E. G. Rieffel, R. Chow, and D. Song. Privacy-preserving aggregation of time-series data. In *NDSS*, 2011.

[133] R. Shokri. Privacy games: Optimal user-centric data obfuscation. *Proceedings on Privacy Enhancing Technologies*, 2015(2):1–17, 2015.

[134] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, pages 256–266, 1997.

[135] V. Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, 2004:332, 2004.

[136] S. Singh. The alternative history of public-key cryptography, 1999.

[137] A. Sorniotti and R. Molva. Secret handshakes with revocation support. In D. Lee and S. Hong, editors, *Information, Security and Cryptology-ICISC 2009*, volume 5984 of *Lecture Notes in Computer Science*, pages 274–299. Springer Berlin Heidelberg, 2010.

[138] L. Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, Oct. 2002.

[139] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *SIGMOD Rec.*, 33(1):50–57, Mar. 2004.

[140] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. *IEEE Trans. on Knowl. and Data Eng.*, 23(8):1200–1214, Aug. 2011.