Privacy and Accountability in Cloud Computation and Storage

by

HARDIK GAJERA
201421004

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY
to

DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY

March, 2021

Declaration

I hereby declare that

i) the thesis comprises of my original work towards the degree of Doctor of Philosophy at Dhirubhai Ambani Institute of Information and Communication Technology and has not been submitted elsewhere for a degree,

ii) due acknowledgment has been made in the text to all the reference material used.

[illegible]

Hardik Gajera

Certificate

This is to certify that the thesis work entitled PRIVACY AND ACCOUNTABILITY IN CLOUD COMPUTATION AND STORAGE has been carried out by HARDIK GAJERA for the degree of Doctor of Philosophy at *Dhirubhai Ambani Institute of Information and Communication Technology* under my supervision.

Manik Lal Das.

Prof. Manik Lal Das
Thesis Supervisor

Acknowledgments

The completion of this thesis would not have been possible without the support and encouragement of several special people. Hence, I would like to take this opportunity to show my gratitude to those who have assisted me in a myriad of ways.

The journey of a Ph.D. requires guidance and assistance from many people. I am grateful to my advisor and mentor, Prof. Manik Lal Das, who guided me and believed in me throughout the journey. In spite of his busy schedule and responsibilities, he was always there to support me and guide me at every step patiently. Without his continuous optimism concerning, encouragement and support, it would not be easy to complete this journey. His attitude to achieving excellence has helped me to achieve optimal results.

I like to thank the whole team of DA-IICT, who directly or indirectly supported me throughout this journey. I especially want to thank Prof. Anish Mathuria, Prof. Prasanjeet Majumdar, and Prof. Saurish Dasgupta, who provided their valuable suggestions to strengthen my research work.

Further, I would like to thank Prof. Pascal Lafourcade, Dr. David Gérault, Dr. Matthieu Giraud, and Dr. Xavier Bultel at Université Clermont Auvergne, CNRS, LIMOS, Clermont-Ferrand, France, for their fruitful discussion through collaboration. I am also grateful to Prof. Pascal Lafourcade and Dr. Matthieu Giraud for making me comfortable during my stay in Clermont-Ferrand, France.

A special thanks to my friends and roommates, Prathmesh Madhu, Ketul Parikh, Ainish Dave, and Moiz Palitanawala, for their much-needed support during the initial days of the journey. I thank all my co-researchers in DA-IICT, especially Dr. Sarita Agarwal, Nidhi Desai, Rishikant Rajdeepak, Dr. Parth

Mehta, and Pranav Verma, who helped me directly or indirectly in fulfilling my dream.

Finally, I would like to thank my family who always supported throughout my Ph.D. journey. To my late brother thank you for taking care of all the family responsibilities and allowing me to focus on my Ph.D. work. To my parents thank you for your love and support throughout my life. To my wife thank you for your support and encouragement during final stage of this journey.

Contents

Abstract

Cloud computing is a cost-effective computing paradigm for convenient, on-demand data access to a shared pool of configurable computing resources such as networks, servers, storage, applications, and services. While providing pay-per-use on-demand service to the service consumer, the cloud service provider should minimize computation error on data stored in a cloud storage server. If there is any error, one can recompute or restore the data, but a user cannot detect an error. There have been some approaches like verifiable computation, secure computation, and multi-party computation, which may find a useful application in cloud storage/computation services; however, most of these approaches assume that the computation's logic on data is public. The problem becomes challenging when the logic of computation is hidden to clients. In this thesis, the notion of *Private Polynomial Evaluation* (PPE) is defined along with a new security model *"indistinguishability against chosen function attack"* (IND-CFA), where an adversary tries to guess which polynomial is used among two polynomials of his/her choice. The existing schemes on verifiable computation with hidden polynomial are not IND-CFA secure. The proposed scheme, Private IND-CFA Polynomial Evaluation (PIPE), is the first IND-CFA secure PPE. It is IND-CFA secure under the decisional Diffie-Hellman (DDH) assumption in the random oracle model.

In a public cloud system, the cloud requires to verify a user's identity before providing any service. Depending on the nature of applications, the cloud server's computation may require to preserve the user's identity from the cloud. For example, in healthcare applications, it is advisable to preserve the privacy of users and the privacy of the data. Another proposed scheme Verifiable Oblivious IND-CFA Polynomial Evaluation scheme (VIP-POPE), in which the server computes over encrypted data, and provides proof of computation, preserve the privacy of the user's data. The proposed scheme VIP-POPE preserves the user's data privacy and is shown secure against IND-CFA adversary and Client's Privacy-Indistiguishability (CPI) security under standard security models. The user's identity verification with the cloud is not considered in the VIP-POPE scheme. In the proposed privacy-preserving verifiable computation (PriVC) scheme, the server can compute on the user's encrypted data and provide the proof of computation that can be verified by the user. The PriVC preserves the user's privacy and ensures the undeniability of the service offered and the service consumed. The PriVC scheme is secure under IND-CFA, and the proof of computation is non-repudiable and unforgeable in the standard model.

Verification of the integrity of the data stored on the public cloud is another important aspect of cloud services. Users generally do not keep a local copy of the data after uploading it over the cloud, and it is hard to remember whole data. In such a scenario, modification or deletion of a small part of the data may go unnoticed. Even though public clouds put lots of effort into maintaining and securing their storage server to ensure an efficient and error-free storage service, one can not rule out the possibility of corruption in data due to human or machine error. Many schemes, like proof of storage (POS), proof of data possession (PDP), proof of retrievability (POR), have been introduced in the literature to address the storage issue. Although there are a few proof of storage with data deduplication (POSD) schemes exist, these schemes are inefficient for real-world applications. In data deduplication, the cloud keeps only one copy of multiple duplicate copies of data that ensures an efficient storage system, and therefore, one cannot ignore it in the cloud storage system. The existing schemes consider only file-level deduplication, which does not improve storage efficiency much compared to block-level data deduplication. Using the idea of VIP-POPE scheme, a new efficient scheme, Data Deduplication with Proof of Storage DPoS, is proposed for proof of storage scheme with data deduplication at the block-level. Imagine a file as a polynomial by breaking the file in fixed-sized blocks and considering each block as a polynomial coefficient. With file as a polynomial, one can use the idea of $\mathsf{VIP} - \mathsf{POPE}$ scheme for proof of storage verification. The unforgeability security of the proposed scheme is proven under the discrete logarithm assumption. The DPoS scheme is efficient in comparison to other related schemes.

List of Principal Symbols and Acronyms

$\mathcal{A}$	Adversary
VPOPE	Verifiable and Private Oblivious Polynomial Evaluation
$\mathsf{IND\text{-}CFA}$	Indistinguishable against Chosen Function Attack
$\mathsf{IND\text{-}CPA}$	Indistingushable against Chosen Plaintext Attack
$\textsc{poly}(\eta)$	Polynomial in η
$\xleftarrow{\$}$	Randomly selected
$\mathrm{Adv}^{abc}_{\Pi,\mathcal{A}}$	Advantage of adversary $\mathcal{A}$ against the experiment abc for scheme Π
t-SDDH	t-Strong Decisional Diffie-Hellman
CPI	Client's Privacy Indistinguishability
CSP	Cloud Service Provider
MLE	Message Locked Encryption
NIZKP	Non-interactive ZKP
OPE	Oblivious Polynomial Evaluation
PDP	Provable Data Possession
PHR	Patient's Health Record
PKE	Public Key Encryption
POR	Proof of Retrievability

POSD	Proof of Storage with Deduplication
PP	Polynomial Protection
PPE	Private Polynomial Evaluation
PPT	Probabilistic Polynomial Time
PRV-CDA	Privacy against Chosen Distribution Attack
QAP	Quadratic Arithmetic Programs
QS	Query Soundness
RCE	Randomized Convergent Encryption
STC	Strong Tag Consistency
TA	Trusted Authority
TC	Tag Consistency
TTP	Trusted Third Party
UNF	Unforgeability
VIP-POPE	Verifiable IND-CFA Paillier based Private Oblivious Polynomial Evaluation
vk	Verification Key
WPP	Weak Polynomial Protection
ZKP	Zero-Knowledge Proof

List of Tables

List of Figures

CHAPTER 1

Introduction

Cloud infrastructure has been widely used for managing data and services of applications owned by industries, organizations and individuals. The main advantages of storing data on public cloud are data availability, reliability and ease of data sharing at a reasonable price [61]. Storing data, in particular sensitive data, on public cloud storage become a potential target of attacker, which if gets leaked or copromized then data owner and service provider will not be able to protect user's data from malicious intention. Even though cloud services have certain advantages mainly in terms of implementation cost, there is a trust issue regarding the services as the service provider is a third party [62]. Cloud services often do not compromise on security of their services. If there is a breach of any kind for any cloud service, then it will affect their reputation and business. However, what happens if there is a small breach at user level and cannot be detected by the user? Clearly, this will not affect the cloud service provider but it may affect the user. Recently, a freelance photographer Dave Cooper lost around $100,000$ video clips due to a bug in Adobe Premiere Pro [52]. Fortunately, he was able to detect deleted files as it is significant in numbers. Now imagine only one of the $100,000$ clips gets deleted due to a bug. Will Dave be able to detect it? Clients (People or organizations) store their data on the cloud without keeping any local copy of it. After some time, the client cannot detect a slight modification or deletion in the data as it is hard to remember complete data. It is beneficial for the cloud service provider to hide data damages if the client cannot detect it. There is a similar issue regarding cloud computing as well. In their Terms and Conditions, all cloud service providers specify that they do not warrant that access to cloud computing service will be error-free or uninterrupted [53]. Here is the excerpt from Amazon Web Services(AWS) customer agreement: *"NO REPRESENTATIONS OR WARRANTIES OF ANY KIND ... THAT THE SERVICE OFFERINGS OR THIRD-PARTY CONTENT WILL BE UNINTERRUPTED, ERROR-FREE OR FREE OF HARMFUL COMPONENTS, AND (IV) THAT ANY CONTENT WILL BE SECURE OR NOT OTHERWISE LOST OR ALTERED"* [63].

1.1 Outsourced Computation

Typically, user trust the cloud server with computed result and makes personal or business related decision based on the computed result. An error in the result may force the user to make wrong decision. Based on the application or scenario, a wrong decision can affect the user or organization. For example, in healthcare related application, a result computed by the cloud server may be a vital component of a decision making process. If there is the slightest chance of error in the computation and the client cannot detect it then it may affect profoundly. In such scenario, the client should be able to verify the computed results.

Mathematical models are powerful tools that are used to make predictions about a system's behavior, for instance, climate or stock exchange. The idea is to collect a large set of data for a while and use it to build a function that allows predicting the evolution of the system in the future. This topic has practical applications in many disciplines such as physics, biology, earth science, meteorology, artificial intelligence, economics, sociology or political science. For example, it can be used to predict global warming in climatology, or the stock exchange behavior for economists. There are many approaches for mathematical modeling: different models can produce different prediction functions for the same system. Consider a company that collects and stores a massive set of data, for example about the state of the soil, such as humidity, acidity, temperature and mineral amount. Using it, it computes some function that predicts the state of the soil for next years. The clients are farmers who want to anticipate the state of the soil during the sowing periods to determine how much seeds to buy and when to plant them. The company gives its client access to the prediction function through a cloud server. A client can then interact with the server to evaluate the function on his data. For economic or property reasons, the company does not want the clients to be able to recover the prediction function nor the mathematical model. Figure 1.1 illustrates a general system model for delegation of prediction function to the cloud.

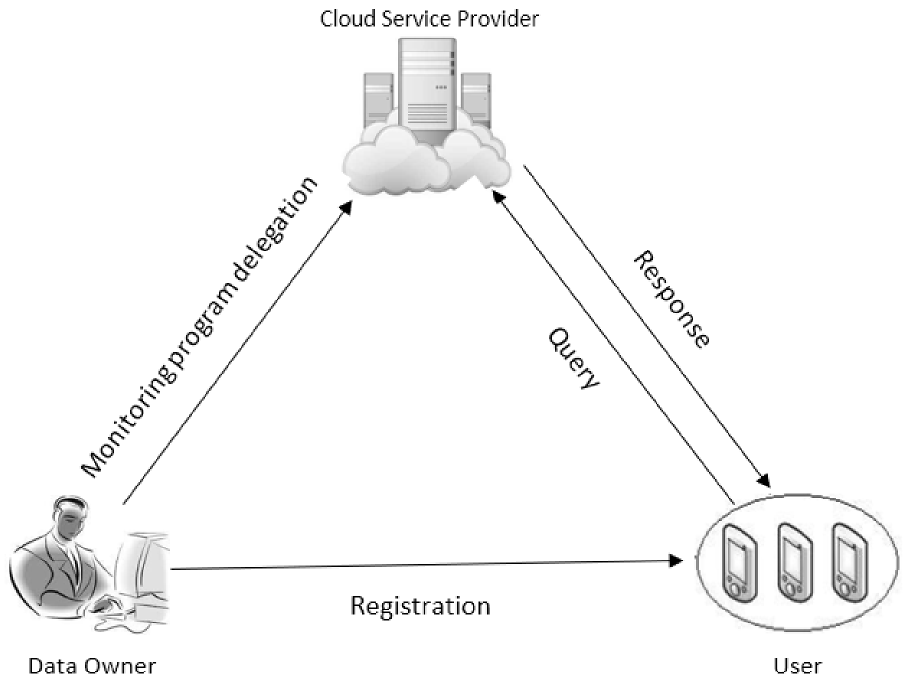


Figure 1.1: General system architecture for cloud services.

In recent times, several mobile health services have been proposed [54, 55, 56, 57, 74]. MediNet [54] discussed a mobile healthcare system that can personalize the self-care process for patients with both diabetes and cardiovascular disease. MediNet uses a reasoning engine to make recommendations to a patient based on current and previous readings from monitoring devices connected to the patient and on information that is known about the patient. HealthKiosk [55] proposed a family-based healthcare system that considers contextual information and alerting mechanisms for continuous monitoring of health conditions, where the system design of HealthKiosk has an important entity known as *sensor proxy* that acts as a bridge between the raw data sensed from the sensing device and the kiosk controller, and also acts as a data processing unit. In [56], a taxonomy of the strategies and types of health interventions have been discussed and implemented with mobile phones. Lin *et al* [74] proposed cloud-assisted privacy-preserving mobile health monitoring system to protect the privacy of users. Their scheme uses the key private proxy re-encryption technique by which the computational cost of the users is primarily done in the cloud server. The underlying problem is how to delegate the computation on a polynomial function to a server with following conditions:

1. The polynomial f remains secret;

2. The user can verify the computation done by the server.

To solve this problem, a new primitive PPE , for private polynomial evaluation, is proposed, which ensures the above two conditions. Figure 1.2 illustrates a PPE scheme, where x is the user data and $f(x)$ is the evaluation of the data x by the function f of the company. Moreover, the proof send by the server, and the verification key vk send by the company allow the user to verify the correctness of the delegated computations.

1.1.1 Verifiable Computation

The second part of the above mentioned problem is widely known as Verifiable Computation (VC). The *Verifiable Computation* (VC) is introduced by Gennaro *et al.* [13]. The aim of VC is to delegate a too costly computation to an untrusted third party. This third party returns the result of the computation and a proof of correctness which is easier to verify. The VC scheme proposed by Gennaro *et al.* is a non-interective fully homomorphic encryption based, publicly not verifiable and requires expensive offline setup work. Primitives where everyone can check the correctness of the computation are said to be *publicly verifiable* [20]. Parno *et al.* [20] proposed a non-interective publicly verifiable VC scheme called Pinocchio. Pinocchio is based on Quadratic Arithmetic Programs (QAP) and it is nearly practical as it supports both boolean and arithmetyic circuits. However, it requires expensive offline setup work. VC has given rise to a bunch of protocols [5, 10, 6, 19, 18]. In 2012, authors of [5] proposed a formal security model for VC. In [10], the authors propose a scheme for generic polynomials evaluations and generic matrix computations. Unlike our scheme, all these works consider that the algorithm (or the polynomial) used by the server is public and available to the verifier.

We found some earlier papers which consider keeping the function secret along with verifying computation [49, 14]. Kate *et al.* [14] propose a commitments to polynomials (CTP) where a user publishes some points (x, y) for a committed hidden polynomial f. The user can open the commitment *a posteriori* to reveal the polynomial. CTP is close to PPE: the verification key in a PPE scheme can be viewed as a commitment in a CTP scheme, the main difference is that this verification key is computed by a trusted party (the company) and the points are evaluated by an untrusted party (the server). The authors formalise the hardness of guessing the polynomial knowing less than k points. In this model, the polynomial is randomly chosen, then they does not consider the case where the adversary tries to distinguish the committed polynomial between two chosen polynomials as in our IND-CFA model. Moreover, Kate *et al.* design two CTP schemes in [14]. The first one is not IND-CFA since the commitment algorithm is deterministic. Moreover, our scheme Private IND-CFA Polynomial Evaluation (PIPE) can be used as a CTP scheme. Our scheme solves an open problem described by Kate *et al.* [14]: designing a scheme that is secure under a weaker assumption than t-SDDH .

Guo *et al.* [49] propose a scheme with similar security properties to delegate the computation of a secret health related function on the users' health record. The polynomials are explicitly assumed to have low coefficients and degree, which greatly reduces their randomness. However, the authors give neither security models nor proof. Later, we *et al.* [51] show that any user can guess the polynomial using the Lagrange's interpolation on several points. We propose a scheme where the degree k is hidden and claim that it does not suffer from this kind of attack. Clearly, hiding the degree k is useless and that no scheme can be secure when user query more than k points to the server. Moreover, the attack on these both schemes requires only one query to the server to gain the knowledge of the hidden polynomial. To the best of our knowledge, the security notion for *Indistinguishability Against Chosen Function Attack* (IND-CFA) is not defined in earlier

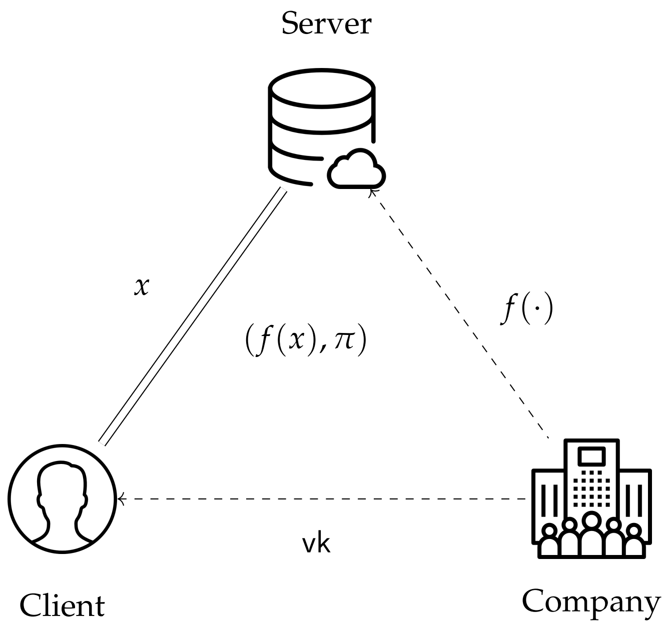


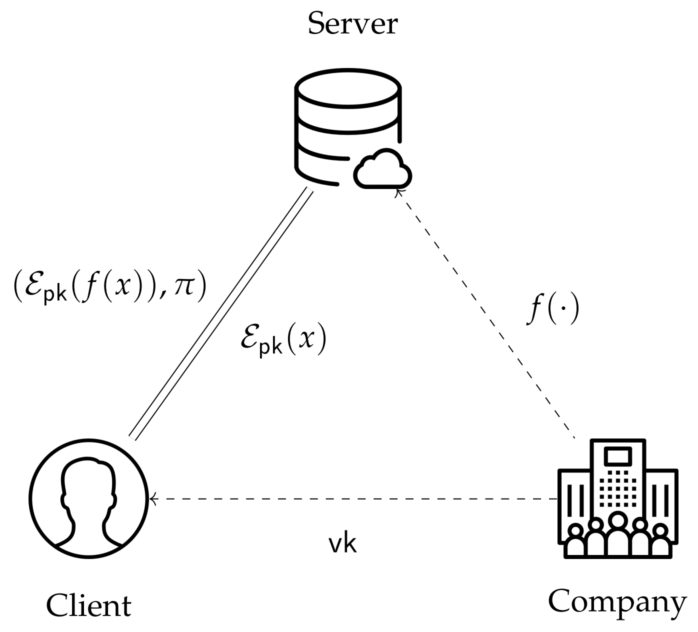
Figure 1.2: The architecture of a PPE scheme.

literature.

Consider the following scenario. A company uses a PPE for prediction functions as in the previous example. An attacker wants to guess which mathematical model is used by the company. Suppose this attacker gains access to some of the data used to build the prediction function, for instance by corrupting a technician. The attacker can build several prediction functions by using different mathematical models and the data he collected and then try to distinguish which of these functions is the one used by the company. Intuitively, in a secure PPE, this task should be as hard for the attacker as if the server only returned $f(x)$ and no additional information for verification. There is no formal security model which considers the above attack scenario. This notion is formalized as IND-CFA and the proposed scheme PIPE (Private IND-CFA Polynomial Evaluation) is secure under the new security notion.

1.1.2 Oblivious Polynomial Evaluation

In PPE schemes, clients send their data x in a plain form which can be very critical for the privacy of clients data. Moreover, in cloud based service application, the cloud is required to verify the identity of users before providing services. In



Server
$(\mathcal{E}_{\mathsf{pk}}(f(x)), \pi)$
$\mathcal{E}_{\mathsf{pk}}(x)$
$f(\cdot)$
vk
Client
Company

Figure 1.3: The architecture of a VPOPE scheme.

medical field, the system needs to preserve the privacy of users and user's data. Medical applications require that client send data concerning their health status. In a world, where insurers offer insurance at prices that depend on the health of customers, this can be problematic. Moreover, clients may not trust the server and do not want to send confidential data to it. In another point of view, clients do not know the security guarantees of the server infrastructure, and then it is legitimate to be concerned about computer attacks and data theft that the server may experience. To address this, a new primitive, called *Verifiable and Private Oblivious Polynomial Evaluation* (VPOPE) , is proposed. A VPOPE scheme allows the client to send their data to the server in an encrypted way. Hence, the server has no information on the client's data. As for PPE schemes, the server provides along with the encrypted evaluation $\mathcal{E}_{\mathsf{pk}}(f(x))$, a proof π allowing the client to verify the correctness of the result with the help the verification key vk. The new primitive is illustrated in Fig. 1.3.

VPOPE schemes are related to *Oblivious Polynomial Evaluation* (OPE) introduced by Naor and Pinkas [16]. In OPE, there are two parties. The first party, A, knows a secret function $f(\cdot)$ and the other one, B has a secret element x. The aim of OPE is that B receives $f(x)$ in such a way A learns nothing about the value x sent by B and that B learns nothing about the function $f(\cdot)$. OPE are used to solve different cryptographic problems as set membership, oblivious keyword search, and set intersection [15, 12, 11]. Although OPE and VPOPE are very similar, their difference lies in the fact that OPE does not consider the verifiability of the computation of $f(x)$, whereas it is a crucial point in VPOPE since the client does not trust the server. The proposed scheme Verifiable IND-CFA Paillier based Private Oblivious Polynomial Evaluation (VIP-POPE) scheme is IND-CFA secure and has efficient computation verification. The verification of the computed result is done by the client. The cost of verification must be as low as possible as client may have resource constrained device. The proposed VIP-POPE scheme takes less time for verification of the computed result as compared to other existing PPE schemes. Further, a Privacy preserving Verifiable Computation (PriVC) scheme is designed to preserve both, user's privacy and data privacy, along with verification of computation. The scheme further ensures undeniability of user in a successful transaction. This means once a user takes service of the cloud, it cannot deny it later. The PriVC scheme is IND-CFA secure and existentially unforgeable. The efficiency of PriVC is tested using realistic parameters in Sage 8.1.

The term Formal Verification looks similar to Verifiable Computation, but both are nor related. Formal Verification [79] techniques are used to check the correctness of a protocol or an algorithm concerning a specific property using either formal methods of mathematics or tools (like ProVerif [81], CryptoVerif [80], etc.).

1.2 Outsourced Storage

Similar to computation, users have to trust the cloud with our data that it will not be modified later. It is possible that specific data get deleted intentionally or unintentionally from the cloud due to very less frequency of usage. If you don't have a local copy of the data, then how will you make sure that the data stored on the cloud is still the same as the data stored initially? The proposed PPE and VIP-POPE schemes solves the problem of verification of computation over plain/encrypted data. However, clients of a healthcare company may want to keep a history of all their data by storing it on the cloud without keeping any local copy. Decision making process in medical field also depends on past data of a patient. A partial or full modification or deletion of past data stored on the cloud may affect a health related decision making process of the patient. It is necessary to have a certain mechanism to verify the integrity of the data stored on the cloud. Once you upload your data on the cloud storage, will it remain as it is over time? Without storing the whole data locally, how do you make sure that the file is not modified? There are several schemes in the literatures which provides integrity proof for the data stored in the cloud [32, 33, 34, 35, 36, 37]. However, most of them doesnot consider data deduplication. Along with the proof of stored data, the data deduplicaiton is also an important aspect in storage services. In data deduplicaiton process, the cloud keeps only one copy of multiple identical data for an efficient storage system. There are several data deduplication schemes proposed in the literatures for both at file level deduplicaiton and at block-level deduplication [24, 25, 26, 28, 27, 29, 31]. In literature, both these research areas, proof of storage and data deduplication, are running separately.

1.2.1 Data Dedupliaction and Proof of Storage

The data deduplication is an integral part of the cloud storage system. It is necessary to have data deduplicaiton in a proof of storage scheme. There are few schemes in literature which considers data deduplication along with proof of storage [38, 39, 40]. These schemes are studied thoroughly and found that all these schemes considers data deduplication at file level and are inefficient for practical implementation as it uses elliptic curve group and/or pairing operations for tag generation and challenge process. Moreover, none of these schemes provides security against duplicate faking attacks. As block-level data deuplication provides more efficient storage system as compared to file level data deduplication, it is necessary to have an efficient and secure proof of storage scheme which considers block-level data deduplication.

In 2007, Ateniese *et al.* [32] introduced Provable Data Possession (PDP) scheme which allows cloud storage server to provide probabilistic proof of stored data to its clients. After this, several PDP schemes were proposed focusing on the data integrity [33, 34, 35]. As PDP does not provide any recovery option for corrupted data, Juels *et al.* [43] introduced Proof of Retrievability (POR) scheme which allows recovery for corrupted data by using error correcting codes along with PDP. There is a challenge to design lightweight PoR scheme for resource constrained devices like smart phones and Internet of Things (IoT) devices. In 2015, Li *et al.* [36] proposed a lightweight scheme for data auditing but it is less efficient in terms of storage requirement and communication cost. To provide data integrity, PoR/PDP schemes are required to use additional data (tag) and this makes it vulnerable to several malicious threats like tag forgery attack, deletion attack, replace attack etc. In tag forgery attack, malicious server tries to hide corruption in the stored data by forging the proof using corrupted data [37]. The schemes which require only tag as proof for replacing or deleting data are vulnerable to deletion and replace attack. The main issue of existing PoR scheme is that very few of them considers data deduplication [38, 39, 40, 41, 42]. Data deduplication is a process of storing only one copy of data by removing redundant copies of the data.

In 2012, Zheng *et al.* [38] introduced Proof of Storage with Deduplication (POSD) scheme by combining PoR scheme with deduplication. Zheng *et al.* focused only on deduplication at file level and not at the block-level. Moreover, the cloud server does not verify the tag corresponding to the file and this leads to possible duplicate faking attacks. In 2017, Shin *et al.* [39] finds a week key attack in Zheng's scheme and proposed a fix for it. However, the fix still does not consider verifying tag and files at the time of file upload process. In 2013, Yuan *et al.* [40] proposed a new POSD scheme and their construction is similar Zheng's scheme. Yuan's scheme lacks tag consistency and focus on deduplication at file level. As per our knowledge there isn't any scheme which considers block-level data deduplication with tag consistency.

One can observe that verification of computation and proof of storage schemes are very similar. In verification of computation, user verify the computation of a polynomial over the challenge input and in proof of storage, user verify the data by verifying the computation of the data over the challenge input. Imagine the data stored on the cloud as a polynomial by considering each data block as a coefficient of the polynomial. If we use this polynomial in PPE scheme then one can verify the integrity of the data by verifying computation over the challenge input as an error in the data will result in the error in the computed result. Using the idea of VIP-POPE scheme, a new efficient scheme, Data Deduplication at block-level with proof of storage DPoS, is proposed. In our proposed scheme DPoS, The data is divided in fixed size blocks and treat each block as a coefficient of a polynomial. In challenge process, the cloud server provides the computed result of corresponding polynomial over the challenge input along with the proof. The user can verify the integrity of the data by verifying the computed result along with the proof. The proposed scheme provides security against duplicate faking attacks and proof forgery attacks. Our scheme is efficient as compared to existing proof of storage with data deduplication schemes.

1.3 Contribution of the thesis

The thesis addresses problem of verifying polynomial computation done by the cloud over plain as well as encrypted data. It addresses the problem of data integrity along with data deduplication in the cloud storage. The thesis also considers the issue related to cross-server ownership and privacy preserving identity verification between the client and the cloud server. The contribution of the thesis is as follows:

1. Verification of the service is essential as there is no reason to trust a public cloud. There are various schemes in the literature which focuses on the verification of computation of a polynomial, but they assume that the polynomial is public. We studied existing verifiable polynomial evaluation schemes, where polynomial is private. In chapter 3, we thoroughly analyse Guo *et. al'* verification scheme for healthcare system and provide details of single query attack. We further note that none of the existing scheme provides formal security models and we also found an attack which breaks

polynomial secrecy. In chapter 4, we formally define a new primitive PPE along with a new formal security model IND-CFA along with Polynomial Protection (PP) and Unforgeability for verification of computation with a hidden polynomial. We also define a primitive VPOPE for verifiable and oblivious polynomial evaluation. We define formal security models Client's Privacy - Indistinguishability (CPI) and Query Soundness (QS) for VPOPE.

2. In Chapter 5, we design a first secure PPE scheme called Private IND-CFA Polynomial Evaluation (PIPE). We prove security of PIPE with respect to security models indistinguishability against chosen function attack (IND-CFA), Polynomial Protection (PP), and Unforgeability. We also design an efficient and secure Verifiable Oblivious Polynomial Evaluation (VPOPE) scheme called $\mathsf{VIP-POPE}$ (for *Verifiable oblivious IND-CFA Polynomial Evaluation*) which requires only fixed number of exponentiation for verification of the computation. This scheme used homomorphic properties of Paillier's encryption scheme [17] to achieve encrypted polynomial evaluation. We implement $\mathsf{VIP-POPE}$ along with existing PPE scheme in Sage 8.1 and observe that $\mathsf{VIP-POPE}$ takes less time for verification of computation as compared to existing schemes.

3. In applications involving PPE scheme, the cloud verifies user's identity before providing services. In healthcare application, identity verification of a user by the cloud must preserve user's privacy. In chapter 6.2, we propose a scheme, PriVC, which considers verifiable computation over encrypted data along with privacy preserving identity verification. The pri scheme is IND-CFA secure and provides user's privacy, data privacy, verification of computation and user's non-repudiation properties.

4. In chapter 7, we discuss requirement for cross-server ownership. Using convergent encryption scheme with deterministic tags, we design a scheme, DeDOP, to tackle data deduplication and cross-server ownership issue in multiple storage server scenario. We considered user side encryption of data. We prove that the scheme DeDOP provides tag consistency at the client level.

5. In chapter 8, we present an efficient proof of storage with data deduplication at block-level scheme. We use the idea of $\mathsf{VIP} - \mathsf{POPE}$ to design a proof of storage scheme. The meta data used in proof of storage part helps during data deduplication process without any additional storage. The proposed scheme is secure against duplicate faking attacks and proof forgery attacks. We show that our scheme is efficient for practical implementation by implementing the proposed scheme along with existing POSD schemes with realistic parameters.

We have analyzed security of all proposed schemes with respect to corresponding standard security models. We have shown that our schemes are efficient for practical implementation by implementing all related schemes on same platform. The proposed schemes provide efficient and practical solution for verification of computation by secret polynomial and data integrity in public cloud.

CHAPTER 2

Background and Preliminaries

In this chapter, we provide some basic definitions [76] for making the remaining thesis self content. Most of these definitions are being used in security analysis, designing and correctness of the proposed constructions. The probabilistic polynomial time (PPT), negligible function and zero-knowledge proof (ZKP) are the notions used in the security analysis of the proposed schemes. The cryptanalysis of Guo *et al.*'s [49] scheme used Lagrange interpolation.

2.1 Mathematical Notion

Definition 1 (Probabilistic Polynomial Time (PPT) algorithm). *A probabilistic algorithm A is called PPT algorithm if there exist a polynomial $p(\cdot)$ such that the running time of A on input $x \in \{0,1\}^*$ is at most $p(|x|)$. The set of all PPT algorithms with respect to the security parameter η is denoted by* $\mathrm{POLY}(\eta)$.

Definition 2 (Negligible function [77]). *A function $\epsilon \colon \mathbb{N} \to \mathbb{R}^+$ is negligible in η if for every positive polynomial $p(\cdot)$ and sufficiently large η, $\epsilon(\eta) < p(\eta)^{-1}$.*

Lagrange's interpolation formula is used to find the single polynomial f of degree at most k from $k+1$ points (x_i, y_i) such that $f(x_i) = y_i$.

Definition 3 (Lagrange's interpolation). *Let k be an integer and F be a field. For all $i \in \{0, \dots, k\}$, let $(x_i, y_i) \in F^2$ such that for all $i_1, i_2 \in \{0, \dots, k\}$, $x_{i_1} \neq x_{i_2}$. There exists one and only one polynomial $f(\cdot)$ of degree at most k such that for all $i \in \{0, \dots, k\}$, $f(x_i) = y_i$. This polynomial is given by Lagrange's interpolation formula 2.1:*

$$f(x) = \sum_{i=0}^{k} \left(y_i \cdot \prod_{j=0, j \neq i}^{k} \frac{x - x_j}{x_i - x_j} \right) \tag{2.1}$$

In the following, we provide definition and security requirements of public key cryptosystems.

2.2 Public Key Encryption

Definition 4 (Public Key Encryption)**.** *A Public Key Encryption (PKE) scheme is defined by three algorithms $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ such that:*

$\mathcal{G}(\eta)$ *It returns a public/private key pair $(\mathsf{pk}, \mathsf{sk})$.*

$\mathcal{E}_{\mathsf{pk}}(m)$ *It returns the ciphertext c of the message m.*

$\mathcal{D}_{\mathsf{sk}}(c)$ *It returns the plaintext m from the ciphertext c.*

A PKE scheme $\Pi = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ is *indistinguishable under chosen-plaintext attack* ($\mathsf{IND\text{-}CPA}$) if for any probalistic polynomial time adversary $\mathcal{A}$, the difference between $\frac{1}{2}$ and the probability that $\mathcal{A}$ wins the $\mathsf{IND\text{-}CPA}$ experiment in Fig. 2.1 is negligible, where the oracle $\mathcal{E}_{\mathsf{pk}}(\mathsf{LR}_b(\cdot,\cdot))$ takes (m_0, m_1) as input and returns $\mathcal{E}_{\mathsf{pk}}(m_b)$. The standard definition of $\mathsf{IND\text{-}CPA}$ experiment allows the adversary to call this oracle only one time. However, in [3] authors prove that the two definitions of $\mathsf{IND\text{-}CPA}$ security are equivalent using an hybrid argument.

```
$\mathsf{Exp}^{\mathsf{IND\text{-}CPA}}_{\Pi,\mathcal{A}}(\eta)$:
  $b \xleftarrow{\$} \{0,1\}$ ;
  $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{G}(\eta)$ ;
  $b_* \leftarrow \mathcal{A}^{\mathcal{E}_{\mathsf{pk}}(\mathsf{LR}_b(\cdot,\cdot))}(\mathsf{pk})$ ;
  return $(b = b_*)$ .
```

Figure 2.1: $\mathsf{IND\text{-}CPA}$ experiment [3].

2.2.1 Paillier Cryptosystem

We now recall the generation, the encryption and decryption algorithms of the Paillier's public key encryption scheme [17] used in our scheme.

Key Generation. We denote by $\mathbb{Z}_n$, the ring of integers modulo n and by $\mathbb{Z}_n^\star$ the set of invertible elements of $\mathbb{Z}_n$. The public key pk of Paillier's encryption scheme is (n, g), where $g \in \mathbb{Z}_{n^2}^\star$ and $n = pq$ is the product of two prime numbers.

The corresponding secret key sk is (λ, μ), where λ is the least common multiple of $p-1$ and $q-1$ and $\mu = (L(g^\lambda \mod n^2))^{-1} \mod n$, where $L(x) = \frac{x-1}{n}$.

Encryption Algorithm. Let m be a message such that $m \in \mathbb{Z}_n$. Let g be an element of $\mathbb{Z}_{n^2}^\star$ and r be a random element of $\mathbb{Z}_n^\star$. We denote by $\mathcal{E}_{\mathsf{pk}}$ the encryption algorithm that produces the ciphertext c from a given plaintext m with the public key $\mathsf{pk} = (n, g)$ as follows: $c = g^m r^n \mod n^2$.

Decryption Algorithm. Let c be the ciphertext such that $c \in \mathbb{Z}_{n^2}$. We denote by $\mathcal{D}_{\mathsf{sk}}$ the decryption function of the plaintext c with the secret key $\mathsf{sk} = (\lambda, \mu)$ defined as follows: $m = L\left(c^\lambda \mod n^2\right) \cdot \mu \mod n$.

Paillier's cryptosystem is a partial homomorphic encryption scheme. We present these properties used in our scheme.

Homomorphic Addition of Plaintexts. Let m_1 and m_2 be two plaintexts in $\mathbb{Z}_n$. The product of the two associated ciphertexts with the public key $\mathsf{pk} = (n, g)$, denoted $c_1 = \mathcal{E}_{\mathsf{pk}}(m_1) = g^{m_1} r_1^n \mod n^2$ and $c_2 = \mathcal{E}_{\mathsf{pk}}(m_2) = g^{m_2} r_2^n \mod n^2$, is the encryption of the sum of m_1 and m_2. We also remark that: $\mathcal{E}_{\mathsf{pk}}(m_1) \cdot \mathcal{E}_{\mathsf{pk}}(m_2)^{-1} = \mathcal{E}_{\mathsf{pk}}(m_1 - m_2)$.

$$\begin{aligned}
\mathcal{E}_{\mathsf{pk}}(m_1) \times \mathcal{E}_{\mathsf{pk}}(m_2) &= c_1 \times c_2 \mod n^2 \\
&= (g^{m_1} \times r_1^n) \times (g^{m_2} \times r_2^n) \mod n^2 \\
&= \left(g^{m_1+m_2} \times (r_1 \times r_2)^n\right) \mod n^2 \\
&= \mathcal{E}_{\mathsf{pk}}(m_1 + m_2 \mod n) .
\end{aligned}$$

Specific Homomorphic Multiplication of Plaintexts. Let m_1 and m_2 be two plaintexts in $\mathbb{Z}_n$ and $c_1 \in \mathbb{Z}^*_{n^2}$ be the ciphertext of m_1 with the public key pk ($c_1 = \mathcal{E}_{\mathsf{pk}}(m_1)$). With Paillier's scheme, c_1 raised to the power of m_2 is the encryption of the product of the two plaintexts m_1 and m_2.

$$\begin{aligned}\mathcal{E}_{\mathsf{pk}}(m_1)^{m_2} &= c_1^{m_2} \mod n^2 \\ &= (g^{m_1} \times r_1^n)^{m_2} \mod n^2 \\ &= \left(g^{m_1 \cdot m_2} \times (r_1^{m_2})^n\right) \mod n^2 \\ &= \mathcal{E}_{\mathsf{pk}}(m_1 \times m_2 \mod n)\ .\end{aligned}$$

Theorem 1. *Paillier's cryptosystem is IND-CPA-secure if and only if the Decisional Composite Residuosity Assumption holds.*

Proof. The proof of Theorem 1 is given in Section 4 of [17]. □

To present our scheme, we first need to prove the following property on Paillier ciphertexts.

Property 1. *Let n be the product of two prime numbers, $x \in \mathbb{Z}_n$, and $g \in \mathbb{Z}^\star_{n^2}$. We set $pk = (n, g)$ a Paillier public key. Let $\{t_i\}_{i=1}^k$ such that for all $i \in \{1, \ldots, k\}$, we have $t_i = t_{i-1}^x \cdot r_i^n$ with $t_0 = g$, and $r_i \in \mathbb{Z}^\star_{n^2}$. Then for all $i \in \{1, \ldots, k\}$, $t_i = \mathcal{E}_{pk}(x^i)$.*

Proof. Considering the public key $\mathsf{pk} = (n, g)$. First, we remark that $t_1 = g^x r_1^n$, i.e., $t_1 = \mathcal{E}_{\mathsf{pk}}(x)$. We assume there exists $i \in \mathbb{N}^\star$ such that $t_i = \mathcal{E}_{\mathsf{pk}}(x^i) = g^{x^i} \cdot r_i^n$. Then, we prove that $t_{i+1} = \mathcal{E}_{\mathsf{pk}}(x^{i+1})$ as shown in the equation 2.2:

$$t_{i+1} = t_i^x \cdot r_{i+1}^n = \left(g^{x^i} \cdot r_i^n\right)^x \cdot r_{i+1}^n = g^{x^{i+1}} \cdot (r_i^x \cdot r_{i+1})^n = \mathcal{E}_{\mathsf{pk}}(x^{i+1}) \tag{2.2}$$

Since $r_i^x \cdot r_{i+1} \in \mathbb{Z}^\star_{n^2}$, t_{i+1} is a Paillier ciphertext of x^{i+1}. □

2.3 Homomorphic Encryption

Definition 5 (Homomorphic Encryption). *Any scheme $\varepsilon = (\mathsf{KeyGen}_\varepsilon, \mathsf{Encrypt}_\varepsilon, \mathsf{Decrypt}_\varepsilon)$ is said to be homomorphic with respect to a circuit C if and only if for any* *plaintexts $\pi_1, \pi_2, \ldots, \pi_n$ and ciphertexts $\psi_1, \psi_2, \ldots, \psi_n$ where $\psi_i = \mathsf{Encrypt}_\varepsilon(sk,\ \pi_i)$ the following equation holds true:*

$$C(\pi_1, \pi_2, \ldots, \pi_n) = \mathsf{Decrypt}_\varepsilon(sk, \mathsf{Evaluate}(C, \psi)).$$

2.3.1 DGHV scheme

Pisa *et al.* [58] proposed a somewhat homomorphic encryption scheme over large integers. Their scheme is an extension of DGHV scheme over bit operations [72].

Definition 6 (Extended DGHV scheme [58])**.** *The extended DGHV scheme contains following three algorithms:* KeyGen, Encryption, Decryption.

- $\mathsf{KeyGen}(\lambda)$*: The B is base parameter and the private key K_{priv} is a coprime to B in $[2^{\eta-1}, 2^{\eta})$ where $\eta = \mathcal{O}(\lambda^2)$.*

- $\mathsf{Encryption}(K_{priv}, m)$*: For encrypting $m \in [0, B)$, it takes a random $r \in (-2^{\rho}, 2^{\rho})$ and $s \in (0, 2^{\gamma}/K_{priv})$. The parameters $\gamma = \mathcal{O}(\lambda^5)$ and $\rho = \mathcal{O}(\lambda)$ as originally proposed in DGHV scheme [72]. It then compute ciphertext as follows:*

$$\psi = m + B \times r + s \times K_{priv}.$$

- $\mathsf{Decryption}(K_{priv}, \psi)$*: Using private key K_{priv}, it decrypts ψ as follows:*

$$m = (\psi \ mod\ K_{priv}) \ mod\ B.$$

2.4 Zero-Knowledge Proof

A zero-knowledge proof (ZKP) allows a prover knowing a witness to convince a verifier that a statement s is in a given language without leaking any information except s.

Definition 7 (NIZKP [9])**.** *A* non-interactive ZKP *(NIZKP) for a language $\mathcal{L}$ is a couple of algorithms* (Prove, Verify) *such that:*

$\mathsf{Prove}(s, w)$: *It outputs a proof π that $s \in \mathcal{L}$ using the witness w.*

$\mathsf{Verify}(s, \pi)$: *It checks whether π is a valid proof that $s \in \mathcal{L}$ and outputs a bit.*

A NIZKP proof verifies the following properties:

Completeness: For any statement $s \in \mathcal{L}$ and the corresponding witness w, we have that $\mathsf{Verify}(s, \mathsf{Prove}(s, w)) = 1$.

Soundness: There is no polynomial time adversary $\mathcal{A}$ such that $\mathcal{A}(\mathcal{L})$ outputs (s, π) such that $\mathsf{Verify}(s, \pi) = 1$ and $s \notin \mathcal{L}$ with non-negligible probability.

Zero-knowledge: A proof π *leaks no information, i.e.* there exists a probabilistic polynomial time algorithm Sim (called the *simulator*) such that outputs of $\mathsf{Prove}(s, w)$ and the outputs of $\mathsf{Sim}(s)$ follow the same probability distribution.

We use the ZKP given by Baudron *et al.* [2] to prove the plaintexts equality of $k \in \mathbb{N}$ Paillier ciphertexts. Let $\mathbb{Z}^{\star}_{n^2}$ be a multiplicative group, where n is the product of two prime numbers p and q, and $t_0 = g \in \mathbb{Z}^{\star}_{n^2}$. The language is the set of all statements $\{(t_i, t_{i-1}) \in \mathbb{Z}^{\star}_{n^2} \times \mathbb{Z}^{\star}_{n^2}\}_{i=1}^{k}$ such that it satisfies equation 2.3:

$$L(t_i^{\lambda} \mod n^2) \times (L(t_{i-1}^{\lambda} \mod n^2))^{-1} \mod n = m\,, \tag{2.3}$$

with $m \in \mathbb{Z}_n$, $\lambda = \operatorname{lcm}(p-1, q-1)$, and $L(x) = \frac{x-1}{n}$ for all $i \in \{1, \dots, k\}$.

We use the ZKP given by Baudron *et al.* [2] to prove the plaintexts equality of $k \in \mathbb{N}$ Paillier ciphertexts. Let $\mathbb{Z}^{\star}_{n^2}$ be a multiplicative group where n is the product of two prime numbers p and q. The language is the set of all statements $(t_1, \dots, t_k) \in (\mathbb{Z}^{\star}_{n^2})^k$ for $k \in \mathbb{Z}_{\geq 2}$ such that for all $i \in \{1, \dots, k\}$, $t_i = t_{i-1}^{x} \cdot r_i^{n} \mod n^2$ where $t_0 \in \mathbb{Z}^{\star}_{n^2}$ and $r_i \in \mathbb{Z}^{\star}_{n^2}$.

Since the ZKP given by Baudron *et al.* [2] is a sigma protocol, we can use the *Fiat-Shamir Transformation* [9] to obtain a NIZKP. We formally define this NIZKP called $\mathsf{DecPaillierEq}$.

Definition 8 ($\mathsf{DecPaillierEq}$ [2])**.** *Let n be the product of two prime numbers p and q and H be a hash function, $\mathcal{L}$ be the set of all $(t_1, \dots, t_k) \in (\mathbb{Z}^{\star}_{n^2})^k$ such that for all $i \in \{1, \dots, k\}$, $t_i = t_{i-1}^{x} \cdot r_i^{n} \mod n^2$ where $t_0 \in \mathbb{Z}^{\star}_{n^2}$ and $r_i \in \mathbb{Z}^{\star}_{n^2}$. We define the NIZKP $\mathsf{DecPaillierEq} = (\mathsf{Prove}, \mathsf{Verify})$ for $\mathcal{L}$ as follow:*

- $\mathsf{Prove}((t_1,\dots,t_k),\omega)$: *Using the witness* $\omega = (x, t_0, \{r_i\}_{i=1}^k)$, *it picks* $\rho \xleftarrow{\$} [0, 2^{\log(n)}]$ *and* $s_i \in \mathbb{Z}_n^*$ *for* $1 \leq i \leq k$, *and computes* $u_i = t_{i-1}^{\rho} \cdot s_i^n \mod n^2$ *for* $1 \leq i \leq k$. *Moreover, it computes* $w = \rho + x \cdot H(t)$ *and sets* $v_i = s_i \cdot r_i^{H(t)} \mod n$ *for* $1 \leq i \leq k$. *Finally, it outputs* $\pi_t = (w, \{u_i\}_{i=1}^k, \{v_i\}_{i=1}^k)$.
- $\mathsf{Verify}((t_1,\dots,t_k),\pi_t)$: *Using* $\pi_t = (w, \{u_i\}_{i=1}^k, \{v_i\}_{i=1}^k)$, *it verifies if* $w \in [0, 2^{\log(n)}]$, *and if* $t_{i-1}^{w} \cdot v_i^n = u_i \cdot t_i^{H(t)} \mod n^2$ *for* $1 \leq i \leq k$. *Then it outputs* 1, *else* 0.

Moreover, Baudron *et al.* [2] prove the following theorem.

Theorem 2. DecPaillierEq *is unconditionally complete, sound and zero-knowledge in the random oracle model.*

Definition 9 (LogEq [78]). *Let G be a multiplicative group of prime order p and H be a hash function, $\mathcal{L}$ be the set of all $(g_1, h_1, g_2, h_2) \in G^4$ where $\log_{g_1}(h_1) = \log_{g_2}(h_2)$. We define the NIZKP* $\mathsf{LogEq} = (\mathsf{Prove}, \mathsf{Verify})$ *for $\mathcal{L}$ by:*

$\mathsf{Prove}((g_1,h_1,g_2,h_2),w)$. *Using the witness $w = \log_{g_1}(h_1)$, this algorithm picks $r \xleftarrow{\$} \mathbb{Z}_p^*$, computes $A = g_1^r$, $B = g_2^r$, $z = H(A,B)$ and $\omega = r + w \cdot z$. It outputs $\pi = (A, B, \omega)$.*

$\mathsf{Verify}((g_1,h_1,g_2,h_2),\pi)$. *Using $\pi = (A,B,\omega)$, this algorithm computes $z = H(A,B)$. If $g_1^{\omega} = A \cdot h_1^z$ and $g_2^{\omega} = B \cdot h_2^z$ then it outputs* 1; *else, it outputs* 0.

LogEq *is unconditionally complete, sound and zero-knowledge in the ROM.*

2.5 Cryptographic Assumptions

Definition 10 (Discret Logarithm assumption). *Let p be a prime number generated according to a security parameter $\eta \in \mathbb{Z}^+$. Let G be a multiplicative group of order p, and $g \in G$ be a generator. The* discrete logarithm (DL) assumption *in (G, p, g) states that there exists a negligible function ϵ such that for all $x \xleftarrow{\$} \mathbb{Z}_p^*$ and $\mathcal{A} \in \textsc{poly}(\lambda)$:*

$$\Pr\left[x' \leftarrow \mathcal{A}(g^x) \colon x = x'\right] \leq \epsilon(\lambda) .$$

Definition 11 (Decisional Composite Residuosity assumption). *Let p and q be two prime numbers generated according to a security parameter $\eta \in \mathbb{N}$. The* decisional composite residuosity assumption *(DCR) states that there exists no adversary $\mathcal{A} \in$ $\textsc{poly}(\eta)$ which can distinguish n-th residues modulo n^2.*

Definition 12 (Approximate GCD (A-GCD) Problem). *Given a set of k integers of the form $x_i = q_i p + r_i$ where $q_i, p, r_i \in \mathbb{Z}$ with q_i, r_i are randomly chosen, the approximate GCD problem is to find p.*

There is no known probabilistic polynomial time algorithm in literature which can solve this problem in polynomial time.

2.6 Feldman's Verifiable Secret Sharing

Verifiable Secret Sharing (VSS) [8] based on the Shamir Secret Sharing [9], where each share is a point (x, y) of a secret polynomial f of degree k. Knowing more than k shares, one can guess the polynomial f and can compute the secret $s = f(0)$. In Feldman's VSS, there is a public value that allows anybody to check the validity of a share. For any point (x, y), anybody can check if y is $f(x)$ or not. This scheme works as follows. Let G be a multiplicative group of prime order p, where DL is hard. Let $f \in \mathbb{Z}_p^*[X]$ be the secret polynomial and $a_i \in F$ be a coefficient for all $0 \leq i \leq k$ such that:

$$f(x) = \sum_{i=0}^{k} a_i \cdot x^i$$

Let $g \in G$ be a generator of G. For all $i \in \{0, \dots, k\}$, we set $h_i = g^{a_i}$. Values g and $\{h_i\}_{0 \leq i \leq k}$ are public, however, the coefficients a_i are hidden under DL hypothesis. We remark that $f(x) = y$ if and only if $g^y = \prod_{i=0}^{k} h_i^{x^i}$:

$$\prod_{i=0}^{k} h_i^{x^i} = \prod_{i=0}^{k} g^{a_i \cdot x_i} = g^{\sum_{i=0}^{k} a_i \cdot x_i} = g^{f(x)} \tag{2.4}$$

Then, we can use equation 2.4 to check that (x, y) is a valid share.

2.7 Conclusion

This chapter presented mathematical notions like probabilistic polynomial time, negligible function, and Lagrange interpolation. We then discussed the Paillier cryptosystem and its homomorphic properties. We note that the Paillier cryptosystem has one unique feature, proof of plaintext equality. We briefly discussed Zero-Knowledge Proof and $\mathsf{DecPaillierEq}$ NIZKP, which proves that k ciphertexts of the same message decrypt to the same plaintext without decrypting each ciphertext. This property is useful in our proposed scheme, $\mathsf{VIP-POPE}$. In the end, we discussed Feldman's Verifiable Secret Sharing (VSS), which is based on Shamir's secret sharing.

CHAPTER 3

Inherent Limitations of Polynomial Evaluation

3.1 Introduction

Broadly, there are three types of entities in cloud computing – Cloud server as service provider, Merchant (Data Owner) as service consumer, and Customer as service consumer. Cloud server facilitates storage and services in which merchant stores the application data and all eligible customers of the merchant get on-demand services from the cloud infrastructure. Data owner hires the cloud infrastructure for storing application data in the cloud storage. While resource outsourcing provides significant advantages to data owners as well as to service consumers, there are some important concerns such as security, privacy, ownership and trust that have been discussed substantially over past decade [49, 74, 73, 75]. For example, the company can delegate the health monitoring systems to the cloud, where a patient can directly communicate with the cloud. However, upon receiving the patient request the cloud can generate a fabricated report for some malicious intent. Therefore, there is a possibility that cloud server can manipulate the data without data owner's knowledge. In order to avoid such scenarios, data owner can prefers to store data in cloud server in a controlled manner so that the cloud server cannot manipulate the data while consumer getting services from it.

Verifiable computation (VC) refers to the cryptographic primitives, where an untrusted computation server can prove the soundness of its computations and it was introduced in [13]. The aim of such a primitive is to allow a weak client to

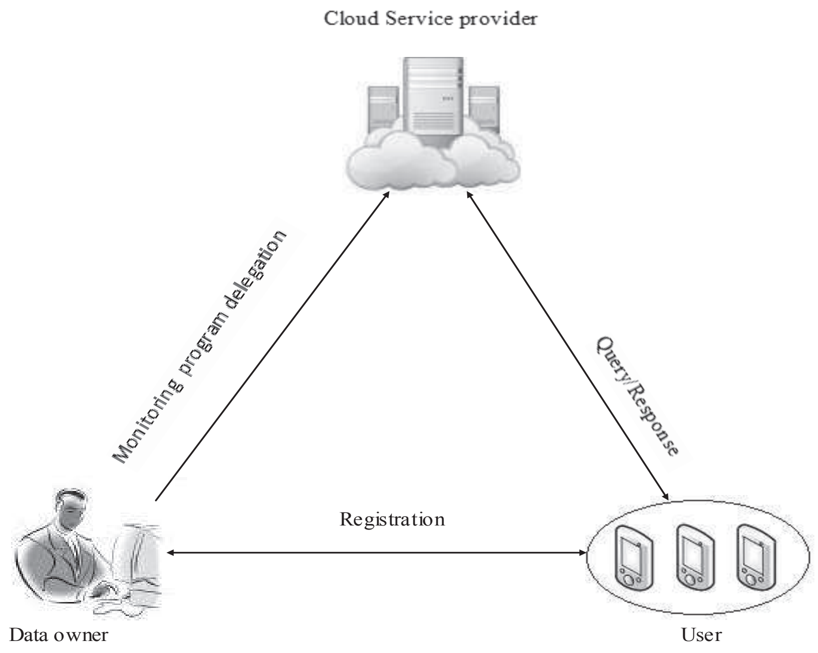


Figure 3.1: A basic Model for mobile healthcare system

delegate difficult computations. Primitives, where anybody can check the soundness of the computation, are said *publicly verifiable* [20]. This subject has led to a very dense literature [18, 5, 10, 6, 19]. In 2012, authors of [5] proposed a formal security model for VC. In [10], the authors propose a scheme for generic polynomials evaluations and generic matrix computations. Unlike our scheme, all these works consider that the algorithm (or the polynomial) used by the server is public and available to the verifier.

A similar scenario was studied by Guo *et al.* [49], where a server receives medical data collected by sensors worn the users and provides the users with an evaluation of their health status. More precisely, the company defines a polynomial f which returns meaningful information, such as potential diseases. Then, it uploads this polynomial to the server and sells to the end users the ability to query that function with their medical data.

We found that the Guo *et al.*'s scheme [49] suffers from major security weaknesses, in particular, the scheme does not provide privacy-preserving services, which is the main claim of the scheme. We provide a mitigation for the weaknesses by modifying the scheme.

3.2 PHR Computation and Verification

Guo *et al.* [49] proposed a scheme, appeared in INFOCOM 2015, that claims verifiable privacy-preserving service in healthcare systems. The scheme has two main objectives - (i) privacy-preserving identity verification, and (ii) verifiable PHR computation. The former provides secure identity verification on cloud without revealing identity of user while later guarantees the correctness of generated PHR. The scheme consists of four entities as follows.

- Trust Authority(TA): TA performs issuance and distributing secret and public parameters to other entities of the scheme.
- Cloud Service Provider(CSP): CSP verifies user identity and computes health record computation using the monitoring program $f(\vec{x})$ provided by the company.
- Company: Company provides health record computation to users with the help of CSP.
- Users: Users are the consumers for their health services/records.

The scheme works as follows. A user receives a private certificate σ from TA. After receiving σ user asks for a blind signature ψ on σ from the company. After that the user is a registered entity for the monitoring program $f(\vec{x})$ and the blind signature ψ is issued for the user. Here, $f(\vec{x})$ is a confidential polynomial function and $\vec{x}$ is the user's data generated by the user as $\vec{x} = (x_1, x_2, x_3, \cdots, x_N)$, $x_i \in Z_n^*$. To access the health records the user encrypts the vector and then sends an encrypted vector with ψ to the CSP. User computes $\vec{c}$ = E($\vec{m}$), where $\vec{m}$ is monitored raw data and E($\cdot$) is a secure encryption scheme. User then generates proofs on σ which are used for authentication. If public verification of given ψ is done by the CSP then it computes $f(\vec{x})$ on given $\vec{c}$. After that the CSP computes the monitoring function and gives results $f(E(\vec{m}))$ and signature δ to the user. User now decrypts using his secret key and checks for correctness of $f(E(\vec{m}))$ and δ based on monitored data $\vec{m}$. The detailed construction of the scheme works with the following phases.

3.2.1 System Setup

TA sets up the system by choosing the security parameters and the corresponding public parameters.

1. General Setup: TA chooses a security parameter ξ and generates public parameters $\texttt{param} = (n, G, G_1, e)$, where $n = pq$ is the order of group G, p and q are large primes, and e is a bilinear pairing mapping.

2. Partially Blind Signature Setup: TA issues domain public parameter $(g, g^s) \in G^2$, where s is a master secret key. TA selects two hash functions $H : \{0,1\}^* \to G$ and $H_0 : \{0,1\}^* \to Z_n^*$. A signing key pair ($pk$, sk), where $pk = H(id_c) \in G$ and $sk = H(id_c)^s$ is generated by TA for the company.

3. Monitoring System Setup: TA chooses $g_0 \in G$ and publishes h, where $h = g_0^p \in_R G_q$. TA issues σ after providing ID id_A for user, where $\sigma = g^{\frac{1}{s+id_A}}$. TA gives the private key $sk = q$ to the user.

3.2.2 Privacy-preserving Identity Verification

This phase is composed of the following four sub-protocols.

1. **Signature Request**.
   $(\theta, \phi) \leftarrow$ Request(g, pk, id_A, w): User asks for some parameters to company for partially blind signature ψ on σ. Before the request is sent, user and company agree on string $l \in \{0,1\}^n$. Then, the company selects $t \in_R Z_n^*$, calculates $\theta = g^t$, $\phi = H(id_c)^t$ and sends (θ, ϕ) to the user.

2. **Partially Blind Signature Generation Process**.
   $\epsilon' \leftarrow$ BlindSign$(\theta, g^s, \phi, l, \sigma)$: User randomly chooses $\alpha, \beta, \gamma \in_R Z_n^\star$ and calculates $\theta' = \theta^\alpha \cdot \left(g^s\right)^\gamma = g^{\alpha t + \gamma s}$, $\phi' = H(id_c)^{\alpha(\beta+t)} H(l)^{-\gamma}$ and $u = \alpha^{-1} H_0(\sigma \parallel \phi') + \beta$, and sends these to the company. Then, the company calculates

$$\epsilon = H(id_c)^{s(t+u)} H(l)^t$$

and sends it back to the user, who unblinds ϵ by calculating $\epsilon' = \epsilon^{\alpha}$.

3. **Commitment and Proof Generation Process**.
   $(com_i, \pi) \leftarrow$ ProveGen$(\theta', \phi', \epsilon', \sigma, l)$. CSP verifies user's identity by using the blind signature $\psi = (\theta', \phi', \epsilon', \sigma, w)$ as shown in equation 3.1.

$$e(e', g)e(X, \sigma)e(Y, g^{-s})e(H(l)^{-1}, \theta') \stackrel{?}{=} e(g, g) \tag{3.1}$$

   where $X = g^{id_A} g^s$ and $Y = \phi' \cdot H(id_c)^{H_0(\sigma \| \phi')}$.
   Note that the verification of the above equation requires the identity id_A of the user along with the blind signature ψ. Therefore, if the user directly sends the blind signature ψ to the CSP, then it reveals the correlation of id_A and the partially blind signature ϵ'.
   Now user generates proofs for the signature and the certificate. For generation of commitments, user chooses $\mu_i, \nu_i \in_R Z_n, i = 1, 2, 3, 4$.

   $\mathtt{com}_1 = \epsilon' h^{\mu_1} = H(id_c)^{\alpha s(t+u)} H(l)^{\alpha t} h^{\mu_1}, \mathtt{com}'_1 = g h^{\nu_1}$
   $\mathtt{com}_2 = g^{id_A + s} h^{\mu_2}, \mathtt{com}'_2 = \sigma h^{\nu_2} = g^{\frac{1}{s + id_A}} h^{\nu_2}$
   $\mathtt{com}_3 = \phi' \cdot H(id_c)^{H_0(\sigma \| \phi')} h^{\mu_3}, \mathtt{com}'_3 = g^{-s} h^{\nu_3}$
   $\mathtt{com}_4 = H(l)^{-1} h^{\mu_4}, \mathtt{com}'_4 = \theta' h^{\nu_4} = g^{\alpha t + \gamma s} h^{\nu_4}$
   After calculating commitment set, user builds the proof

$$\pi = \Pi_1^4 (\mathtt{com}_i h^{-\mu_i})^{\nu_i} (\mathtt{com}'_i)^{\mu_i}$$

   and then sends $(\{\mathtt{com}_i, \mathtt{com}'_i\}_{i=1}^4, \pi)$ to the CSP for verification.

4. **Identity Verification Process**.
   (0,1) $\leftarrow$ Verify$(\{\mathtt{com}_i, \mathtt{com}'_i\}_{i=1}^4, \pi, h, e(g, g))$. CSP checks the equation 3.2 and returns 1 for successful verification, 0 for unsuccessful verification.

$$\prod_{i=1}^{4} e(\mathtt{com}_i, \mathtt{com}'_i) = e(g, g)e(\pi, h) \tag{3.2}$$

3.2.3 Verifiable PHR Computation

After identity verification, user uploads PHR by the following steps.

1. Monitoring Program Delegation: The company delegates the monitoring program to the cloud and then user's PHR is computed by the cloud. The company sends the coefficient vector $\vec{a} = (a_0, a_1, \cdots, a_k)$ and string l to the cloud, where l is used for identifying correlation program.

2. PHR Encryption: Let PHR m be an entry from data vector $\vec{m} = (m_1, m_2, \cdots, m_N), m_i \in Z_n$. User chooses a set of random numbers $\vec{r} = (r_0, r_1, \cdots, r_k), r_i \in Z_n$. Then, the user sends $\vec{r}$ to the company. After getting $\vec{r}$, the company calculates $\vec{r'} = \vec{r} \cdot \vec{a} = (a_0 r_0, a_1 r_1, \cdots, a_k r_k)$. Then, company sends $h^{\bar{r}} = h^{\sum_{i=0}^{k} r'_i}$ and $g^{\bar{r}}$ to the user, and $\bar{r}$ to the CSP, where $\bar{r} = \sum_{i=0}^{k} a_i r_i$. User picks $d \in_R Z_n$ and generates the ciphertext of PHR as

$$c = \left(g h^{d \cdot r_0}, g^{m} h^{d \cdot r_1}, g^{m^2} h^{d \cdot r_2}, \cdots, g^{m^k} h^{d \cdot r_k} \right)$$

   where each entry is computed as $c_i = g^{m^i} \cdot (h^{r_i})^d$. Now, user sends $\{c, \lambda, H(l)\}$ to the CSP, where $\lambda = \frac{1}{(x-m)\cdot d} \bmod n$. User also requests the company to compute a public parameter $g^{f(x)}$, which later the company sends to the CSP.

3. Verifiable PHR Computation: PHR is computed as follows.

$$\begin{aligned} v &= \prod_{i=0}^{k} \left(g^{m^i} \cdot (h^{r_i})^d \right)^{-a_i} \\ &= \prod_{i=0}^{k} g^{-a_i \cdot m^i} \cdot h^{-a_i r_i d} \\ &= g^{\sum_{i=0}^{k} -a_i \cdot m^i} \cdot h^{\sum_{i=0}^{k} -a_i r_i d} \\ &= g^{-f(m)} \cdot h^{-d \sum_{i=0}^{k} r'_i} \end{aligned}$$

   CSP computes $\lambda' = \frac{\lambda}{\bar{r}} = \frac{1}{(x-m)\cdot d \cdot \bar{r}}$ and signature δ using $g^{f(x)}$ as,

$$\begin{aligned}
delta &= \left(g^{f(x)} \cdot v\right)^{\lambda'} \\
&= \left(g^{f(x)-f(m)} \cdot h^{-d\sum_{i=0}^{k} r_i'}\right)^{\frac{1}{(x-m)\cdot d\cdot \bar{r}}} \\
&= g^{\frac{f(x)-f(m)}{(x-m)}\cdot\frac{1}{d\bar{r}}} \cdot h^{-\frac{1}{(x-m)}} \\
&= \left(g^{w(x)} \cdot h^{-\frac{d\bar{r}}{(x-m)}}\right)^{\frac{1}{d\bar{r}}}
\end{aligned}$$

where $w(x)$ is a $(k-1)$-degree polynomial function. If $f(m)$ is the value based on m, then only it satisfies this condition $w(x) \equiv \frac{f(x)-f(m)}{(x-m)}$. Then, CSP sends $\{v, \delta\}$ to the user.

4. PHR Result Decryption and Verification: Using the private key $sk = q$ the user decrypts v as in equation 3.3

$$\left(\frac{1}{v}\right)^q = (g^{f(m)}h^{d\bar{r}})^q = (g^q)^{f(m)}h^{d\bar{r}q} = (g^q)^{f(m)} \in G_p \tag{3.3}$$

User can recover $f(m)$ by computing the discrete log of $\left(\frac{1}{v}\right)^q$ with base g^q. Here, $f(m)$ is bounded by M where M is very small compared to p,q and therefore, it is feasible to compute the discrete log of $\left(\frac{1}{v}\right)^q$.
For getting the proof on $f(m)$, the user sends encrypted $(x, f(m))$ to the company. Then, the company constructs coefficient vector $w(x)$ as $\vec{w} = (w_0, w_1, \cdots, w_{k-1})$ and proves $W = g^Z$, where $Z = \sum_{i=0}^{k-1} w_i x^i$ and responds to the user. Now, the user calculates $(g^{\bar{r}})^d = g^{d\bar{r}}$ and $\eta = (h^{\bar{r}})^{-d/(x-m)}$. Finally, the user verifies the equation 3.4 to see whether the CSP has computed correct results or not.

$$e(W \cdot \eta, g) \overset{?}{=} e(\theta, g^{d\bar{r}}) \tag{3.4}$$

3.3 Security Weaknesses

We show two security flaws in Guo *et al*'s scheme [49]. The company's goal is the confidentiality of the monitoring program $f(x)$. If a malicious user obtain $f(x)$

then he can use it for free and he can even sell it to someone else. We note that the company delegates the monitoring program $f(x)$ to the CSP, with the assumption that the computation of $f(x)$ on patients' PHR can be done by the CSP without loosing the confidentiality of the monitoring program $f(x)$. In other words, the monitoring program $f(x)$ should not be known to any other party except the Company and the CSP. Furthermore, anyone can pass the identity verification process without even communicating with the TA or company and therefore, if a malicious user leaks $H(l)$ to a non-user (attacker), then the attacker can use the system with all credentials.

3.3.1 Insider Attack

The monitoring program is a polynomial of degree k and hence, it can be represented as a $k+1$ length vector, $\vec{a} = (a_0, a_1, a_2, \ldots, a_k)$, where a_i is the coefficient of x^i in the polynomial.

$$f(x) = \sum_{i=0}^{k} a_i x^i = a_0 + a_1 x + a_2 x^2 + \cdots + a_k x^k.$$

The company wants to keep this vector $\vec{a}$ secret from everyone except the cloud. Therefore, there are total $k+1$ unknowns and it is easy to find values for these unknowns if we have $k+1$ linearly independent equations involving the coefficients $\{a_i\}_{i=0}^{k}$. An authenticated user(insider) can use the service for $k+1$ times and get PHR report $f(m_i)$, where m_i is the PHR sent by the user on i^{th} time use of the service. Using the set $\{(m_i, f(m_i))\}_{i=0}^{k}$, the user can create the system of equations in $k+1$ variable and solve it for the vector $\vec{a}$. More concretely, assume that the user has the set $\{(m_i, f(m_i))\}_{i=0}^{k}$. Then for each $i \in \{0, 1, 2, \ldots, k\}$, we have

$$a_0 + a_1 m_i + a_2 m_i^2 + \cdots + a_k m_i^k = f(m_i)$$

Without loss of generality, we assume that these $(k+1)$ equations are linearly independent (if not, then the user can always use the service until it is true). We can represent this system of equation in terms of matrices as follows.

$$A = \begin{bmatrix} 1 & m_1 & \dots & m_1^k \\ 1 & m_2 & \dots & m_2^k \\ \vdots & \vdots & \ddots & \vdots \\ 1 & m_{k+1} & \dots & m_{k+1}^k \end{bmatrix} \quad X = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{bmatrix} \quad B = \begin{bmatrix} f(m_1) \\ f(m_2) \\ \vdots \\ f(m_k) \end{bmatrix}$$

$$AX = B$$

Solution of the above system of equations is given by

$$X = A^{-1}B.$$

Now, the user can easily solve the above system of equation for the vector $X = (a_0, a_1, \dots, a_k)$ and the user can use it to compute $f(m) = \sum_{i=0}^{k} a_i m^i$ for any PHR m. In the scheme[49], it is assumed that the degree of the polynomial is around 10 and that makes this attack more easy. Although this attack does not violate privacy of other users, it reveals the confidential monitoring program $f(x)$ of all users pertaining to the company who owns the monitoring program. In this attack, the user obtains $f(x)$ and thereby, computes the result of $f(x)$ without contacting the CSP or the Company, which reveals the confidentiality of the monitoring program $f(x)$.

3.3.2 Outsider Attack

We note that the cloud does not use any extra information other than the commitments sent by the user and the public parameters published by TA. This makes the process vulnerable to unauthenticated identity verification. The attacker can choose commitments as follows.

$\texttt{com}_1 = g, \ \texttt{com}'_1 = g$
$\texttt{com}_2 = g, \ \texttt{com}'_2 = g^{-2}$
$\texttt{com}_3 = g, \ \texttt{com}'_3 = g^{2}$
$\texttt{com}_4 = \pi, \ \texttt{com}'_4 = h$, where $\pi \in_R G$

Since g and h are public parameters, the attacker does not have any trouble in choosing these commitments and π can be any random element of the group G. The attacker sends π and $(\{\mathtt{com}_i, \mathtt{com}'_i\}_{i=1}^4)$ to the cloud for verification. Upon receiving the commitments from the user, the cloud verifies the equality of the equation 3.5

$$\prod_{i=1}^{4} e(\mathtt{com}_i, \mathtt{com}'_i) = e(g,g)e(\pi,h) \tag{3.5}$$

Proof. We prove the equality of the equation 3.5.

$$\begin{aligned}\prod_{i=1}^{4} e(\mathtt{com}_i, \mathtt{com}'_i) &= e(g,g)e(g,g^{-2})e(g,g^2)e(\pi,h) \\ &= e(g,g)e(g,g)^{-2}e(g,g)^2e(\pi,h) \\ &= e(g,g)^{1-2+2}e(\pi,h) \\ &= e(g,g)e(\pi,h)\end{aligned}$$

Therefore, anyone can pass through the identity verification process. Once the verification is successful, the cloud allows the attacker to use the service. Here, we assume that the attacker already has $H(l)$ and k. The attacker follows the rest of the process same as an authenticated user described in the previous section and gets (v, δ) in response from the cloud, where

$$v = g^{-f(m)}h^{-d\bar{r}}$$

The v contains information about $f(m)$ and the attacker's aim is to get the the PHR report $f(m)$ for the PHR m. Note that the attacker is not an authenticated user and he does not have the secret key q and hence, can not decrypt v. However, the attacker can find $f(m)$ using brute force because size of $f(m)$ is small. Since the attacker follows the rest of the process after identity verification, the attacker

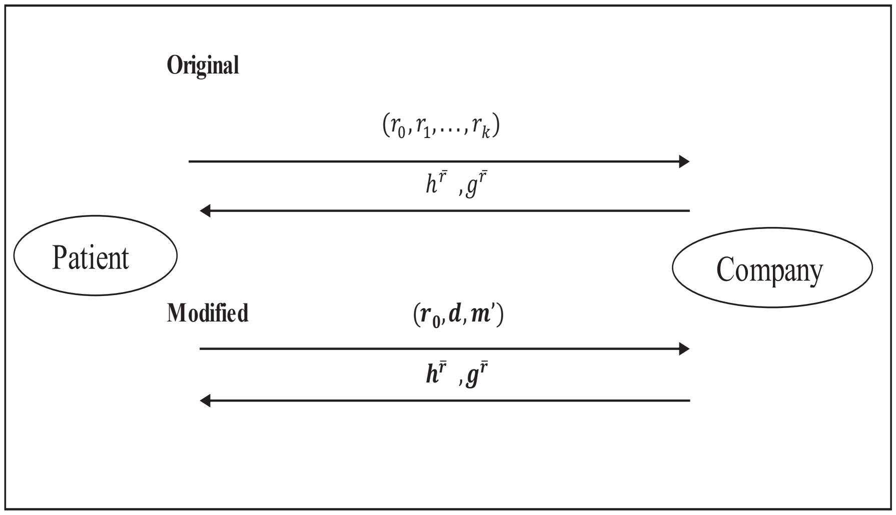

Original
$(r_0, r_1, \ldots, r_k)$
$h^{\bar{r}}, g^{\bar{r}}$
Patient
Company
Modified
(r_0, d, m')
$h^{\bar{r}}, g^{\bar{r}}$

Figure 3.2: Prevention of Insider attack: Changes between Original and Modified schemes

will have d and $h^{\bar{r}}$. The attacker computes

$$v' = vh^{d\bar{r}} = g^{-f(m)}.$$

In [49], the authors have considered that values of m and $f(m)$ are not more than 1000. Therefore, the attacker can simply check whether v' is equal to g^{-j} for every $j \in \{0, 1, 2, \ldots, 1000\}$. By using only 1000 iterations, the attacker can successfully get $f(m)$.

3.4 Proposed Improvements

3.4.1 Prevention of Insider Attack

The insider attack is possible because the attacker knows the degree k of the polynomial. We provide a way to keep the polynomial $f(x)$ secure by keeping the degree of the polynomial secret.

Let m be the PHR value and user wants to get a report $f(m)$ for it. User chooses two random numbers $r_0, d \in Z_n$ and a random prime $p_1 \in \mathbb{Z}_n$. User computes $m' = m + p_1$ and sends (r_0, d, m') to the company. The Figure 2 reflects the changes suggested for preventing the observed insider attack in Guo *et al*'s scheme.

After receiving (r_0, d, m'), the company generates k random integers $r_1, r_2, \ldots, r_k \in Z_n$ using r_0. Company calculates

$$\vec{r'} = \vec{r}.\vec{a} = (a_0 r_0, a_1 r_1, \ldots, a_k r_k)$$

and

$$c = (gh^{dr_0}, g^{m'} h^{dr_1}, g^{m'^2} h^{dr_2}, \ldots, g^{m'^k} h^{dr_k}).$$

Company sends $(h^{\bar{r}}, g^{\bar{r}})$ to the user and $(\bar{r}, c)$ to the cloud, where $\bar{r} = \sum_{i=0}^{k} a_i r_i$. User selects a random point $x \in Z_n$ and computes $\lambda = \frac{1}{x - m'} d$. User sends x to the company and $(\lambda, H(l))$ to the cloud. Company computes $g^{f(x)}$ and sends it to the cloud. Upon receiving $(\lambda, H(l))$ from user and $(\bar{r}, c, g^{f(x)})$ from the company, the cloud computes v and δ. Everything remains same except that c is encryption of m' instead of m. After decrypting v, user gets $f(m') = f(m + p_1)$. For sufficiently large value of p_1, we have $f(m + p_1) \bmod p_1 = f(m)$. The verification process remains same. Since the user does not know the degree k, the user can not retrieve coefficients of the polynomial $f(x)$.

3.4.2 Prevention of Outsider Attack

We modify the scheme in such a way that only registered user can use the service to get PHR report $f(m)$ for a given PHR m. Note that the cloud computes $f(m)$ only after successful identity verification process. After generation of the blind signature, the company and the cloud agree on some random number $z \in_R Z_{p^*}$ and a timestamp t_m. Then, the company computes $g_1 = g^{H(t_m \| z)}$ and sends g_1 with ϵ to the user. The Figures 3 and 4 reflect the changes suggested for preventing the observed outsider attack in Guo *et al*'s scheme.

After receiving $\{g_1, \epsilon\}$ the user computes commitments. Except $\texttt{com}_2$ all other commitments remain same. We modify $\texttt{com}_2$ as follows:

$$\texttt{com}_2 = g_1^{id_A + s} h^{\mu_2}$$

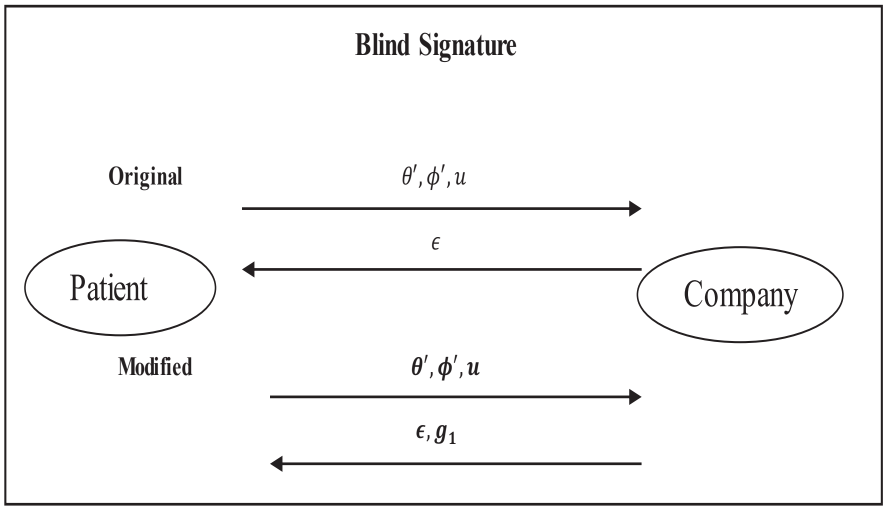


Figure 3.3: Prevention of Outsider attack: Changes shown in Blind signature

Now, based on this modification, user computes the proof

$$\pi = \prod_{i=1}^{4} (\mathtt{com}_i h^{-\mu_i})^{\nu_i} (\mathtt{com}'_i)^{\mu_i}$$

and sends $(\{\mathtt{com}_i, \mathtt{com}'_i\}_{i=1}^{4}, \pi)$ to the cloud for verification. During the identity verification process, the cloud verifies the equality of the equation 3.6 and returns 1 for successful verification and 0 for unsuccessful verification.

$$\prod_{i=1}^{4} e(\mathtt{com}_i, \mathtt{com}'_i) = e(g_1, g)e(\pi, h) \tag{3.6}$$

Correctness:

$$\prod_{i=1}^{4} e(\mathtt{com}_i, \mathtt{com}'_i)$$

$$= e(\epsilon' h^{\mu_1}, g h^{\nu_1}) e(\phi' \cdot H(id_c)^{H_0(\sigma \| \phi')} h^{\mu_3}, g^{-s} h^{\nu_3})$$

$$\cdot e(H(l)^{-1} h^{\mu_4}, g^{\alpha t + \gamma s} h^{\nu_4}) e(g_1^{id_A + s} h^{\mu_2}, g^{\frac{1}{s + id_A}} h^{\nu_2})$$

$$= e(H(id_c)^{\alpha s(t+u)} H(l)^{\alpha t}, g) e(\phi' \cdot H(id_c)^{H_0(\sigma \| \phi')}, g^{-s})$$

$$\cdot e(H(l)^{-1}, g^{\alpha t + \gamma s}) e(h^{\mu_1}, g) e(\epsilon' h^{\mu_1}, h^{\nu_1}) e(h^{\mu_3}, g^{-s})$$

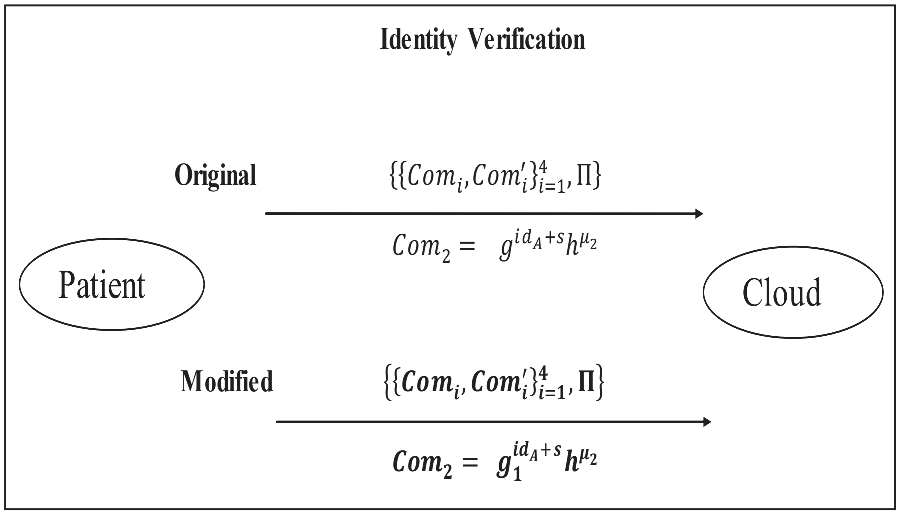

Identity Verification
Original
$\{\{Com_i, Com'_i\}_{i=1}^4, \Pi\}$
$Com_2 = g^{id_A+s}h^{\mu_2}$
Patient
Cloud
Modified
$\{\{Com_i, Com'_i\}_{i=1}^4, \Pi\}$
$Com_2 = g_1^{id_A+s}h^{\mu_2}$

Figure 3.4: Prevention of Outsider attack: Changes shown for Identity Verification

$$\cdot e(\phi' \cdot H(id_c)^{H_0(\sigma\|\phi')}h^{\mu_3}, h^{\nu_3})e(h^{\mu_4}, g^{\alpha t+\gamma s})$$

$$\cdot e(H(l)^{-1}h^{\mu_4}), h^{\nu_4})e(g_1^{id_A+s}h^{\mu_2}, g^{\frac{1}{s+id_A}}h^{\nu_2})$$

$$= e(H(id_c)^{\alpha s(t+u)}, g)e(H(id_c)^{\alpha(\beta+t)+H_0(\sigma\|\phi')}, g^{-s})$$

$$\cdot e(H(l)^{\alpha t}, g)e(H(l), g)^{\gamma s-\alpha t-\gamma s}e(h^{\mu_1}, g)e((\epsilon' h^{\mu_1})^{\nu_1}, h)$$

$$\cdot e(g^{-s\mu_3}, h)e((\phi' \cdot H(id_c)^{H_0(\sigma\|\phi')}h^{\mu_3})^{\nu_3}, h)e(g^{\mu_4(\alpha t+\gamma s)}, h)$$

$$\cdot e(H(l)^{-1}(h^{\mu_4})^{\nu_4}, h)e(g_1, g)e(g_1^{(id_A+s)\nu_2+\frac{\mu_2}{s+id_A}}, h)e(h^{\mu_2\nu_2}, h)$$

$$= e(g_1, g)\prod_{i=1}^{4} e((com_i h^{-\mu_i})^{\nu_i}(com'_i)^{\mu_i}, h) = e(g_1, g)e(\pi, h)$$

Here, the attacker does not have g_1, so he can not pass the identity verification process. Without passing the verification process, the attacker can not compute $f(m)$ for any PHR m. We note that after the identity verification there is also a need for message authentication (to avoid user impersonation attack) between the company and the user in the PHR computation phase of the scheme.

	Guo *et al*'s scheme [49]	Improved scheme
Computational cost	$((3k+M+40)log(n)+2k+25)G+6(k+1)M+8E$	$((3k+M+40)log(n)+k+25)G+(5k+7)M+8E$
Storage cost	*Public:* $7log(n)$ *bits* *Private:* $(k+6)log(n)$ *bits*	*Public:* $7log(n)$ *bits* *Private:* $(k+7)log(n)$ *bits*
Communication cost	$(3k+33)log(n)$ *bits*	$(k+35)log(n)$ *bits*

Table 3.1: Comparison of Guo *et al*'s scheme and the improved scheme

3.4.3 Performance Analysis

We compare the Guo *et al*'s scheme and the proposed improved scheme with respect to the computational, storage and communication costs requirement in the schemes. In the table 1, k is the degree of the monitoring program $f(x)$, n is a public parameter, and M is the size of the message space. The table 1 provides computational complexity of both schemes in terms of the number of group multiplications (G), the number of integer multiplications (M) and the number of bilinear pairing computations (E). For exponentiation of a group element, we consider the square and multiply algorithm to count the number of group multiplications. The improved scheme is comparable with Guo *et al*'s scheme in terms of computation and storage costs and provides better efficiency in terms of communication cost.

3.5 Inherent Limitation

In the scheme [49], the degree k of the polynomial is public. However, even if k is secret, any user can guess k and f after $k+1$ interactions with the server as follows. The user chooses an input x_0 and sends it to the server. He receives y_0 and computes the polynomial f_0 of degree 0 using Lagrange's interpolation on (x_0, y_0). Next, the attacker chooses a second input x_1 and asks $y_1 = f(x_1)$ to the server. He computes the polynomial f_1 of degree 1 using Lagrange's interpolation on $\{(x_0, y_0), (x_1, y_1)\}$. At each iteration i, he compares f_{i-1} and f_i. If $f_{i-1} = f_i$, he returns $k = i-1$ and $f = f_{i-1}$. Indeed, when $i > k$, the attacker uses Lagrange's interpolation with more than k points of f. Since the degree of f is k, the interpolation on $i > k$ points always returns f. By repeating this process until the interpolation returns the same polynomial $f_i = f_{i+1}$ for two consecutive iterations, he recovers both the degree and the polynomial. There is no way to prevent it, as it is an inherent limitation of PPE. Note that this limitation was already considered in the security model of Kate *et al.* [14]. Thus, to preserve the protection of the polynomial, the server must refuse to evaluate more than k points for each client. It is assume that the clients do not collude to collect more than k points.

3.6 Single Query Attack

In addition to the protection of f, the scheme [49] requires that the user's data is encrypted for the server. More formally, the user uses an encryption algorithm to compute $x' = \mathsf{Enc}_k(x)$ and sends this cipher to the server which returns y'. Then, the user computes $y = \mathsf{Dec}_k(y')$ such that $y = f(x)$ where f is the secret polynomial. The encryption scheme is based on the discrete logarithm assumption. The decryption algorithm works in two steps: First the user computes a value h such that $h = g^{f(x)}$ where g is a generator of a multiplicative group of large prime order n, next he computes the discrete logarithm of h in base g using *Pollard's lambda method* [50]. The authors assume that the size of $f(x)$ is *reasonable*: more formally, they define a set of possible inputs $\mathcal{X}$ and $M \in \mathbb{N}$ such that $\forall x \in \mathcal{X}, 0 \leq f(x) < M$. The authors suppose that users can compute Pollard's lambda algorithm on any $h = g^y$ where $y < M$. Actually, for practical reasons, since $h = g^{f(x) \mod n}$ and $\log_g(h) = f(x)$, we suppose that $0 \leq f(x) < n$ for any input x of *reasonable size* (*i.e.* $x << n$). Hence, we consider f as a positive polynomial in $\mathbb{Z}$ with sufficiently small coefficients.

It is easy to evaluate a *"small"* M' such that $M' > M$. It is suffisent to choose M' such that Pollard's lambda algorithm on $g^{M'}$ is easily computable by a powerfull server but is too slow for a practical application. For example, if Pollard's lambda algorithm takes less than one minute for the server but more than one hour for the user's computer, we can suppose that $M' > M$ and attacks that are polynomial in M' are practical. To sum up, the user has the following tools:

- $M' \in \mathbb{N}$ such that $\forall x \in \mathcal{X}$, $0 \leq f(x) < M'$ and such that algorithms that require $p(M')$ operations (where p is a polynomial) are *easily computable*.

- A server which returns $y = f(x)$ for any input x. This server can be used at most k times where k is the degree of the polynomial.

The authors assume that the size of $f(x)$ is *reasonable*: more formally, they define a set of possible inputs $\mathcal{X}$ and $M \in \mathbb{N}$ such that $\forall x \in \mathcal{X}, 0 \leq f(x) < M$. The authors also assume that $f(x) \geq 0$ for any x and that $\mathcal{X} \subset \mathbb{N}$. We show that any user can guess the secret polynomial during his first interaction with the server. We first prove the following two properties.

Property 2. *For any polynomial $f \in \mathbb{Z}[X]$ and any integers x and y, there exists $P \in \mathbb{Z}$ such that*

$$f(x+y) = f(x) + y \cdot P \; .$$

Proof. We show by induction that for any integers x, y and k there exists $P_k \in \mathbb{Z}$ such that $(x+y)^k = x^k + y \cdot P_k$. For $k = 0$, we set $P_0 = 0$ because $(x+y)^0 = 1 = x^0 = x^0 + y \cdot P_0$ when $P_0 = 0$. We assume that the property is true for k, we show that for any integers x and y there exists:

$$P_{k+1} = P_k \cdot x + x^k + P_k \cdot y$$

such that $P_{k+1} \in \mathbb{Z}$ and:

$$(x+y)^{k+1} = x^{k+1} + y \cdot P_{k+1}$$

We verify that:

$$\begin{aligned}
(x+y)^{k+1} &= (x+y) \cdot (x+y)^k \\
&= (x+y) \cdot (x^k + y \cdot P_k) \\
&= x^{k+1} + x \cdot y \cdot P_k + y \cdot x^k + y \cdot P_k \cdot y \\
&= x^{k+1} + y \cdot (P_k \cdot x + x^k + P_k \cdot y) \\
&= x^{k+1} + y \cdot P_{k+1}
\end{aligned}$$

Then for any polynomial $f(x) = \sum_{i=0}^{k} a_i \cdot x^i$ of degree k, we can set $P = \sum_{i=0}^{k} a_i \cdot P_i$ such that:

$$
\begin{aligned}
f(x+y) &= \sum_{i=0}^{k} a_i \cdot (x+y)^i \\
&= \sum_{i=0}^{k} a_i \cdot (x^i + y \cdot P_i) \\
&= \sum_{i=0}^{k} a_i \cdot x^i + \sum_{i=0}^{k} a_i \cdot y \cdot P_i \\
&= f(x) + y \cdot \sum_{i=0}^{k} a_i \cdot P_i \\
&= f(x) + y \cdot P
\end{aligned}
$$

□

Note that for any positive integers a and b such that $a < b$, we have $a \mod b = a$. Then, we can deduce the following property from Property 2.

Property 3. *For any polynomial $f \in \mathbb{Z}[X]$ and any integers x and y such that $y > f(x) \geq 0$ and $f(x+y) \geq 0$, it holds that:*

$$f(x+y) \mod y = f(x) .$$

Proof. From the previous property, we have $f(x+y) = f(x) + y \cdot P$, where P is an integer. Assume $P < 0$, we define $P' = -P > 0$, then $f(x+y) = f(x) - y \cdot P' \geq 0$. Hence we have $f(x) \geq y \cdot P' > f(x) \cdot P'$.

- If $f(x) > 0$ then we deduce $1 = f(x)/f(x) > P'$ and $1 > P'$.
- Else, if $f(x) = 0$ then $0 \geq y \cdot P' > 0$.

In both cases, we obtain a contradiction. We conclude that $P \geq 0$. Finally, we deduce

$$f(x+y) \mod y = f(x) + y \cdot P \mod y = f(x) .$$

□

Our attack on [49] works as follows. The attacker chooses a vector of k integers $(x_1, x_2, \ldots, x_k) \in \mathbb{N}^k$ such that, for all $0 < i \leq k$:

$$x'_i = \left(\sum_{j=1}^{i} x_j \right) \in \mathcal{X} \ .$$

For the sake of clarity, we show the attack in the case where $\{1, \ldots, k\} \subset \mathcal{X}$. Thus the attacker chooses the vector $(x_1, x_2, \ldots, x_k) = (1, 1, \ldots, 1)$ and sends $x = k + M'$ to the server that returns the encryption of $y = f(x)$. Pollard's lambda algorithm complexity on M' is $O({M'}^{1/2})$. We consider that $k << M'$, $k \approx 10$ as in [49], thus $x < 2 \cdot M'$, the complexity of the decryption with Pollard's lambda algorithm is $O(f(2M')^{1/2}) \approx O({M'}^{k/2})$. For all $1 \leq i \leq k$, the attacker computes:

$$M'_i = k - i + M';$$

$$y_i = y \mod M'_i \ .$$

Since for all $a \in \mathcal{X}$, $M' > f(a)$, we have for all $1 \leq i \leq k$:

$$M'_i = k - i + M' \geq M' > f(a) \ .$$

Using Property 3, since $i \in \mathcal{X}$, we deduce that

$$\begin{aligned} y_i &= f(x) \mod M'_i \\ &= f(k + M') \mod M'_i \\ &= f(k - i + i + M') \mod M'_i \\ &= f\left(i + M'_i\right) \mod M'_i \\ &= f(i) \ . \end{aligned}$$

$$
\begin{aligned}
y'_i = y \mod & \left[\left(\sum_{j=i+1}^{j=k} x_j\right) + M'\right] \\
&= f(x) \mod M'_i \\
&= f\left(\left(\sum_{j=1}^{j=k} x_j\right) + M'\right) \mod M'_i \\
&= f\left(\left(\sum_{j=1}^{j=i} x_j\right) + \left(\sum_{j=i+1}^{j=k} x_j\right) + M'\right) \mod M'_i \\
&= f\left(x'_i + M'_i\right) \mod M'_i = f(x'_i)
\end{aligned}
$$

Hence, the attacker practically obtains $k+1$ points from one single queried point. He then uses Lagrange's interpolation on $((1, y_1), (2, y_2) \dots (k, y_k), (x, y))$ to guess f. Finally, after only one query, an attacker can compute f with reasonable computation time.

It is possible to attack the modified scheme [51] in a similar way. Indeed, as in [49], the user knows a value M such that $\forall x \in \mathcal{X}$, $f(x) < M$. A simple countermeasure could be to not allow the user to evaluate inputs that are not in $\mathcal{X}$. Unfortunately, this is not possible in these two schemes since the user encrypts his data x, then the server does not know if $x \in \mathcal{X}$ or not.

3.7 Conclusion

In this chapter, we presented a recent privacy-preserving verifiable scheme in the context of the healthcare system. The scheme has two main parts: Privacy-preserving identity verification and Verifiable PHR computation. In Privacy-preserving identity verification, the user can authenticate itself with the cloud server without leaking any information related to the identity. We later show that any unregistered user can generate valid parameters for the verification. The user can get unauthorized access to the cloud service. In another attack, we show that the user can obtain the monitoring program, which is secret and only known to the company and the cloud. We provide a quick fix to each of these attacks.

Later, we show that there is a generic inherent limitation to any verifiable polynomial evaluation scheme. For a polynomial of degree k, if a user has at least $(k+1)$ many evaluations of the polynomial, the user can obtain the polynomial using Lagrange interpolation. Therefore, if a user can obtain the polynomial with less than $(k+1)$ many evaluations, then only it can be called an attack. We revisit the PHR computation scheme and show that it is possible to obtain the scheme's polynomial using a single query.

The leakage of the monitoring program in a single query is a significant issue. First, the user will stop using cloud service after obtaining the polynomial. Second, there is not a standard security notion that considers leakage of the monitoring program. In the next chapter, we further discuss the leakage issue for general verifiable polynomial evaluation, and we propose standard security models that capture leakage of the polynomial.

CHAPTER 4

Verifiable Private Polynomial Evaluation

4.1 Introduction

From harmless smart gardening [86] to critical applications such as forest fire detection [87], data monitoring through sensors is becoming pervasive. In particular, sensors for monitoring health related data are more and more widely adopted, be it through smart watches that track the heart rate, or sensors implemented in the patient's body [67]. This medical data can sometimes be used to assess the health status of an individual, by applying a single variable polynomial prediction function on it [68]. However, when it comes to medical data, extreme care must be taken in order to avoid any leakage. Recently, the leak of medical data of 1.5 million SingHealth users in Singapore strongly incentivised to improve the security and privacy surrounding medical data [69]. In this context, we consider the following problem:

How can a company use medical data recorded by clients to give them predictions about their health status in a privacy way?

For instance, this company may collect fitbit data from its customers, and use it to predict things such as a risk factor for certain diseases.

For economic reasons, this company keeps the polynomial secret: it invested time to build it, and required to collect lots of data. Its economic model is based on the secrecy of the polynomial: the clients pay the company to obtain the polynomial's output on their medical data. If the polynomial was public, then the clients would directly compute it, and the company would cease to exist. However, as the company grows, it becomes difficult to treat all the computation requests, so that the company needs to delegate this computation to a cloud service. The company trusts the cloud service provider and gives the secret polynomial; however, the clients may not trust the server to produce correct results, so that the company would like the server to be able to prove the correctness of each prediction to the client, i.e, prove that its output is correct with regards to the secret prediction function.

In this scenario, the problem is how to delegate computations on a secret polynomial function to an external server in a verifiable way. In a Private Polynomial Evaluation (PPE) scheme, the company outsources the secret polynomial function $f(\cdot)$ to an external server. Moreover, the company provides some public information vk called *verification key*. This verification key is used with the proof π generated by the server during the delegated computation of $f(x)$ to allow clients to verify the correctness of the result returned by the server.

The underlying problem is how to delegate the computation on a secret polynomial function to a server, in a verifiable way. By *secret* we mean that no user should be able to retrieve the polynomial used by the server, and by *verifiable* we mean that the delegate must be able to prove his computation is correct. To solve this challenging problem, we propose a new primitive PPE, for private polynomial evaluation, which ensures that:

1. The polynomial f is protected from users as much as possible;
2. The user can verify the result given by the server.

We revisit formal security models for PPE schemes for two main reasons:

- In [14] the authors propose some models, where the secret polynomial is randomly chosen. However, they present several applications, where the polynomial is not random. Their models are clearly not sufficient for analyzing the security of this kind of applications.
- The schemes presented in [49] and [51] consider polynomials that are not randomly chosen. For example, it is possible to use polynomials that have small coefficients without loss of security. The authors give neither security

models nor security proofs. The next section presents a practical attack on these two schemes, where a user exploits some public information. To avoid such attacks, a model is required, where public information does not give significant advantage.

Our goal is to design a model, where the public parameters and the server proofs of soundness give no advantage to an attacker. Ideally, we would like the attacker has no more chance of success than if he only had access to a server reliably returning polynomial evaluations with no proof of soundness. we would like an attacker that has no more chance to succeed his attack than if the server is *trusted and does not give any proof together with the evaluated point* Our security model considers an attacker that tries to determine which polynomial is used by a PPE among two polynomials of his choice. This model is inspired by the IND-CPA model used in public key cryptography.

4.2 System Model for PPE

We first give brief introduction of our proposed health-care system. The system mainly consists of three entities: cloud service provider (CSP), the company and users.

- Cloud Service Provider (CSP): CSP verifies evaluates a function $f(x)$ over a user input m. The CSP also computes a proof of computation for $f(m)$.
- Users: Users are consumers of the company.
- Company: The company provide appropriate service in terms of evaluation of a function $f(x)$ with the help of CSP. The company also generates and securely distributes public and private parameters to all involved entities.

Figure 4.1 illustrates the process of a PPE scheme, where x is the user data and $f(x)$ is the evaluation of the data x by the function f of the company. Moreover, the proof send by the server and the key verification vk send by the company allow the user to verify the correctness of the delegated computation. The PPE scheme can solve the computational verification objective of the proposed healthcare system.

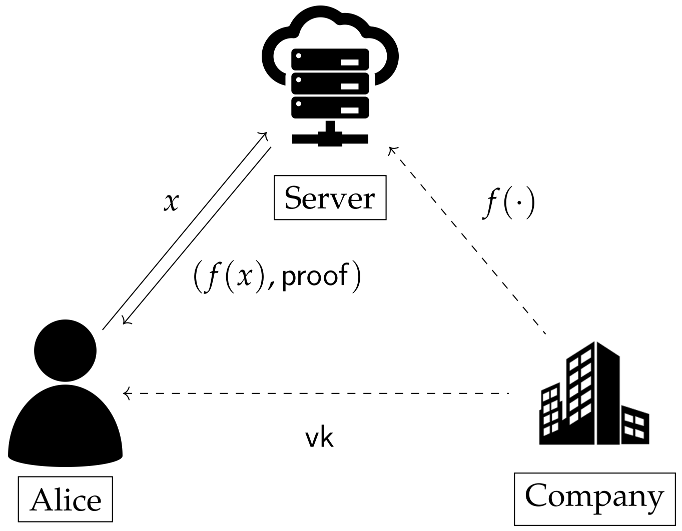


Figure 4.1: System model of a PPE scheme.

Assumptions

In the scheme [49], the degree k of the polynomial is public. However, even if k is secret, any user can guess k and f after $k+1$ interactions with the server as follows. The user chooses an input x_0 and sends it to the server. He receives y_0 and computes the polynomial f_0 of degree 0 using Lagrange's interpolation on (x_0, y_0). Next, the attacker chooses a second input x_1 and asks $y_1 = f(x_1)$ to the server. He computes the polynomial f_1 of degree 1 using Lagrange's interpolation on $\{(x_0, y_0), (x_1, y_1)\}$. At each iteration i, he compares f_{i-1} and f_i. If $f_{i-1} = f_i$, he returns $k = i - 1$ and $f = f_{i-1}$. Indeed, when $i > k$, the attacker uses Lagrange's interpolation with more than k points of f. Since the degree of f is k, the interpolation on $i > k$ points always returns f. By repeating this process until the interpolation returns the same polynomial $f_i = f_{i+1}$ for two consecutive iterations, he recovers both the degree and the polynomial. There is no way to prevent it, as it is an inherent limitation of PPE. Note that this limitation was already considered in the security model of Kate *et al.* [14]. Thus, to preserve the protection of the polynomial, the server must refuse to evaluate more than k points for each client. We also assume that the clients do not collude to collect more than k points.

Consider the following scenario, where PPE is used for prediction function. An attacker corrupts some technicians of the company who collect and store the data. The attacker wants to guess what is the secret mathematical model used by the company. The attacker uses some different models to build some possible prediction functions from the collected data. Its goal is to distinguish which function is used from the public values given by the company. To prevent such an attack, we need a security model, where it is hard to distinguish what is the used function among several possible functions. Then we design a PPE scheme that has this security property.

In order to be able to define a strong security model, we first need to formally define a Private Polynomial Evaluation scheme.

Definition 13. *Let f be a polynomial, a* Private Polynomial Evaluation *scheme (PPE) is a tuple of algorithms* $(\mathsf{setup}, \mathsf{init}, \mathsf{compute}, \mathsf{verif})$ *such that:*

$\mathsf{setup}(\lambda)$**:** *Returns a ring F and a public setup* pub.

$\mathsf{init}(\mathsf{pub}, f)$**:** *Returns a server key* sk *and a verification key* vk.

$\mathsf{compute}(\mathsf{pub}, \mathsf{vk}, x, \mathsf{sk}, f)$**:** *Returns y and a proof π that $y = f(x)$.*

$\mathsf{verif}(\mathsf{pub}, \mathsf{vk}, x, y, \pi)$**:** *Returns* 1 *if the proof π is "accepted"; otherwise* 0.

We start by redefining the "weak" security notions that have been presented in the literature. Then we introduce the notion of chosen function attack and the natural notion of unforgeability.

4.2.1 Polynomial Protection

We introduce the polynomial protection (PP) security: a PPE is PP secure when there exists no adversary that guess a new point (not computed by the server) of the secret polynomial f. In this model, the polynomial is randomly chosen, and the adversary cannot use the server more than k times, where k is the degree of f. This security model is similar to the *"Hiding"* model given in [14] except that the adversary chooses the evaluated points by himself. We define the weak

```
$\mathsf{Exp}^{k\text{-}\mathsf{PP}}_{\Pi,\mathcal{A}}(\lambda)$:
  $(\mathsf{pub}, F) \leftarrow \mathsf{setup}(\lambda)$;
  $f \xleftarrow{\$} F[X]_k$;
  $\Sigma \leftarrow \emptyset$;
  $c \leftarrow 0$;
  $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{init}(\mathsf{pub}, f)$;
  $(x_*, y_*) \leftarrow \mathcal{A}^{\mathsf{CO}_{\mathsf{PP}}(\cdot)}(\mathsf{pub}, \mathsf{vk}, F, k)$;
  If $(x_*, y_*) \notin \Sigma$ and $f(x_*) = y_*$:
    Then return 1;
  Else return 0;

$\mathsf{Exp}^{k\text{-}\mathsf{IND}\text{-}\mathsf{CFA}}_{\Pi,\mathcal{A}}(\lambda)$:
$b \xleftarrow{\$} \{0,1\}^*$;
  $(\mathsf{pub}, F) \leftarrow \mathsf{setup}(\lambda)$;
  $(f_0, f_1, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pub}, F, k)$;
  $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{init}(\mathsf{pub}, f_b)$;
  $b_* \leftarrow \mathcal{A}_2^{\mathsf{CO}_{\mathsf{CFA}}(\cdot)}(\mathsf{pub}, \mathsf{vk}, F, k, \mathsf{st})$;
  If $f_0 \notin F[X]_k$ or $f_1 \notin F[X]_k$:
    Then return 0;
  Else return $(b = b_*)$;

$\mathsf{CO}_{\mathsf{PP}}(x)$:
  $(y, \pi) \leftarrow \mathsf{compute}(\mathsf{pub}, \mathsf{vk}, x, \mathsf{sk}, f)$;
  $c \leftarrow c + 1$;
  $\Sigma \leftarrow \Sigma \cup \{(x, y)\}$;
  If $c = k + 1$:
    Then return $\perp$;
  Else return $(y, \pi)$;

$\mathsf{Exp}^{\mathsf{UNF}}_{\Pi,\mathcal{A}}(\lambda)$:
  $(\mathsf{pub}, F) \leftarrow \mathsf{setup}(\lambda)$;
  $(f, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pub}, F)$;
  $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{init}(\mathsf{pub}, f)$;
  $(x_*, y_*, \pi_*) \leftarrow \mathcal{A}_2(\mathsf{pub}, \mathsf{sk}, \mathsf{vk}, F, f, \mathsf{st})$;
  If $f(x_*) \neq y_*$ and $\mathsf{verif}(\mathsf{pub}, \mathsf{vk}, x_*, y_*, \pi_*)$:
    Then return 1;
  Else return 0;

$\mathsf{CO}_{\mathsf{CFA}}(x)$:
  $(y, \pi) \leftarrow \mathsf{compute}(\mathsf{pub}, \mathsf{vk}, x, \mathsf{sk}, f_b)$;
  If $f_0(x) \neq f_1(x)$:
    Then return $\perp$;
  Else return $(y, \pi)$;
```

Figure 4.2: Security experiments and oracles definitions.

polynomial protection (WPP) as the same model than PP except that the adversary has no access to the server.

Definition 14 (PP and WPP). *Let Π be a PPE, $\mathcal{A}$ be a probabilistic polynomial time (PPT) adversary. $\forall k \in \mathbb{N}$, the* k-Polynomial Protection *(k-PP) experiment for $\mathcal{A}$ against Π denoted by $Exp^{k\text{-}PP}_{\Pi,\mathcal{A}}(\lambda)$ is defined in Figure 4.2, where $\mathcal{A}$ has access to the* server oracle $\mathsf{CO}_{PP}(\cdot)$. *We define the advantage of the adversary $\mathcal{A}$ against the k-PP experiment by:*

$$\mathsf{Adv}^{k\text{-}PP}_{\Pi,\mathcal{A}}(\lambda) = Pr\left[1 \leftarrow Exp^{k\text{-}PP}_{\Pi,\mathcal{A}}(\lambda)\right]$$

A scheme Π is k-PP secure if this advantage is negligible for any $\mathcal{A} \in \textsc{poly}(\lambda)$.

We define the k-Weak Polynomial Protection *(k-WPP) experiment as the k-PP experiment except that $\mathcal{A}$ does not have access to the oracle $\mathsf{CO}_{PP}(\cdot)$. In a similar way, we* *define the* WPP *advantage and security.*

The only difference between PP and WPP is that the adversary has no access to the oracle in WPP. Thus PP security implies the WPP security.

Theorem 3. *For any* Π *and* k*, if* Π *is* k*-*PP *secure then* Π *is* k*-*WPP *secure.*

Proof. Let k be an integer and Π be a PPE. Let $\mathcal{A} \in \text{POLY}(\lambda)$ such that $\mathsf{Adv}^{k\text{-WPP}}_{\mathcal{A},\Pi}(\lambda)$ is non-negligible. We observe that $\mathcal{A}$ can be used as adversary who does not use the oracle $\mathsf{compute}(\mathsf{pub}, \mathsf{vk}, \cdot, \mathsf{sk}, f)$ for the k-PP experiment on Π. In this case, we have $\mathsf{Adv}^{k\text{-PP}}_{\mathcal{A},\Pi}(\lambda) = \mathsf{Adv}^{k\text{-WPP}}_{\mathcal{A},\Pi}(\lambda)$ which is non-negligible. By contraposition, if $\mathsf{Adv}^{k\text{-PP}}_{\mathcal{A},\Pi}(\lambda)$ is negligible then $\mathsf{Adv}^{k\text{-WPP}}_{\mathcal{A},\Pi}(\lambda)$ is negligible. $\square$

4.2.2 Chosen Function Attack

We define a model for *indistinguishability against chosen function attack*. In this model, the adversary chooses two polynomials (f_0, f_1) and tries to guess what is the polynomial f_b used by the server, where $b \in \{0,1\}$. The adversary has access to the server that evaluates and proves the soundness of $y = f_b(x)$, only if $f_0(x) = f_1(x)$. This is an inherent limitation, indeed, if the adversary can evaluate another point (x, y) such that $f_0(x) \neq f_1(x)$ then he can obviously compare y with $f_0(x)$ and $f_1(x)$ in order to guess b. In practice, an adversary chooses (f_0, f_1) such that $f_0 \neq f_1$, but with k points (x_i, y_i) such that $f_0(x_i) = f_1(x_i)$. It allows the adversary to maximize its oracle's calls in order to increase its chances of success. We remark that the schemes [49] and [51] are not IND-CFA: users know a value M and the set of inputs $\mathcal{X}$ such that $\forall x \in \mathcal{X}, f(x) < M$. An attacker may choose two polynomials f_0 and f_1 such that for a chosen a, $f_0(a) < M$ and $f_1(a) > M$. Since $\mathcal{X}$ is public, the attacker returns f_0 if and only if $a \in \mathcal{X}$. Then the attacker chooses f_0 and f_1 such that for a chosen a, $f_0(a) < M$ and $f_1(a) > M$. Then if $a \in \mathcal{X}$ it knows that the polynomial is f_0.

Definition 15 (IND-CFA)**.** *Let* Π *be a PPE,* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *be a two-party PPT adversary and* k *be an integer. The* k-Indistinguishability against Chosen Function Attack *(*k*-IND-CFA) experiment for* $\mathcal{A}$ *against* Π *is defined in Figure 4.2, where* $\mathcal{A}$ *has access to* *the* server oracle $\mathsf{CO}_{CFA}(\cdot)$. *The advantage of the adversary $\mathcal{A}$ against the k-IND-CFA experiment is given by:*

$$\mathsf{Adv}_{\Pi,\mathcal{A}}^{k\text{-}IND\text{-}CFA}(\lambda) = \left|\frac{1}{2} - Pr\left[1 \leftarrow Exp_{\Pi,\mathcal{A}}^{k\text{-}IND\text{-}CFA}(\lambda)\right]\right|$$

A scheme Π is k-IND-CFA secure if this advantage is negligible for any $\mathcal{A} \in \text{POLY}(\lambda)^2$.

In Theorem 4, we prove that the IND-CFA security implies the WPP security: if there exists an adversary $\mathcal{A}$ against the WPP experiment who is able to *decrypt* a random polynomial from the public values, then we can use it to guess f_b in an IND-CFA experiment for any chosen polynomials (f_0, f_1). However, surprisingly it is not true for the PP security (Theorem 5). The reason is that the oracle of the IND-CFA experiment has restriction, then it cannot be used to simulate the oracle of the PP experiment in a security reduction.

Theorem 4. *Let Π be an k-IND-CFA secure PPE then it is k-WPP secure.*

Proof. Let k be an integer and Π be a PPE. Suppose that there exists $\mathcal{A} \in \text{POLY}(\lambda)$ such that $\delta(\lambda) = \mathsf{Adv}_{\Pi,\mathcal{A}}^{k\text{-WPP}}(\lambda)$ is non-negligible. We show how to build $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2) \in \text{POLY}(\lambda)^2$ such that $\mathsf{Adv}_{\mathcal{B},\Pi}^{k\text{-IND-CFA}}(\lambda)$ is non-negligible, then we conclude the proof by contraposition. Algorithm $\mathcal{B}$ works as follows:

$\mathcal{B}_1(\mathsf{pub}, F, k)$**:** $\mathcal{B}_1$ picks $f_0 \xleftarrow{\$} F[X]_k$ and $r \xleftarrow{\$} F^*$, and builds f_1 such that $f_1(x) = f_0(x) + r$. Remark that $\forall x, f_0(x) \neq f_1(x)$. $\mathcal{B}_1$ returns $(f_0, f_1, \perp)$.

$\mathcal{B}_2(\mathsf{pub}, \mathsf{vk}, F, k, \perp)$ It runs $(x_*, y_*) \leftarrow \mathcal{A}(\mathsf{pub}, \mathsf{vk}, F, k)$. If $\exists\, b_* \in \{0,1\}$ such that $y_* = f_{b_*}(x_*)$ then it returns b_*; else, it returns $b_* \xleftarrow{\$} \{0,1\}$.

We evaluate the probability that $\mathcal{B}$ wins the experiment, *i.e.* $b_* = b$. First, we remark that if $\mathcal{A}$ wins his experiment, then $b_* = b$ with probability 1. On the other hand, if $\mathcal{A}$ does not win the experiment, we consider two different cases:

1. $\mathcal{A}$ returns (x_*, y_*) such that $f_{1-b}(x_*) = y_*$. The probability that $\mathcal{A}$ returns such a point is at most $1/|F|$ and is negligible for sufficiently large ring F. Indeed, since $\mathcal{A}$ has no information about f_{1-b}, his best strategy to guess a point of f_{1-b} is to randomly pick a point in F^2. In this case, $\mathcal{B}$ wins the experiment with probability 0.

2. $\mathcal{A}$ returns (x_*, y_*) such that $f_{1-b}(x_*) \neq y_*$. The probability that $\mathcal{A}$ returns such a point is at least $1 - 1/|F|$. In this case, $\mathcal{B}$ wins the experiment with probability $1/2$.

We recall that $\epsilon(\lambda) = \Pr[f_{1-b}(x_*) = y_*] \leq 1/|F|$. We have:

$$\begin{aligned}
\Pr[b = b_*] &= \Pr[f_b(x_*) = y_*] \cdot \Pr[b = b_* | f_b(x_*) = y_*] \\
&\quad + \Pr[f_b(x_*) \neq y_*] \cdot \Pr[b = b_* | f_b(x_*) \neq y_*] \\
&= \delta(\lambda) \cdot 1 + (1 - \delta(\lambda)).\Pr[b = b_* | f_b(x_*) \neq y_*] \\
&= \delta(\lambda) + (1 - \delta(\lambda)) \cdot (\Pr[f_{1-b}(x_*) = y_*] \\
&\quad \cdot \Pr[b = b_* | f_{1-b}(x_*) = y_*] + \Pr[f_{1-b}(x_*) \neq y_*] \\
&\quad \cdot \Pr[b = b_* | f_b(x_*) \neq y_* \text{ and } f_{1-b}(x_*) \neq y_*]) \\
&= \delta(\lambda) + (1 - \delta(\lambda)) \cdot \left(0 + (1 - \epsilon(\lambda)) \cdot \frac{1}{2}\right) \\
&= \frac{\delta(\lambda)}{2} + \epsilon(\lambda) \cdot \frac{1 - \delta(\lambda)}{2} + \frac{1}{2}
\end{aligned}$$

We deduce the advantage of $\mathcal{B}$:

$$\mathsf{Adv}^{k\text{-IND-CFA}}_{\mathcal{B},\Pi}(\lambda) = \left|\Pr[b = b_*] - \frac{1}{2}\right| \geq \frac{\delta(\lambda)}{2} + \frac{\delta(\lambda) - 1}{2|F|}$$

Since $\delta(\lambda)$ is non negligible, then $-\frac{\delta(\lambda)-1}{2|F|}$ is negligible for sufficiently large ring F and $\mathsf{Adv}^{k\text{-IND-CFA}}_{\mathcal{B},\Pi}(\lambda) = \delta/2$ is non-negligible. □

Theorem 5. *Let Π be a k-IND-CFA secure PPE, it does not imply that Π is k-PP.*

Proof. Let k be an integer and let $\Pi = (\mathsf{setup}, \mathsf{init}, \mathsf{compute}, \mathsf{verif})$ be a PPE that is k-PP and k-IND-CFA secure. Let $\Pi' = (\mathsf{setup}', \mathsf{init}', \mathsf{compute}', \mathsf{verif}')$ such that:

$\mathsf{setup}'(\lambda)$**:** It returns $(\mathsf{pub}, F) \leftarrow \mathsf{setup}(\lambda)$.

$\mathsf{init}'(\mathsf{pub}, f)$**:** It runs $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{init}(\mathsf{pub}, f)$, picks α in the uniform distribution on the inputs set of f and returns $(\mathsf{sk}', \mathsf{vk}') = (\mathsf{sk}, (\mathsf{vk}, \alpha))$.

$\mathsf{compute}'(\mathsf{pub}, \mathsf{vk}', x, \mathsf{sk}', f)$**:** This algorithm runs $(y, \pi) \leftarrow \mathsf{compute}(\mathsf{pub}, \mathsf{vk}, x, \mathsf{sk}, f)$. If $x \neq \alpha$ then sets $\pi' = (\pi, \perp)$; else, $\pi' = (\pi, f)$. Returns (y, π').

$\mathsf{verif}'(\mathsf{pub}, \mathsf{vk}', x, y, \pi')$: It returns $b \leftarrow \mathsf{verif}(\mathsf{pub}, \mathsf{vk}, x, y, \pi)$.

To prove the theorem, we show that (i) Π' is k-IND-CFA secure and (ii) Π' is not k-PP secure.

i) We prove it by contraposition. Suppose that there exists $\mathcal{A} \in \text{POLY}(\lambda)^2$ such that $\delta(\lambda) = \mathsf{Adv}^{k\text{-IND-CFA}}_{\Pi',\mathcal{A}}(\lambda)$ is non-negligible. We remark that during the IND-CFA experiment, the adversary $\mathcal{A}$ chooses his two polynomials (f_0, f_1) before α is chosen. Then the probability that $f_0(\alpha) = f_1(\alpha)$ is negligible for sufficiently large set F:

$$\Pr[f_0(\alpha) = f_1(\alpha)] \leq \frac{k}{|F|}$$

When $f_0(\alpha) \neq f_1(\alpha)$, α is randomly chosen and gives no additional information to $\mathcal{A}$. $\mathcal{A}$ cannot call the oracle $\mathsf{CO}_{\mathsf{CFA}}(\cdot)$ using α as input since $f_0(\alpha) \neq f_1(\alpha)$. Thus α is useless, in this case the experiments $\mathsf{Exp}^{k\text{-IND-CFA}}_{\Pi,\mathcal{A}}(\lambda)$ and $\mathsf{Exp}^{k\text{-IND-CFA}}_{\Pi',\mathcal{A}}(\lambda)$ are equivalent. We remark that:

$$\begin{aligned}
&\Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{IND\text{-}CFA}}_{\Pi',\mathcal{A}}(\lambda)\right] = \\
&\Pr\left[f_0(x) = f_1(x)\right] \cdot \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{IND\text{-}CFA}}_{\Pi',\mathcal{A}}(\lambda) | f_0(x) = f_1(x)\right] \\
&+ \Pr\left[f_0(x) \neq f_1(x)\right] \cdot \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{IND\text{-}CFA}}_{\Pi',\mathcal{A}}(\lambda) | f_0(x) \neq f_1(x)\right]
\end{aligned}$$

We then evaluate the probability that $\mathcal{A}$ breaks π:

$$\begin{aligned}
&\Pr[1 \leftarrow \mathsf{Exp}^{\mathsf{IND\text{-}CFA}}_{\Pi',\mathcal{A}}(\lambda)] = \frac{1}{\Pr[f_0(x) \neq f_1(x)]} \cdot \\
&\Big(\Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{IND\text{-}CFA}}_{\Pi',\mathcal{A}}(\lambda)\right] - \\
&\Pr\left[f_0(x) = f_1(x)\right] \cdot \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{IND\text{-}CFA}}_{\Pi',\mathcal{A}}(\lambda) | f_0(x) = f_1(x)\right]\Big) \\
&\geq \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{IND\text{-}CFA}}_{\Pi'}(\lambda)\right] - \Pr\left[f_0(x) = f_1(x)\right] \\
&\geq \frac{1}{2} \pm \delta(\lambda) - \frac{k}{|F|}
\end{aligned}$$

Hence $\mathsf{Adv}^{k\text{-IND-CFA}}_{\Pi,\mathcal{A}}(\lambda) \geq \delta(\lambda) - \frac{k}{|F|}$ is non-negligible, which contradicts that Π is k-IND-CFA.

ii) We construct $\mathcal{A} \in \text{POLY}(\lambda)$ such that $\mathsf{Adv}^{k\text{-PP}}_{\Pi',\mathcal{A}}(\lambda)$ is non-negligible: $\mathcal{A}$ receives $(\mathsf{pub}, (\mathsf{vk}, \alpha), F, k)$ as input and uses α as input for the oracle $\mathsf{CO}_{\mathsf{CFA}}(.)$ that returns $(f(\alpha), (\pi, f))$. $\mathcal{A}$ chooses $x \in F$ and returns $(x, f(x))$. Thus, $\mathsf{Adv}^{k\text{-}\mathsf{PP}}_{\Pi', \mathcal{A}}(\lambda) = 1$.

□

However, we would like to have a simple and sufficient condition under which the IND-CFA security implies the PP security. For this, we define the *proof inducted by a PPE* which is the NIZKP used by the algorithm compute. We show that if this NIZKP is zero-knowledge, then the IND-CFA security implies the PP security.

Definition 16. *Let* $\Pi = (\mathsf{setup}, \mathsf{init}, \mathsf{compute}, \mathsf{verif})$ *be a PPE, the* non-interactive proof inducted by Π, *denoted* $P_\Pi = (\mathsf{proof}_\Pi, \mathsf{ver}_\Pi)$ *is defined as follows. For any* $\lambda, k \in \mathbb{N}$, $(\mathsf{pub}, F) \leftarrow \mathsf{setup}(\lambda)$, $f \in F[X]_k$ *and* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{init}(\mathsf{pub}, f)$*:*

$\mathsf{proof}_\Pi((\mathsf{pub}, \mathsf{vk}, x, y), (f, \mathsf{sk}))$**:** *returns* π, *where* $(y', \pi) \leftarrow \mathsf{compute}(\mathsf{pub}, \mathsf{vk}, x, \mathsf{sk}, f)$.

$\mathsf{ver}_\Pi((\mathsf{pub}, \mathsf{vk}, x, y), \pi)$**:** *runs* $b \leftarrow \mathsf{verif}(\mathsf{pub}, \mathsf{vk}, x, y, \pi)$ *and returns it.*

We say that Π *is* Zero-Knowledge *(ZK) if* P_Π *is Zero-Knowledge.*

Theorem 6. *Let* Π *be a ZK and k-IND-CFA secure PPE, then* Π *is k-PP secure.*

Proof. Let Π be a PPE which is zero-knowledge. $\forall k \in \mathbb{N}$, we assume that there exists $\mathcal{A} \in \text{POLY}(\lambda)$ such that $\delta(\lambda) = \mathsf{Adv}^{k\text{-}\mathsf{PP}}_{\Pi, \mathcal{A}}(\lambda)$ is non negligible and that for any $\mathcal{A} \in \text{POLY}(\lambda)^2$, $\mathsf{Adv}^{k\text{-}\mathsf{IND}\text{-}\mathsf{CFA}}_{\Pi, \mathcal{A}}(\lambda)$ is negligible. We show that there exists $\mathcal{B} \in \text{POLY}(\lambda)^2$ such that $\mathsf{Adv}^{k\text{-}\mathsf{IND}\text{-}\mathsf{CFA}}_{\Pi, \mathcal{B}}(\lambda)$ is non-negligible. We obtain a contradiction then we deduce that for any ZK PPE Π such that $\mathsf{Adv}^{k\text{-}\mathsf{IND}\text{-}\mathsf{CFA}}_{\Pi, \mathcal{A}}(\lambda)$ is negligible for any $\mathcal{A} \in \text{POLY}(\lambda)^2$, then $\mathsf{Adv}^{k\text{-}\mathsf{PP}}_{\Pi, \mathcal{A}}(\lambda)$ is negligible for any $\mathcal{A} \in \text{POLY}(\lambda)$.

By hypothesis Π is zero-knowledge then there exists an algorithm Sim such that the outputs of $\mathsf{proof}_\Pi((\mathsf{pub}, \mathsf{vk}, x, y), (f, \mathsf{sk}))$ and $\mathrm{Sim}((\mathsf{pub}, \mathsf{vk}, x, y))$ follow the same probability distribution for any instance $(\mathsf{pub}, \mathsf{vk}, x, y)$ and the corresponding secret (f, sk). We build the following adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$:

$\mathcal{B}_1(\mathsf{pub}, F, k)$ It picks $f_0, f_1 \xleftarrow{\$} F[X]_k$ and returns $(f_0, f_1, \perp)$.

$\mathcal{B}_2(\mathsf{pub}, \mathsf{vk}, F, k, \perp)$ It picks $b' \xleftarrow{\$} \{0, 1\}$:

- It runs $(x_*, y_*) \leftarrow \mathcal{A}(\mathsf{pub}, \mathsf{vk}, F)$ and simulates the oracle $\mathsf{CO}_{\mathsf{PP}}(\cdot)$ as follows: on input x, $\mathcal{B}_2$ computes $y = f_{b'}(x)$ and runs $\pi \leftarrow \mathsf{Sim}((\mathsf{pub}, \mathsf{vk}, x, y))$. It then returns (y, π) to $\mathcal{A}$.
- If $f_{b'}(x_*) = y_*$ then $\mathcal{B}_2$ returns $b_* = b'$; else, it picks $b_* \overset{\$}{\leftarrow} \{0, 1\}$ and returns b_*.

We evaluate the probability that $\mathcal{B}$ wins the experiment, *i.e.* $b_* = b$ where b is the challenge of $\mathcal{B}$:

- if $b' = b$, then the PP experiment is perfectly simulated and $\mathcal{A}$ returns (x_*, y_*) such that $f_{b'}(x_*) = y_*$ with non negligible probability. Then $\mathcal{B}$ wins the experiment with non-negligible advantage. Remark that in this case, the probability that $b_* = b$ is the same as in the proof of Theorem 4:

$$\mathsf{Pr}\left[b_* = b | b' = b\right] = \frac{\delta(\lambda)}{2} + \epsilon(\lambda) \cdot \frac{1 - \delta(\lambda)}{2} + \frac{1}{2}$$

  where $\epsilon(\lambda) = \mathsf{Pr}[f_{1-b}(x_*) = y_*] \leq 1/|F|$.

- If $b' \neq b$, then the probability that $\mathcal{A}$ returns (x_*, y_*) such that $f_{b'}(x_*) = y_*$ is negligible: $\mathcal{A}$ knows at most k points of the polynomial $f_{b'}$, then his best strategy to find another point is to pick (x_*, y_*) randomly in F^2. Then $\mathcal{B}$ wins the experiment with negligible advantage. More formally, we have:

$$\begin{aligned}
&\mathsf{Pr}\left[b_* = b | b' \neq b\right] \\
&= \mathsf{Pr}[f_b(x_*) = y_*] \cdot \mathsf{Pr}[b = b_* | b' \neq b \text{ and } f_b(x_*) = y_*] \\
&+ \mathsf{Pr}[f_b(x_*) \neq y_*] \cdot \mathsf{Pr}[b = b_* | b' \neq b \text{ and } f_b(x_*) \neq y_*] \\
&= 0 + (1 - \epsilon(\lambda)) \cdot \frac{1}{2} = \frac{1}{2} - \frac{\epsilon(\lambda)}{2}
\end{aligned}$$

Finally, we have:

$$\begin{aligned}
\mathsf{Pr}[b_* = b] &= \mathsf{Pr}\left[b' = b\right] \cdot \mathsf{Pr}\left[b_* = b | b' = b\right] \\
&\quad + \mathsf{Pr}\left[b' \neq b\right] \cdot \mathsf{Pr}\left[b_* = b | b' \neq b\right] \\
&= \frac{1}{2} \cdot \left(\frac{\delta(\lambda)}{2} + \epsilon(\lambda) \cdot \frac{1 - \delta(\lambda)}{2} + \frac{1}{2} + \frac{1}{2} - \frac{\epsilon(\lambda)}{2} \right) \\
&= \frac{\delta(\lambda)}{4} - \epsilon(\lambda) \cdot \frac{\delta(\lambda)}{4} + \frac{1}{2}
\end{aligned}$$

Finally, we show that $\mathcal{B}$ have a non-negligible advantage for sufficiently large F:

$$\begin{aligned}
\mathsf{Adv}^{k\text{-}\mathsf{IND\text{-}CFA}}_{\Pi,\mathcal{B}}(\lambda) &= \left| \mathsf{Pr}[b_* = b] - \frac{1}{2} \right| \\
&= \frac{\delta(\lambda)}{4} - \epsilon(\lambda) \cdot \frac{\delta(\lambda)}{4} \\
&\geq \frac{\delta(\lambda)}{4} - \frac{\delta(\lambda)}{4 \cdot |F|}
\end{aligned}$$

□

In Figure 4.3, we illustrates all relations between our security properties.

$$\begin{tikzcd}
& k\text{-IND-CFA} \arrow[r, Rightarrow] \arrow[d, dash] \arrow[rd, Rightarrow, "/" marking] & k\text{-WPP} \\
\text{ZK} \arrow[r, dash] & \texttt{and} \arrow[r, Rightarrow] & k\text{-PP} \arrow[u, Rightarrow]
\end{tikzcd}$$

Figure 4.3: Security relations.

4.2.3 Unforgeability

We define the unforgeability property for a PPE. A PPE is unforgeable when a dishonest server cannot produce a valid proof on the point (x, y) when $f(x) \neq y$. The secret polynomial f is chosen by the server.

Definition 17. *Let Π be a PPE, $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a two-party PPT adversary. The* Unforgeability *(UNF) experiment for $\mathcal{A}$ against Π is defined in Figure 4.2. We define* *the advantage of the adversary $\mathcal{A}$ against the UNF experiment by:*

$$\mathsf{Adv}^{UNF}_{\Pi,\mathcal{A}}(\lambda) = Pr\left[1 \leftarrow Exp^{UNF}_{\Pi,\mathcal{A}}(\lambda)\right]$$

A scheme Π is UNF secure if this advantage is negligible for any $\mathcal{A} \in \text{POLY}(\lambda)^2$.

4.2.4 Security Against Collusion Attacks

There are two possible collusion scenarios: the collusion of a client and the server, and collusion of two or more clients.

Scenario 1: In a collusion of a client and the server, the server can provide the secret polynomial to the client. This is an inherent problem and cannot be prevented. The client can share public parameters and verification keys with the server but these parameters are already public and known to the server. The collusion does not give any advantage to the server to forge fake proof of computation.

Scenario 2: In a collusion of two or more clients, sharing Paillier secret keys with each other does not provide any information about the secret polynomial. All the verification keys and public parameters are the same for each client. The inherent limitation is that the collusion of clients can share their evaluated points and if the total number of points is more than k, where k is the degree of the secret polynomial, then clients can derive the polynomial. This problem exists in any polynomial computation and cannot be prevented.

4.3 Verifiable Private Polynomial Evaluation

Feldman's VSS can be used to design a PPE that is k-PP secure: using the public values g and $\{h_i\}_{0\leq i\leq k}$, any user can check that the point (x, y) computed by the server is a point of f. However, in a practical use, the polynomial f is not randomly chosen in a large set. An IND-CFA attacker knows that $f = f_0$ or $f = f_1$ for two known polynomials (f_0, f_1), since he knows the coefficients $\{a_{0,i}\}_{0\leq i\leq k}$ and

$\{a_{1,i}\}_{0\leq i\leq k}$ of these two polynomials, he can compute the values $\{g^{a_{0,i}}\}_{0\leq i\leq k}$ and $\{g^{a_{1,i}}\}_{0\leq i\leq k}$ and he can compare it with the public set $\{h_i\}_{0\leq i\leq k}$.

In order to construct our k-IND-CFA PPE, called PIPE, we give an ElGamal key pair $(\mathsf{pk}, \mathsf{sk})$ to the server and we encrypt all the h_i. Then for all $i \in \{0, \dots, k\}$, the users do not know $h_i = g^{a_i}$ but know the ElGamal ciphertext (c_i, d_i) such that $c_i = g^{r_i}$ and $d_i = \mathsf{pk}^{r_i} \cdot h_i$, where r_i is randomly chosen. Since ElGamal is IND-CPA secure, an attacker that chooses two polynomials (f_0, f_1) cannot distinguish, for $0 \leq i \leq k$, if the ciphertext (c_i, d_i) encrypts a coefficient of f_0 or of f_1. Thus, the attacks on the previous scheme are no longer possible.

Moreover, the user can check that $f(x) = y$ for a point (x, y) using the values $\{(c_i, d_i)\}_{0\leq i\leq k}$. We set $r(x) = \sum_{i=0}^{k} r_i \cdot x^i$. The user computes:

$$c = \prod_{i=0}^{k} c_i^{x^i} = \prod_{i=0}^{k} g^{r_i \cdot x^i} = g^{\sum\limits_{i=0}^{k} r_i \cdot x^i} = g^{r(x)}.$$

On the other hand, he computes:

$$\begin{aligned} d' = \prod_{i=0}^{k} d_i^{x^i} &= \left(\prod_{i=0}^{k} \mathsf{pk}^{r_i \cdot x^i}\right) \cdot \left(\prod_{i=0}^{k} g^{a_i \cdot x^i}\right) \\ &= \mathsf{pk}^{\sum\limits_{i=0}^{k} r_i \cdot x^i} \cdot g^{\sum\limits_{i=0}^{k} a_i \cdot x^i} \\ &= \mathsf{pk}^{r(x)} \cdot g^{f(x)}. \end{aligned}$$

Finally, $(c, d') = (g^{r(x)}, \mathsf{pk}^{r(x)} \cdot g^{f(x)})$ is an ElGamal ciphertext of $g^{f(x)}$. Then, to convince the user that (x, y) is a valid point of f, the server proves that (c, d') is a ciphertext of g^y using a NIZKP of $\log_g(c) = \log_{\mathsf{pk}}(d'/g^y)$.

4.3.1 Construction of PIPE

Definition 18. *Let* $\mathsf{PIPE} = (\mathsf{setup}, \mathsf{init}, \mathsf{compute}, \mathsf{verif})$ *be a PPE defined by:*

$\mathsf{setup}(\lambda)$**:** *Using the security parameter λ, it generates G a group of prime order p and a generator $g \in G$. It chooses a hash function $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_p^*$ and it sets $F = \mathbb{Z}_p^*$. It sets $\mathsf{pub} = (G, p, g, H)$ and returns (pub, F).*

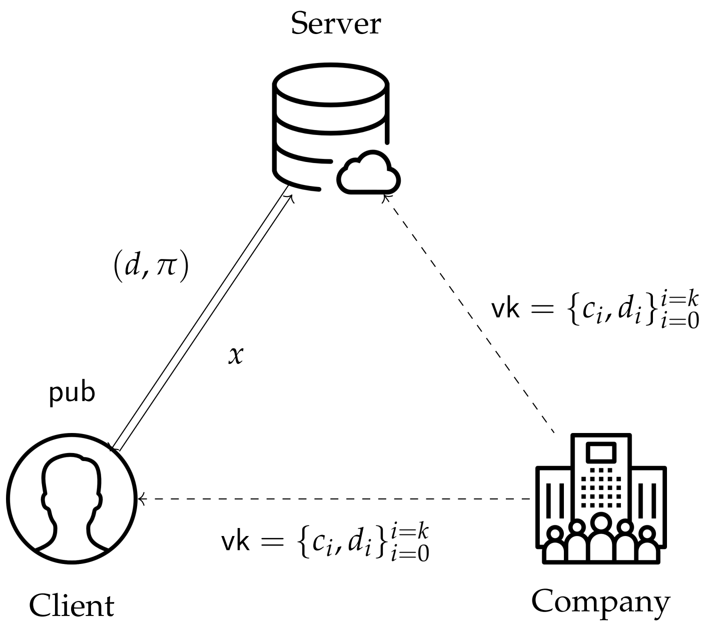

Server
(d, π)
x
pub
$\mathsf{vk} = \{c_i, d_i\}_{i=0}^{i=k}$
$\mathsf{vk} = \{c_i, d_i\}_{i=0}^{i=k}$
Client
Company

Figure 4.4: Illustration of the PIPE scheme.

$\mathsf{init}(\mathsf{pub}, f)$**:** *We set* $f(x) = \sum_{i=0}^{k} a_i \cdot x^i$*. This algorithm picks* $\mathsf{sk} \xleftarrow{\$} \mathbb{Z}_p^*$ *and computes* $\mathsf{pk} = g^{\mathsf{sk}}$*. For all* $i \in \{0, \ldots, k\}$*, it picks* $r_i \xleftarrow{\$} \mathbb{Z}_p^*$ *and computes* $c_i = g^{r_i}$ *and* $d_i = \mathsf{pk}^{r_i} \cdot g^{a_i}$*. It sets* $\mathsf{vk} = (\{(c_i, d_i)\}_{0 \le i \le k}, \mathsf{pk})$ *and returns* $(\mathsf{vk}, \mathsf{sk})$*.*

$\mathsf{compute}(\mathsf{pub}, \mathsf{vk}, x, \mathsf{sk}, f)$**:** *Using* vk *which is equal to* $(\{(c_i, d_i)\}_{0 \le i \le k}, \mathsf{pk})$*, this algorithm picks* $\theta \xleftarrow{\$} \mathbb{Z}_p^*$*, computes:*

$$c = \prod_{i=0}^{k} c_i^{x^i} \quad \text{and} \quad \pi = (g^\theta, c^\theta, \theta + H(g^\theta, c^\theta) \cdot \mathsf{sk}).$$

It returns $(f(x), \pi)$*.*

$\mathsf{verif}(\mathsf{pub}, \mathsf{vk}, x, y, \pi)$**:** *Using* $\mathsf{vk} = (\{(c_i, d_i)\}_{0 \le i \le k}, \mathsf{pk})$ *and* $\pi = (A, B, \omega)$*, this algorithm computes*

$$c = \prod_{i=0}^{k} c_i^{x^i} \quad \text{and} \quad d = \frac{\left(\prod_{i=0}^{k} d_i^{x^i}\right)}{g^y}.$$

If $g^\omega = A \cdot \mathsf{pk}^{H(A,B)}$ *and* $c^\omega = B \cdot d^{H(A,B)}$*, then the algorithm returns* 1*; else, it returns* 0*.*

4.3.2 Security Analysis

We first show that PIPE is secure against chosen polynomial attacks under the DDH assumption [70].

Theorem 7. $\forall k \in \mathbb{N}$, PIPE *is k-IND-CFA under the DDH assumption in the ROM.*

Proof. We suppose that there exists $\mathcal{A} \in \text{POLY}(\lambda)^2$ such that $\mathsf{Adv}^{k\text{-}\mathsf{IND\text{-}CFA}}_{\mathsf{PIPE},\mathcal{A}}(\lambda)$ is non-negligible and we show that there exists an algorithm $\mathcal{B} \in \text{POLY}(\lambda)$ such that $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\text{ElGamal},\mathcal{B}}(\lambda)$ is non-negligible. We build $\mathcal{B}$ as follows:

- $\mathcal{B}$ receives (G, p, g, h) and runs $(f_0, f_1, \mathsf{st}) \leftarrow \mathcal{A}_1((G, p, g), \mathbb{Z}_p^*, k)$.
- For all $i \in \{0, \dots, k\}$, let $(a_{0,i}, a_{1,i})$ be the respective k^{th} coefficients of f_0 and f_1. $\mathcal{B}$ runs the oracle $\mathsf{Enc}_{\mathsf{pk}}(\mathsf{LR}_b(\cdot,\cdot))$ on input $(g^{a_{0,i}}, g^{a_{1,i}})$ and obtains the ElGamal ciphertext (c_i, d_i) of $g^{a_{b,i}}$.
- $\mathcal{B}$ runs $b_* \leftarrow \mathcal{A}_2((G, p, g), (\{(c_i, d_i)\}_{0 \le i \le k}, h), \mathbb{Z}_p^*, k, \mathsf{st})$. To simulate the oracle $\mathsf{CO}_{\mathsf{CFA}}(\cdot)$ on x to $\mathcal{A}$, $\mathcal{B}$ computes:

$$c = \prod_{i=0}^{k} c_i^{x^i} \quad \text{and} \quad d = \left(\prod_{i=0}^{k} d_i^{x^i}\right) \cdot \frac{1}{g^{f_0(x)}}.$$

  In the real experiment, the proof π is computed as in LogEq. Since this protocol is ZK, there exists a polynomial time simulator Sim in the ROM such that the outputs of the simulator come from the same distribution that the outputs of the real proof algorithm. Then $\mathcal{B}$ computes $y = f_0(x)$. If $y = f_1(x)$ it uses $\mathsf{Sim}((g, h, c, d))$ to compute π and returns (y, π) to $\mathcal{A}$; else, it returns $\perp$.
- Finally, $\mathcal{B}$ outputs b_*.

We observe that:

1. The experiment k-IND-CFA is perfectly simulated for $\mathcal{A}$.
2. $\mathcal{B}$ wins the IND-CPA experiment if and only if $\mathcal{A}$ wins the k-IND-CPA experiment.

Since $\mathsf{Adv}^{k\text{-}\mathsf{IND\text{-}CFA}}_{\mathsf{PIPE},\mathcal{A}}(\lambda)$ is non-negligible, then $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathrm{ElGamal},\mathcal{B}}(\lambda)$ is non-negligible. Since ElGamal cryptosystem is IND-CPA secure under the DDH assumption, $\mathcal{B}$ can be used to break the DDH assumption, which contradict our hypothesis and conclude the proof. □

Using Theorem 6, we only need to prove that PIPE is ZK to have that it is *k-PP*.

Theorem 8. *For any $k \in \mathbb{N}$,* PIPE *is ZK in the random oracle model.*

Proof. Let P_{PIPE} the proof inducted by P. We show that for any $\lambda, k \in \mathbb{N}$, $(\mathsf{pub}, F) \leftarrow \mathsf{setup}(\lambda)$, $f \in F[X]_k$ and $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{init}(\mathsf{pub}, f)$, there exists a simulator denoted $\mathsf{sim}(\mathsf{pub}, \mathsf{vk}, x, y)$ that outputs values in the same distribution as the following algorithm $\mathsf{proof}_{\mathsf{PIPE}}((\mathsf{pub}, \mathsf{vk}, x, y), (f, \mathsf{sk}))$.

- sim picks $(\omega, h) \xleftarrow{\$} (\mathbb{Z}_p^*)^2$ and computes:

$$A = g^{\omega}/\mathsf{pk}^h \quad \text{and} \quad B = \frac{\left(\prod_{i=0}^{k} c_i^{x^i}\right)^{\omega}}{\left(\left(\prod_{i=0}^{k} d_i^{x^i}\right) \cdot \frac{1}{g^y}\right)^h}.$$

- It adds the pair of input/output $((A, B), h)$ to the table of the random oracle H and it returns (A, B, ω). Since ω and h come from the uniform distribution on $\mathbb{Z}_p^*$, then sim and $\mathsf{proof}_{\mathsf{PIPE}}$ induct similar distributions.

□

Theorem 9. PIPE *is unconditionally UNF secure in the ROM.*

Proof. The proof π is computed as in LogEq (Definition 9). This NIZKP scheme is unconditionally *sound*, then there exists no PPT algorithm that forges a valid proof on a false statement with non-negligible probability, *i.e.* a statement (g, pk, c, d), where $\log_g(\mathsf{pk}) \neq \log_c(d)$. We show that if there exists $\delta(\lambda) = \mathcal{A} \in \text{POLY}(\lambda)^2$ such that $\mathsf{Adv}^{\mathsf{UNF}}_{\mathsf{PIPE},\mathcal{A}}(\lambda)$ is non negligible, then there exists $\mathcal{B} \in \text{POLY}(\lambda)$ that forges a valid proof of an instance, where $\log_g(\mathsf{pk}) \neq \log_c(d)$. It contradicts the soundness of LogEq which concludes the proof. $\mathcal{B}$ works as follows:

- It runs $(\mathsf{pub}, F) \leftarrow \mathsf{setup}(\lambda)$, $(f, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pub}, F)$, $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{init}(\mathsf{pub}, f)$, where $\mathsf{vk} = (\{(c_i, d_i)\}_{0 \leq i \leq k}, \mathsf{pk})$ and $(x, y, \pi) \leftarrow \mathcal{A}_2(\mathsf{pub}, \mathsf{sk}, \mathsf{vk}, F, f, \mathsf{st})$, where $\pi = (A, B, \omega)$.

- $\mathcal{B}$ computes:
$$c = \prod_{i=0}^{k} c_i^{x^i} \quad \text{and} \quad d = \left(\prod_{i=0}^{k} d_i^{x^i} \right) \cdot \frac{1}{g^y}$$
and builds the statement (g, pk, c, d). It returns the instance (g, pk, c, d) together with the proof π.

We observe that since $\mathsf{Adv}^{\mathsf{UNF}}_{\mathsf{PIPE},\mathcal{A}}(\lambda)$ is non negligible then the probability that $f(x) \neq y$ and $1 \leftarrow \mathsf{verif}(\mathsf{pub}, \mathsf{vk}, x, y, \pi)$ is non-negligible. Moreover:

i) $f(x) \neq y$ implies:
$$\begin{aligned} d = \left(\prod_{i=0}^{k} d_i^{x^i} \right) \cdot \frac{1}{g^y} &= c^{\mathsf{sk}} \cdot g^{f(x)-y} \\ &\neq c^{\mathsf{sk}} \end{aligned}$$
Then $\log_g(\mathsf{pk}) \neq \log_c(d)$.

ii) $1 \leftarrow \mathsf{verif}(\mathsf{pub}, \mathsf{vk}, x, y, \pi)$ implies $g^{\omega} = A \cdot \mathsf{pk}^{H(A,B)}$ and $c^{\omega} = B \cdot d^{H(A,B)}$. Then π is a valid proof.

Then $\mathcal{B}$ returns a valid proof of a false instance with non-negligible probability $\delta(\lambda)$. □

4.4 Conclusion

In this chapter, we gave a formal definition for a primitive called PPE, which allows a company to delegate computations on a secret polynomial for users in a verifiable way. In essence, the user sends x and receives y from the server, along with a proof that convinces him that $y = f(x)$, even though he does not know the polynomial f. We then defined security notions for PPE: Polynomial protection, Indistinguishability against Chosen Function Attack (IND-CFA), and unforgeability. The IND-CFA notion captures leakage of the polynomial, and the unforgeability notion captures forgery of the computation. We further note that there are two possible collusions in the system. The collusion of a user and the server does not give any advantage to the server. The collusion of two or more users can share their evaluations, and it is an inherent limitation that cannot be prevented.

We then presented PIPE, the first IND-CFA secure PPE. We used the Elgamal encryption scheme along with Verifiable Secret Sharing to achieve IND-CFA security. We then prove its IND-CFA security under the decisional Diffie-Hellman (DDH) assumption in the random oracle model (ROM).

CHAPTER 5

Verifiable and Private Oblivious Polynomial Evaluation

5.1 Introduction

PPE schemes do not protect the privacy of the clients: their data is handled in clear by the server. After the SingHealth hack, the company wants to be sure that even if an intruder hacks the server, he will not be able to steal the medical data of its clients. To solve this problem we propose a new primitive called *Private Oblivious Polynomial Evaluation* (VPOPE). A VPOPE scheme is a PPE scheme, in which the data of the client cannot be read by the cloud server. More precisely, the client sends his encrypted data to the server, and the server, returns encrypted evaluation of the prediction function $\mathcal{E}_{\mathsf{pk}}(f(x))$, as well as a proof π allowing the client to verify the correctness of the result with the help the verification key vk. However, during the process, the server never learns anything about x.

We propose a VPOPE scheme which allow a client to send encrypted data for a private evaluation in a verifiable way. Before we present our security model, we first formally define a Private Oblivious Polynomial Evaluation scheme.

Definition 19. *A* Verifiable and Private Oblivious Polynomial Evaluation *(VPOPE) scheme is composed of eight algorithms* $(\mathsf{setup}, \mathsf{init}, \mathsf{keyGen}, \mathsf{queryGen}, \mathsf{queryDec}, \mathsf{compute}, \mathsf{decrypt}, \mathsf{verif})$ *defined as follows:*

- $\mathsf{setup}(\eta)$: *Using the security parameter η, this algorithm generates a ring F, public parameters* pub *and secret parameters* sec*. It returns* $(\mathsf{pub}, F, \mathsf{sec})$.

- $\mathsf{init}(F, f, \mathsf{sec})$: *Using F, the secret polynomial f, and parameters* sec, *this algorithm returns a verification key* vk *and a server key* sk *associated to the secret polynomial f.*

- $\mathsf{keyGen}(\eta, \mathsf{pub}, k)$: *Using the security parameter η and public parameters* pub, *this algorithm generates and returns a client's key pair (pk_c, sk_c).*

- $\mathsf{queryGen}(pk_c, x)$: *Using a public key pk_c and an input x, this algorithm generates an encrypted query t associated to x, a proof π_t proving that t is a valid encrypted query, and returns (t, π_t).*

- $\mathsf{queryDec}(sk_c, t)$: *Using a secret key sk_c and an encrypted request t, this algorithm outputs x if t is a valid request of x, $\perp$ otherwise.*

- $\mathsf{compute}(t, \pi_t, f, \mathsf{sk}, F)$: *Using t, π_t, f,* sk, *and F, this algorithm returns an encrypted value d along with a proof π_d proving that d is an encryption of $f(x)$ if the proof π_t is "accepted". Else it returns $\perp$.*

- $\mathsf{decrypt}(sk_c, d)$: *Using a secret key sk_c and the encrypted value d, this algorithm returns y, the decryption of d.*

- $\mathsf{verif}(x, sk_c, \mathsf{pub}, y, \pi_d, \mathsf{vk})$: *This algorithm returns* 1 *if the proof π_d is "accepted",* 0 *otherwise.*

5.2 Security Models for VPOPE

We use security notions of PPE schemes formalized in previous section, namely *Polynomial Protection* (PP), *Unforgeability* (UNF), and *Indistinguishability against Chosen Function Attack* (IND-CFA), and adapt them to VPOPE schemes. Since VPOPE schemes consider encrypted data on client side, we consider the *Client's Privacy - Indistinguishability* (CPI) security property defined by Naor and Pinkas [16] to include the privacy on data client. Moreover, we define the *Query Correctness* (QC) notion in order to prove that a client cannot have other information than points that she queried.Since VPOPE schemes consider encrypted data on the client-side, we recall the *Client's Privacy - Indistinguishability* (CPI) security

```
$\mathsf{Exp}^{\mathsf{CPI}}_{\Pi,\mathcal{A}}(\eta)$:
    $b \xleftarrow{\$} \{0,1\}$ ;
    $(\mathsf{pub}, F, \mathsf{sec}) \leftarrow \mathsf{setup}(\eta)$ ;
    $f \xleftarrow{\$} F[X]^k$ ;
    $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{init}(F, f, \mathsf{sec})$ ;
    $(\mathsf{pk}_c, \mathsf{sk}_c) \leftarrow \mathsf{keyGen}(\eta, \mathsf{pub}, k)$ ;
    $(x_0, x_1, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pk}_c, \mathsf{pub}, F)$ ;
    $(t, \pi_t) \leftarrow \mathsf{queryGen}(\mathsf{pk}_c, x_b)$ ;
    $b_* \leftarrow \mathcal{A}_2^{\mathsf{CO}_{\mathsf{CPI}}(\cdot)}(t, f, \mathsf{sk}, F, \mathsf{st})$ ;
    return $(b = b_*)$ .

$\mathsf{CO}_{\mathsf{CPI}}(x)$:
    $(t, \pi_t) \leftarrow \mathsf{queryGen}(\mathsf{pk}_c, x)$ ;
    return $t$ .
```

Figure 5.1: CPI experiment.

property defined by Naor and Pinkas [16] to include the privacy of the client's data. Moreover, we define the *Query Soundness* (QS) notion to prove that a client cannot have other information than points that she queried. In all the security models, we denote by $F[x]^k$, the set of all polynomials of degree k over a finite field F.

5.2.1 Client's Privacy - Indistinguishability

We first recall the *Client's Privacy - Indistinguishability* (CPI) security for VPOPEschemes introduced by Naor and Pinkas [16]. In this model, the adversary chooses two queries (x_0, x_1) and tries to guess the evaluation x_b asked by the client. The adversary has access to the ciphertext oracle $\mathsf{CO}_{\mathsf{CPI}}(\cdot)$ taking x as input and returns the encrypted query t. A VPOPEscheme is CPI-secure if no adversary can output the query chosen by the client with a better probability than by guessing.

Definition 20 (Client's privacy - indistinguishability.)**.** *Let* Π *be a VPOPE,* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2) \in \text{POLY}(\eta)^2$ *be a two-party adversary. The* client's privacy - indistinguishability *(CPI) experiment for* $\mathcal{A}$ *against* Π *is defined in Fig. 5.1, where* $\mathcal{A}$ *has access to the oracle* $\mathsf{CO}_{CPI}(\cdot)$*. The advantage of the adversary* $\mathcal{A}$ *against the CPI experiment is given* *by:*

$$\mathsf{Adv}^{CPI}_{\Pi,\mathcal{A}}(\eta) = \left| \frac{1}{2} - \Pr\left[1 \leftarrow Exp^{CPI}_{\Pi,\mathcal{A}}(\eta)\right] \right| .$$

A scheme Π *is CPI-secure if this advantage is negligible for any* $\mathcal{A} \in \text{POLY}(\eta)^2$.

5.2.2 Chosen Function Attack

We recall the model for *k-Indistingui-shability against Chosen Function Attack* (*k*-

```
$\mathsf{Exp}^{k\text{-IND-CFA}}_{\Pi,\mathcal{A}}(\eta)$:
    $b \xleftarrow{\$} \{0,1\}$ ;
    $(\mathsf{pub}, F, \mathsf{sec}) \leftarrow \mathsf{setup}(\eta)$ ;
    $(\mathsf{pk}_c, \mathsf{sk}_c) \leftarrow \mathsf{keyGen}(\eta, \mathsf{pub}, k)$ ;
    $(f_0, f_1, \mathsf{st}_2) \leftarrow \mathcal{A}_1(\mathsf{pk}_c, \mathsf{pub}, F, k)$ ;
    $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{init}(F, f_b, \mathsf{sec})$ ;
    $b_* \leftarrow \mathcal{A}_2^{\mathsf{CO}_{\mathsf{CFA}}(\cdot)}(\mathsf{pk}_c, \mathsf{sk}_c \mathsf{pub}, F, \mathsf{vk}, k, \mathsf{st})$ ;
    if $f_0 \notin F[X]^k$ or $f_1 \notin F[X]^k$:
        then return $\perp$ ;
    else return $(b = b_*)$ .
```

Figure 5.2: IND-CFA experiment.

IND-CFA). In this model, the adversary chooses two polynomials (f_0, f_1) and tries to guess the polynomial f_b used by the server, where $b \in \{0,1\}$. The adversary has access to a server oracle $\mathsf{CO}_{\mathsf{CFA}}(\cdot)$ and sends to her an encrypted query t associated to her data x along with a proof π_t. The oracle decrypts the query t and obtains x if t is valid. If $f_0(x) = f_1(x)$, the oracle returns d i.e. the encrypted value of $f_b(x)$, along with a proof π_d.

If $f_0(x) \neq f_1(x)$, then the server returns nothing. In practice, an adversary chooses (f_0, f_1) such that $f_0 \neq f_1$, but with k points (x_i, y_i) such that $f_0(x_i) = f_1(x_i)$. It allows the adversary to maximize his oracle calls in order to increase his chances of success.

Definition 21. (k-IND-CFA)**.** *Let* Π *be a VPOPE,* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2) \in \text{POLY}(\eta)$ *be a two-party adversary and k be an integer. The k-IND-CFA experiment for* $\mathcal{A}$ *against* Π *is defined in Fig. 5.2, where* $\mathcal{A}$ *has access to the* server oracle $\mathsf{CO}_{CFA}(\cdot)$. *The advantage of*

```
$\mathsf{CO}_{\mathsf{CFA}}(t, \pi_t)$:
    $(d, \pi_d) \leftarrow \mathsf{compute}(t, \pi_t, f_b, \mathsf{sk}, F)$ ;
    if $x \leftarrow \mathsf{queryDec}(t, \mathsf{sk}_c)$ and $x \neq \perp$ and $f_0(x) = f_1(x)$:
        then return $(d, \pi_d)$ ;
    else return $\perp$ .
```

Figure 5.3: Server oracle for IND-CFA.

the adversary $\mathcal{A}$ against the k-IND-CFA experiment is given by:

$$\mathsf{Adv}^{\textit{k-IND-CFA}}_{\Pi,\mathcal{A}}(\eta) = \left| \frac{1}{2} - \Pr\left[1 \leftarrow \mathit{Exp}^{\textit{k-IND-CFA}}_{\Pi,\mathcal{A}}(\eta)\right] \right| .$$

A scheme Π is k-IND-CFA-secure if this advantage is negligible for any $\mathcal{A} \in \text{POLY}(\eta)^2$.

5.2.3 Unforgeability

Finally, we recall the unforgeability property. A VPOPEis unforgeable when a dishonest server cannot produce a valid proof for a point (x, y) such that $y \neq f(x)$. In this model, the secret polynomial f is chosen by the server.

```
$\mathsf{Exp}^{\mathsf{UNF}}_{\Pi,\mathcal{A}}(\eta)$:
    $(\mathsf{pub}, F, \mathsf{sec}) \leftarrow \mathsf{setup}(\eta)$ ;
    $(\mathsf{pk}_c, \mathsf{sk}_c) \leftarrow \mathsf{keyGen}(\eta, \mathsf{pub}, k)$ ;
    $(f, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pk}_c, \mathsf{sec})$ ;
    $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{init}(F, f, \mathsf{sec})$ ;
    $(x_*, y_*, \pi_*) \leftarrow \mathcal{A}_2(\mathsf{pub}, \mathsf{sk}, \mathsf{vk}, F, f, \mathsf{st})$ ;
    if $f(x_*) \neq y_*$ and $\mathsf{verif}(x_*, \mathsf{sk}_c, \mathsf{pub}, y_*, \pi_*, \mathsf{vk}) = 1$:
        then return 1 ;
    else return 0 .
```

Figure 5.4: UNF experiment.

Definition 22. (Unforgeability). *Let Π be a VPOPE, $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2) \in \text{POLY}(\eta)$ be a two-party adversary. The* unforgeability *(UNF) experiment for $\mathcal{A}$ against Π is defined in Fig. 5.4. We define the advantage of the adversary $\mathcal{A}$ against the UNF experiment by:*

$$\mathsf{Adv}^{\textit{UNF}}_{\Pi,\mathcal{A}}(\eta) = \Pr\left[1 \leftarrow \mathit{Exp}^{\textit{UNF}}_{\Pi,\mathcal{A}}(\eta)\right] .$$

A scheme Π is UNF-secure if this advantage is negligible for any $\mathcal{A} \in \text{POLY}(\eta)^2$.

5.2.4 Query Soundness

```
$\mathsf{Exp}^{\mathsf{QS}}_{\Pi,\mathcal{A}}(\eta)$:
    $(\mathsf{pub}, F, \mathsf{sec}) \leftarrow \mathsf{setup}(\eta)$ ;
    $f \xleftarrow{\$} F[X]^k$ ;
    $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{init}(F, f, \mathsf{sec})$ ;
    $(\mathsf{pk}_c, \mathsf{sk}_c) \leftarrow \mathsf{keyGen}(\eta, \mathsf{pub}, k)$ ;
    $(t, \pi_t) \leftarrow \mathcal{A}((\mathsf{pk}_c, \mathsf{sk}_c), \mathsf{pub}, F, \mathsf{vk})$ ;
vif $\mathsf{queryDec}(t) \neq \perp$ and $\mathsf{compute}(t, \pi_t, f, \mathsf{sk}, F) \neq \perp$
  and $f(\mathsf{queryDec}(\mathsf{sk}_c, t)) \neq \mathsf{decrypt}(\mathsf{sk}_c, d)$ such that
  $(d, \pi_d) \leftarrow \mathsf{compute}(t, \pi_t, f, \mathsf{sk}, F)$:
        then return 1 ;
    else return 0 .
```

Figure 5.5: QS experiment.

We now define a model for *Query Soundness* (QS). In this model, the adversary tries to learn other information than points of the secret polynomial that she queried by sending a particular query t along with a proof π_t to the server.

Definition 23 (Query Soundness)**.** *Let Π be a VPOPE, and $\mathcal{A} \in \text{POLY}(\eta)$ be an adversary. The* Query Soundness *(QS) experiment for $\mathcal{A}$ against Π is defined in Fig. 5.5. The advantage of the adversary $\mathcal{A}$ against the QS experiment is given by:*

$$\mathsf{Adv}^{\mathsf{QS}}_{\Pi,\mathcal{A}}(\eta) = \Pr\left[1 \leftarrow Exp^{\mathsf{QS}}_{\Pi,\mathcal{A}}(\eta)\right] .$$

A scheme Π is QS-secure if this advantage is negligible for any $\mathcal{A} \in \text{POLY}(\eta)$.

5.3 PPE for Encrypted data

We propose a Verifiable Oblivious IND-CFA Polynomial Evaluation scheme (VIP − POPE) scheme which allow a client to send encrypted data for a private evaluation in a verifiable way. We have used a Paillier cryptosystem, a homomorphic encryption scheme, to encrypt user's input in the VIP − POPE scheme. The security of the Paillier cryptosystem is based on the decisional composite residuocity assumption.

We first give the intuition of our scheme $\mathsf{VIP}-\mathsf{POPE}$ and then give its formal definition.

5.3.1 Intuition

We use homomorphic properties of Paillier's cryptosystem to design our VPOPE scheme called $\mathsf{VIP}-\mathsf{POPE}$. The key idea is to use the fact that a client can generate an encrypted query $t = \{t_i\}_{i=1}^{k}$ where $t_i = \mathcal{E}_{\mathsf{pk}}(x^i)$ and k is the degree of the secret polynomial $f(\cdot)$ to allow the server to compute $\mathcal{E}_{\mathsf{pk}}(f(x))$. Since the server knows coefficients $\{a_i\}_{i=0}^{k}$ of $f(\cdot)$, it computes $\mathcal{E}_{\mathsf{pk}}(f(x))$ as follows:

$$\mathcal{E}_{\mathsf{pk}}(a_0) \cdot \prod_{i=1}^{i=k} \mathcal{E}_{\mathsf{pk}}(x^i)^{a_i} = \prod_{i=0}^{i=k} \mathcal{E}_{\mathsf{pk}}(a_i x^i) = \mathcal{E}_{\mathsf{pk}}\left(\sum_{i=0}^{i=k} a_i x^i\right) = \mathcal{E}_{\mathsf{pk}}(f(x)) \ .$$

For instance, assume that $f(x) = \theta_0 + \theta_1 x + \theta_2 x^2$. Since the degree of $f(\cdot)$ is 2, the client can send two encrypted query. Let $t = (t_1, t_2)$ be the first encrypted query associated to $x = 3$, hence $(t_1, t_2) = (\mathcal{E}_{\mathsf{pk}}(3), \mathcal{E}_{\mathsf{pk}}(3^2)) = (\mathcal{E}_{\mathsf{pk}}(3), \mathcal{E}_{\mathsf{pk}}(9))$ Then, the server computes $\mathcal{E}_{\mathsf{pk}}(f(3)) = \mathcal{E}_{\mathsf{pk}}(\theta_0) \cdot \prod_{i=1}^{i=2} t_i^{\theta_i} = \mathcal{E}_{\mathsf{pk}}(\theta_0) \cdot \mathcal{E}_{\mathsf{pk}}(3)^{\theta_1} \cdot \mathcal{E}_{\mathsf{pk}}(9)^{\theta_2} = \mathcal{E}_{\mathsf{pk}}(\theta_0 + \theta_1 \cdot 3 + \theta_2 \cdot 9)$ and sends back the result to the client. Decrypting the result, the client knows one point to the secret polynomial $f(\cdot)$, namely $P_1 = (3, f(3))$.

Now assume the client forges an untrustworthy encrypted query $t' = (t'_1, t'_2)$ for the value $x = 4$ which is equal to $(\mathcal{E}_{\mathsf{pk}}(4), \mathcal{E}_{\mathsf{pk}}(9))$ instead of $(\mathcal{E}_{\mathsf{pk}}(4), \mathcal{E}_{\mathsf{pk}}(16))$. After the server computation and the decryption, the client obtains the value $\theta_0 + \theta_1 \cdot 4 + \theta_2 \cdot 9$ from t'. In this case, the client can deduce information on the secret polynomial, i.e., value of a coefficient that she should not obtain. The client is able to compute the value of the coefficient θ_1 using $P_1 = (3, \theta_0 + \theta_1 \cdot 3 + \theta_2 \cdot 9)$ and $\theta_0 + \theta_1 \cdot 4 + \theta_2 \cdot 9$. She computes $(\theta_0 + \theta_1 \cdot 4 + \theta_2 \cdot 9) - (\theta_0 + \theta_1 \cdot 3 + \theta_2 \cdot 9) = \theta_1$.

To avoid this kind of attack, the client must provide a proof of validity π_t for each query $t = \{t_i\}_{i=1}^{k}$ that she sends to the server, i.e., a proof that $t_i = \mathcal{E}_{\mathsf{pk}}(x^i)$ for all $i \in \{1, \ldots, k\}$. Based on Property 1, such a proof can be built using the NIZKP $\mathsf{DecPaillierEq}$ presented in Definition 8.

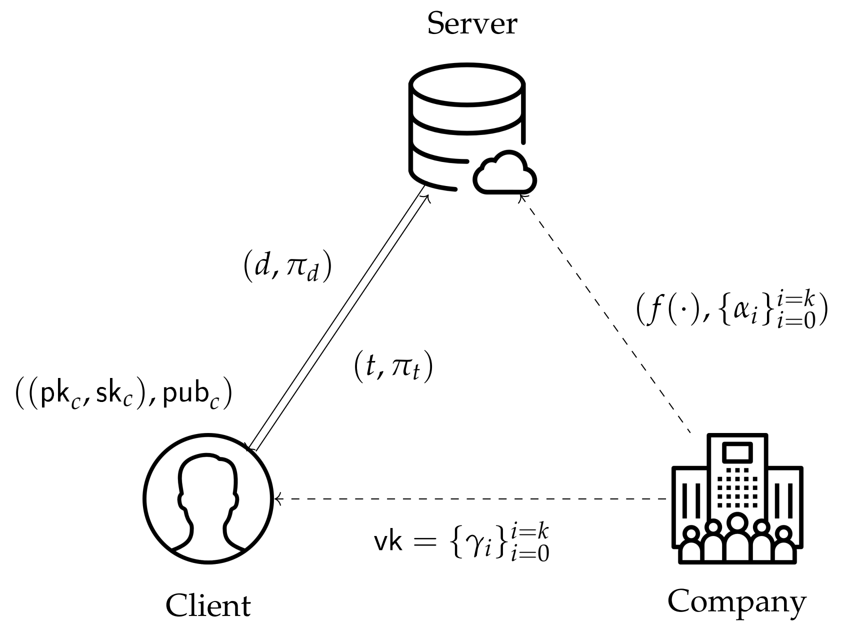

Server
(d, π_d)
$(f(\cdot), \{\alpha_i\}_{i=0}^{i=k})$
(t, π_t)
$((\mathsf{pk}_c, \mathsf{sk}_c), \mathsf{pub}_c)$
$\mathsf{vk} = \{\gamma_i\}_{i=0}^{i=k}$
Client
Company

Figure 5.6: Illustration of the $\mathsf{VIP-POPE}$ scheme.

In order to achieve the $\mathsf{IND\text{-}CFA}$ property of our scheme, we first selects randomly two one-time secrets s_1 and s_2, and provide to the client the verification key $\mathsf{vk} = (h^{s_1}, h^{s_2}, s_1 s_2^{-1} r_i)$, where r_i are random elements. Since the tuple does not depend on the secret polynomial, it does not give any useful information to an adversary that try to guess which polynomial among two is used by the server.

Moreover, random elements r_i and the one-time secret s_1 are used to hide the coefficients of $f(\cdot)$ by defining $\alpha_i = (a_i + r_i)s_1$. To prove to the client that the evaluation is correct.

This leads us to the formal definition of our scheme $\mathsf{VIP-POPE}$.

5.3.2 Construction of $\mathsf{VIP-POPE}$

We give the formal definition of our scheme $\mathsf{VIP-POPE}$.

Definition 24. *Let* $\mathsf{VIP-POPE} = (\mathsf{setup}, \mathsf{init}, \mathsf{keyGen}, \mathsf{queryGen}, \mathsf{queryDec}, \mathsf{compute}, \mathsf{decrypt}, \mathsf{verif})$ *be a VPOPEscheme defined by:*

- $\mathsf{setup}(\eta)$: *Using the security parameter η, this algorithm first generates a prime number q. It selects a multiplicative group G of order q and generated by h. It picks $(s_1, s_2) \leftarrow (\mathbb{Z}_q^\star)^2$ and sets* $\mathsf{pub} = (h^{s_1}, h^{s_2}, h, q)$, $\mathsf{sec} = (s_1, s_2)$, *and $F = \mathbb{Z}_q$.*

Finally, it outputs pub*, F, and* sec*.*

- init(F, f, sec) : *We set $f(x) = \sum_{i=0}^{i=k} a_i \cdot x^i$ where $a_i \in \mathbb{Z}_q$. For all $i \in \{0, \ldots, k\}$, it picks $r_i \in \mathbb{Z}_q^\star$ and computes $\alpha_i = (a_i + r_i) \cdot s_1$ and $\gamma_i = s_1 \cdot s_2^{-1} \cdot r_i$. Finally, it sets* $\mathsf{vk} = \{\gamma_i\}_{i=0}^k$, $\mathsf{sk} = \{\alpha_i\}_{i=0}^k$*, and returns* $(\mathsf{vk}, \mathsf{sk})$*.*

- keyGen(η, pub, k) : *For a client c, it picks two primes p_c and q_c such that $(k+1)q^2 < p_c q_c$ and $p_c \approx q_c$. It sets $n_c = p_c q_c$. According to n_c, it generates a Paillier key pair such that $\mathsf{pk}_c = (n_c, g_c)$ and $\mathsf{sk}_c = (\lambda_c, \mu_c)$ as described in Section 2.2.1. It outputs $(\mathsf{pk}_c, \mathsf{sk}_c)$.*

- queryGen(pk_c, x) : *Using x and the Paillier public key pk_c, this algorithm computes, for all $i \in \{1, \ldots, k\}$, $t_i = \mathcal{E}_{\mathsf{pk}}(x^i)$ and returns the encrypted query $t = (\mathsf{pk}_c, \{t_i\}_{i=1}^k)$ along with a proof π_t of equality of plaintexts using* $\mathsf{proof}^{\mathsf{PaillierEq}}$*.*

- queryDec(sk_c, t) : *First this algorithm parses t as $(\mathsf{pk}_c, \{t_i\}_{i=1}^k)$. Using the Paillier secret key sk_c, this algorithm sets $x = \mathcal{D}_{\mathsf{sk}_c}(t_1)$. If $\mathcal{D}_{\mathsf{sk}_c}(t_i) = x^i$ for $2 \leq i \leq k$, it outputs x, $\perp$ otherwise.*

- compute$(t, \pi_t, f, \mathsf{sk}, F)$: *If π_t is accepted by* $\mathsf{verify}^{\mathsf{PaillierEq}}$*, this algorithm uses $\{t_i\}_{i=1}^k$ from t, coefficients $\{a_i\}_{i=0}^k$ of the polynomial function $f(\cdot)$, and $\{\alpha_i\}_{i=0}^k$ from the server secret key* sk *to compute:*

$$d = \mathcal{E}_{\mathsf{pk}_c}(a_0) \cdot \prod_{i=1}^{i=k} t_i^{a_i} \quad \textit{and} \quad \pi_d = \mathcal{E}_{\mathsf{pk}_c}(\alpha_0) \cdot \prod_{i=1}^{i=k} t_i^{\alpha_i} \ ,$$

  and returns (d, π_d), else it returns $\perp$.

- decrypt(sk_c, d) : *Using the Paillier secret key sk_c which is equal to (λ_c, μ_c), this algorithm returns $y = \mathcal{D}_{\mathsf{sk}_c}(d)$ mod q.*

- verif$(x, \mathsf{sk}_c, \mathsf{pub}, y, \pi_d, \mathsf{vk})$: *Using x, sk_c,* vk*, and the proof π_d, this algorithm computes:*

$$y' = \mathcal{D}_{\mathsf{sk}_c}(\pi_d) \bmod q \quad \textit{and} \quad z = \sum_{i=0}^{i=k} \gamma_i \cdot x^i \ .$$

  If $(h^{s_1})^y \cdot (h^{s_2})^z = h^{y'}$, then the algorithm returns 1, else it returns 0.

We illustrate $\mathsf{VIP-POPE}$ scheme in Fig. 5.6.

Parameter Selection. First, consider the additive group $F = \mathbb{Z}_q$ of order q. The size of the prime q must be at least 1024 bits to make the discrete logarithm problem hard in the group G. We recall that the polynomial $f(\cdot)$ is equal to $\sum_{i=0}^{i=k} a_i \cdot x^i$ where $a_i \in \mathbb{Z}_q$ for all $i \in \{0, \ldots, k\}$. Hence, all evaluations are in $\mathbb{Z}_q$; thus we assume that for all $i \in \{0, \ldots, k\}$, we have $0 \leq x^i < q$, and that $0 \leq f(x) < q$. Moreover, the client encrypts for all $i \in \{1, \ldots, k\}$ the value x^i. The evaluation performed by the server is done over encrypted values, i.e., $\mathcal{E}_{\mathsf{pk}_c}(a_0) \cdot \prod_{i=1}^{i=k} \mathcal{E}_{\mathsf{pk}_c}(x^i)^{a_i} = \mathcal{E}_{\mathsf{pk}_c}(a_0 + a_1 \cdot x + \cdots + a_k \cdot x^k)$; then, we need to have $\sum_{i=0}^{i=k} a_i \cdot x^i < n_c = p_c \cdot q_c$ for successful decryption due to Paillier cryptosystem properties, where $\mathbb{Z}_{n_c}$ is the plaintext space of Paillier cryptosystem, p_c and q_c are two prime numbers. Since $0 \leq a_i < q$ and $0 \leq x^i < q$, we have $a_i \cdot x^i < q^2$ for each $i \in \{0, \ldots, k\}$ that gives us $a_0 + a_1 \cdot x + \cdots + a_k \cdot x^k < (k+1) \cdot q^2$. Hence, we need to have $(k+1) \cdot q^2 < n_c$ to always have successful decryption. Moreover, we recommend the size of each prime p_c and q_c to be at least 1024 bits to make the factorization of n_c hard. We show the completeness of our VPOPE scheme, $\mathsf{VIP-POPE}$. In other words, if the server returns a correct encrypted evaluation of $f(\cdot)$ using the encrypted data t given by the client, then the equation's verification must be satisfied. In this case, we consider an honest server, then we have $y = \sum_{i=0}^{k} a_i x^i$, and $y' = \sum_{i=0}^{k} \alpha_i x^i$. Therefore, the equation 5.1 holds:

$$
\begin{aligned}
\prod_{i=0}^{i=k} t_1^{t_2^i a_i} &= \prod_{i=0}^{i=k} \mathcal{E}_{\mathsf{pk}_c}(\alpha)^{\alpha^i x^i a_i} = \prod_{i=0}^{i=k} \mathcal{E}_{\mathsf{pk}_c}(\alpha^{i+1} a_i \cdot x^i) \\
&= \prod_{i=0}^{i=k} \mathcal{E}_{\mathsf{pk}_c}(\alpha a_i \cdot x^i) \\
&= \mathcal{E}_{\mathsf{pk}_c}\left(\alpha \sum_{i=0}^{i=k} a_i \cdot x_i\right) = \mathcal{E}_{\mathsf{pk}_c}(\alpha f(x))
\end{aligned}
\tag{5.1}
$$

In the same way, we have equation 5.2:

$$\begin{aligned}\prod_{i=0}^{i=k} t_1^{t_2^i \alpha_i} &= \prod_{i=0}^{i=k} \mathcal{E}_{\mathsf{pk}_c}(\alpha)^{\alpha^i x^i \alpha_i} = \prod_{i=0}^{i=k} \mathcal{E}_{\mathsf{pk}_c}(\alpha^{i+1}\alpha_i \cdot x^i) \\ &= \prod_{i=0}^{i=k} \mathcal{E}_{\mathsf{pk}_c}(\alpha\alpha_i \cdot x^i) \\ &= \mathcal{E}_{\mathsf{pk}_c}\left(\alpha \sum_{i=0}^{i=k} \alpha_i \cdot x_i\right)\end{aligned} \tag{5.2}$$

Since $y = \mathcal{D}_{\mathsf{sk}_c}(d) \bmod q = \mathcal{D}_{\mathsf{sk}_c}(\mathcal{E}_{\mathsf{pk}_c}(\alpha f(x))) \bmod q = f(x) \bmod q = \sum_{i=0}^{i=k} a_i \cdot x^i \bmod q$, therefore, the equation 5.3 holds:

$$\begin{aligned}&(h_c^{s_1})^y \cdot (h_c^{s_2})^{\sum_{i=0}^{i=k} \gamma_i \cdot x^i} \\ &\qquad = h_c^{s_1 \cdot \sum_{i=0}^{i=k} a_i \cdot x^i + s_2 \cdot \sum_{i=0}^{i=k} s_1 \cdot s_2^{-1} \cdot r_i \cdot x^i} \\ &\qquad = h_c^{s_1 \cdot \sum_{i=0}^{i=k} a_i \cdot x^i + s_1 \cdot \sum_{i=0}^{i=k} r_i \cdot x^i} \\ &\qquad = h_c^{\sum_{i=0}^{i=k} (a_i + r_i) \cdot s_1 \cdot x^i} \\ &\qquad = h_c^{\sum_{i=0}^{i=k} \alpha_i \cdot x^i} \\ &\qquad = h_c^{y'}\end{aligned} \tag{5.3}$$

5.3.3 Security Analysis

We first prove that our VPOPE scheme, $\mathsf{VIP-POPE}$, is secure in our security model. Then we expose some comparisons with PPE schemes of the literature [4, 23].

We present the security proofs of $\mathsf{VIP-POPE}$ in our security model.

Theorem 10. $\mathsf{VIP-POPE}$ *is a CPI-secure VPOPE scheme under the DCR assumption.*

Proof. We assume that there exists $\mathcal{A} \in \text{POLY}(\eta)^2$ such that $\mathsf{Adv}^{\mathsf{CPI}}_{\mathsf{VIP-POPE},\mathcal{A}}(\eta)$ is non-negligible and we show that there exists an algorithm $\mathcal{B} \in \text{POLY}(\eta)$ such that $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathrm{Paillier},\mathcal{B}}(\eta)$ is non-negligible. We build $\mathcal{B}$ as follows:

- $\mathcal{B}$ receives pk from $\mathsf{setup}(\eta)$ and runs $(x_0, x_1, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pk}, \mathsf{pub})$.

- $\mathcal{B}$ runs the oracle $\mathcal{E}_{\mathsf{pk}}(\mathsf{LR}_b(\cdot,\cdot))$ on (x_0^i, x_1^i) for $i \in \{0,\dots,k\}$ and obtains $t = \{c_i\}_{i=0}^k$, Paillier ciphertexts of x_b^i.

- $\mathcal{B}$ runs $b_* \leftarrow \mathcal{A}_2(t, \mathsf{pk}, \mathsf{pub}, \mathsf{sk}, \mathsf{vk}, f, \mathsf{st})$. To simulate the oracle $\mathsf{CO}_{\mathsf{CPI}}(\cdot)$ on x to $\mathcal{A}$, $\mathcal{B}$ computes $t = \{\mathcal{E}_{pk}(x^i)\}_{i=0}^k$.

- Finally, $\mathcal{B}$ outputs b_*.

We remark that:

1. The experiment CPI is perfectly simulated for $\mathcal{A}$.

2. $\mathcal{B}$ wins the $\mathsf{IND\text{-}CPA}$ experiment if and only if $\mathcal{A}$ wins the CPI experiment.

Since $\mathsf{Adv}^{\mathsf{CPI}}_{\mathsf{VIP-POPE},\mathcal{A}}(\eta)$ is non-negligible, then $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\text{Paillier},\mathcal{B}}(\eta)$ is non-negligible. However, Paillier cryptosystem is $\mathsf{IND\text{-}CPA}$ under the DCR assumption, then $\mathcal{B}$ can be used to break the DCR assumption, which contradict our hypothesis and conclude the proof. $\square$

Theorem 11. *For any $k \in \mathbb{N}$, $\mathsf{VIP-POPE}$ achieves k-EnPP property.*

Proof. Let $\mathcal{A} \in \text{POLY}(\eta)$ be an adversary. Since $\mathcal{A}$ can query the oracle $\mathsf{CO}_{\mathsf{EnPP}}(\cdot)$ at most k times, $\mathcal{A}$ can obtains at most k points $\{(x_i, y_i)\}_{i=1}^k$ such that $y_i = f(x_i)$. Without loss of generality, we can assume that the aim of $\mathcal{A}$ is to compute $f(x_{k+1})$ for some value x_{k+1} of her choice, such that $x_{k+1} \neq x_i$ for $1 \leq i \leq k$. We show that $\mathsf{Adv}^{\mathsf{EnPP}}_{\mathsf{VIP-POPE},\mathcal{A}}(\eta)$ is negligible.

Since the degree of the secret polynomial $f(x)$ is equal to k, there are $k+1$ unknown coefficients. Obtaining $\{(x_i, f(x_i))\}_{i=1}^k$ such that $x_i \neq x_j$ if $i \neq j$ only give k equations with $k+1$ unknown coefficients. Hence, none of these coefficients can be derived. In particular, the constant coefficient $a_0 = f(0)$ is in the uniform distribution of $\mathbb{Z}_q^\star$. Using the Lagrange interpolation, the value $f(x_{k+1})$ can be described as in equation 5.4:

$$f(x_{k+1}) = f(0)\prod_{j=1}^{k}\frac{x_{k+1}}{x_j} + \sum_{i=1}^{k} f(x_i)\prod_{x}\frac{x_{k+1}-x_i}{x_j-x_i} \tag{5.4}$$

We set $a = \prod_{j=1}^{k} \frac{x_{k+1}}{x_j}$, and $b = \sum_{i=1}^{k} f(x_i) \prod_{x}^{k} \frac{x_{k+1}-x_i}{x_j-x_i}$ and we have the equation 5.5:

$$f(x_{k+1}) = a \cdot a_0 + b \tag{5.5}$$

which is an affine cipher. Since a and b are in $\mathbb{Z}_q^\star$ and a_0 is in the uniform distribution of $\mathbb{Z}_q^\star$, then $f(x_{k+1})$ is in the uniform distribution of $\mathbb{Z}_q^\star$. Hence, we have:

$$\mathsf{Adv}^{\mathsf{EnPP}}_{\mathsf{VIP-POPE},\mathcal{A}}(\eta) = 1/q ,$$

which is negligible with respect to the security parameter η. $\square$

Theorem 12. *For any $k \in \mathbb{N}$, $\mathsf{VIP-POPE}$ is a k-IND-CFA-secure VPOPE scheme.*

Proof. Let $\mathcal{A} \in \text{POLY}(\eta)$ be an algorithm. We show that there exists an algorithm $\mathcal{B} \in \text{POLY}(\eta)$ simulating the experiment $\mathsf{Exp}^{k\text{-IND-CFA}}_{\mathsf{VIP-POPE},\mathcal{A}}(\eta)$. We build $\mathcal{B}$ as follows:

- $\mathcal{B}$ picks $b \xleftarrow{\$} \{0,1\}$.
- $\mathcal{B}$ generates $((\mathsf{pk},\mathsf{sk}), h, \mathbb{Z}_q^\star) \leftarrow \mathsf{setup}(\eta)$.
- $\mathcal{B}$ runs $(f_0, f_1, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathbb{Z}_q^\star, k)$. It sets $f_0(x) = \sum_{i=0}^{k} a_{0,i} x^i$ and $f_1(x) = \sum_{i=0}^{k} a_{1,i} x^i$.
- $\mathcal{B}$ picks $(s_1, s_2, r) \xleftarrow{\$} (\mathbb{Z}_q^\star)^3$. For all $i \in \{0, \ldots, k\}$, it picks $r_i \xleftarrow{\$} \mathbb{Z}_q^\star$, and sets $a_i = a_{b,i} + r \cdot (a_{0,i} - a_{1,i})$, $\alpha_i = (a_i + r_i)s_1$, and $\gamma_i = s_1 s_2^{-1} r_i$. It sets $f(x) = \sum_{i=0}^{k} a_i x^i$, $f'(x) = \sum_{i=0}^{k} \alpha_i x^i$, $Z(x) = \sum_{i=0}^{k} \gamma_i x^i$, and returns $\mathsf{vk} = (h^{s_1}, h^{s_2}, \{\gamma_i\}_{i=0}^{k})$.
- $\mathcal{B}$ runs $b_* \leftarrow \mathcal{A}_2((\mathsf{pk},\mathsf{sk}), h, \mathbb{Z}_q^\star, \mathsf{vk}, k, \mathsf{st})$. To simulate the oracle $\mathsf{CO}_{\mathsf{CFA}}(\cdot)$ to $\mathcal{A}$ on $t_j = \{\mathcal{E}_{\mathsf{pk}}(x_j^i)\}_{i=1}^{k}$ (for $1 \leq j \leq k$), the Paillier encryption of x^i for $1 \leq i \leq k$, $\mathcal{B}$ computes:

  $$d_j = \prod_{i=0}^{k} \mathcal{E}_{\mathsf{pk}}(x_j^i)^{a_i} \quad \text{and} \quad \pi_j = \prod_{i=0}^{k} \mathcal{E}_{\mathsf{pk}}(x_j^i)^{\alpha_i} ,$$

  and returns (d_j, π_j).
- Finally, $\mathcal{B}$ outputs b_*.

We remark that since s_1, s_2, and r_i (for $0 \leq i \leq k$) are chosen in the uniform distribution of $\mathbb{Z}_q^\star$, then each element of vk comes from the uniform distribution on $\mathbb{Z}_q^\star$. Moreover, if $f_0(x) - f_1(x) \neq 0$, the evaluation is in the uniform distribution of $\mathbb{Z}_q^\star$ for all $b \in \{0,1\}$ since r comes from the uniform distribution of $\mathbb{Z}_q^\star$. We have:

$$\begin{aligned}
(h^{s_1})^{f(x)}(h^{s_2})^{Z(x)} &= h^{s_1 \sum_{i=0}^{k} a_i x^i + s_2 \sum_{i=0}^{k} \gamma_i x^i} \\
&= h^{s_1 \sum_{i=0}^{k} a_i x^i + s_2 \sum_{i=0}^{k} s_1 s_2^{-1} r_i x^i} \\
&= h^{s_1 \sum_{i=0}^{k} a_i x^i + s_1 \sum_{i=0}^{k} r_i x^i} \\
&= h^{\sum_{i=0}^{k} (a_i + r_i) s_1 x^i} \\
&= h^{\sum_{i=0}^{k} \alpha_i x^i} \\
&= h^{f'(x)}
\end{aligned}$$

We deduce that the experiment k-$\mathsf{IND\text{-}CFA}$ is perfectly simulated for $\mathcal{A}$. Then $\mathcal{A}$ cannot do better than the random to guess the value of the chosen b. Hence, we have:

$$\Pr[1 \leftarrow \mathsf{Exp}^{k\text{-}\mathsf{IND\text{-}CFA}}_{\mathsf{VIP-POPE},\mathcal{A}}(\eta)] = 1/2\,,$$

and so $\mathsf{Adv}^{k\text{-}\mathsf{IND\text{-}CFA}}_{\mathsf{VIP-POPE},\mathcal{A}}(\eta)$ is negligible which conclude the proof. $\square$

Theorem 13. *For any $k \in \mathbb{N}$, $\mathsf{VIP} - \mathsf{POPE}$ is UNF-secure under the DL assumption.*

Proof. We assume that there exists $\mathcal{A} \in \textsc{poly}(\eta)^2$ such that $\mathsf{Adv}^{\mathsf{UNF}}_{\mathsf{VIP-POPE},\mathcal{A}}(\eta)$ is non-negligible. We show that $\mathcal{A}$ can be used to construct an algorithm $\mathcal{B}$ that computes $\log_h(h^{s_1})$.

First, we prove by contradiction that if $y_* \neq f(x_*)$, then we also have $y'_* \neq y'$, where $y' = \sum_{i=0}^{k} \alpha_i x_*^i$.

Assume we have $y'_* \neq f(x_*)$ and $y'_* = y'$, then:

$$(h^{s_1})^{y_*}(h^{s_2})^{\sum_{i=0}^{k} \gamma_i x_*^i} = h^{y'_*}\,,$$

$$(h^{s_1})^{f(x_*)}(h^{s_2})^{\sum_{i=0}^{k} \gamma_i x_*^i} = h^{y'}\,,$$

and $(h^{s_1})^{y_*}(h^{s_2})^{\sum_{i=0}^{k} \gamma_i x_*^i} = (h^{s_1})^{f(x_*)}(h^{s_2})^{\sum_{i=0}^{k} \gamma_i x_*^i}$. This contradicts the precondition $y_* \neq f(x_*)$. Hence, the case $y_* \neq f(x_*)$ and $y'_* = y'$ cannot happen.

For similar reasons, the case $y'_* \neq y'$ and $y_* = f(x_*)$ cannot happen neither. Therefore, we must have both inequalities $y_* \neq f(x_*)$ and $y'_* \neq y'$ hold.

Next, we prove that such an adversary $\mathcal{A}$ allows us to compute $\log_h(h^{s_1})$. Since the verification equation $(h^{s_1})^{y_*}(h^{s_2})^{\sum_{i=0}^{k}\gamma_i x_*^i} = h^{y'_*}$ holds, we have:

$$\begin{cases} (h^{s_1})^{y_*}(h^{s_2})^{\sum_{i=0}^{k}\gamma_i x_*^i} = h^{y'_*} \\ (h^{s_1})^{f(x_*)}(h^{s_2})^{\sum_{i=0}^{k}\gamma_i x_*^i} = h^{y'} \end{cases}$$

$$\Rightarrow (h^{s_1})^{(y_*+\sum_{i=0}^{k}\gamma_i x_*^i)-(f(x_*)+\sum_{i=0}^{k}\gamma_i x_*^i)} = h^{y'_*-y'}$$

$$\Leftrightarrow h^{s_1} = h^{\frac{y'_*-y'}{y_*-f(x_*)}} .$$

Hence, we have $log_g(h^{s_1}) = \frac{y'_*-y'}{y_*-f(x_*)}$. Since we have proved that $y_* \neq f(x_*)$, the discrete logarithm $\log_h(h^{s_1})$ can computed with the same probability as $\mathcal{A}$ wins the UNF experiment. Therefore, based on the DL assumption, there cannot exist an adversary $\mathcal{A}$ such that $\mathsf{Adv}^{\mathsf{UNF}}_{\mathsf{VIP-POPE},\mathcal{A}}(\eta)$ is non-negligible. □

5.3.4 Experimental Results

Schemes	Setup size	Key size	Verif. cost	Pairing	Assumption	Model	Privacy
PolyCommit$_{\mathsf{Ped}}$ [14]	$\mathcal{O}(k)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	Pairing based	t-SDH	Standard	No
PIPE [4]	$\mathcal{O}(1)$	$\mathcal{O}(k)$	$\mathcal{O}(k \cdot \log(q))$	Paring free	DDH	ROM	No
Xia's scheme [23]	$\mathcal{O}(1)$	$\mathcal{O}(k)$	$\mathcal{O}(k \cdot \log(q))$	Pairing free	DL	Standard	No
VIP − POPE	$\mathcal{O}(1)$	$\mathcal{O}(k)$	$\mathcal{O}(3 \cdot \log(q) + k) + D$	Pairing free	DL/DCR	Standard	Yes

Table 5.1: Comparison of VIP − POPE with other PPE schemes.
**We denote by D the constant cost of one Paillier decryption.

In Table 5.1, we provide comparison of our scheme with PolyCommit$_{\mathsf{Ped}}$ [14], PIPE [4] and Xia's scheme [23]. We observe that the verification key size and verification cost are constant in PolyCommit$_{\mathsf{Ped}}$ while in all other schemes it depends on the degree k. The verification equation in PolyCommit$_{\mathsf{Ped}}$ involves several bilinear pairing which is costly compared to other operations. The verification key size and verification cost are not constant in our scheme but our scheme is pairing free and efficient as compared to other pairing free schemes. Moreover, our scheme VIP − POPE provides client's data privacy while other three schemes do not provide any privacy. To support our claim about efficiency, we implement all these schemes under same environment and with same parameters. We test these schemes for different values of degrees with realistic parameters. In our

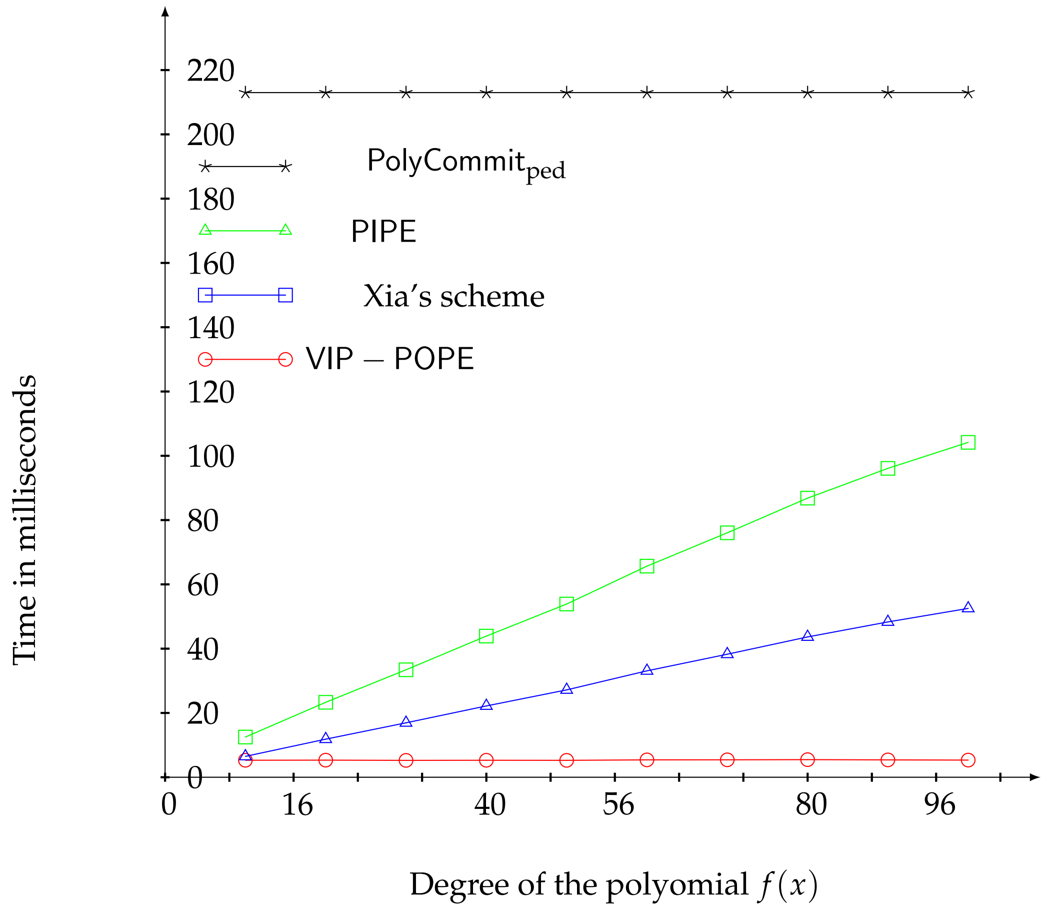


Figure 5.7: Verification cost comparison.

scheme, the verification of the result obtained from the server is done by a client. In such a case, the verification cost becomes important aspect of the scheme. We claim that our scheme is most efficient so far in terms of verification cost. To support our claim, we implement VIP − POPE, PIPE and Xia's scheme in SageMath 8.1 on 64-bit PC with Intel Core i5 - 6500 CPU @ 3.2 GHz and 4 GiB RAM. The new scheme, VIP − POPE, provides privacy of client's data while the other two schemes, PIPE and Xia's scheme, do not provide privacy of client's data. To keep the comparison as fair as possible, we implement all three schemes with realistic and same parameters. We compare the cost of only verification equation in all the three schemes. For our scheme, we choose a 1024 bit prime q and 160 bit prime q_1 such that $q' = 2q_1q + 1$ is a prime. We choose another 1024 bit prime p and set $n = pq'$. The coefficients of the polynomial $f(x)$, the secret values (s_1, s_2) and $\{r_i\}_{i=0}^{k}$ are all selected uniformly at random from $\mathbb{Z}_q^\star$. For Xia's scheme and PIPE, we keep the value of q, the polynomial $f(x)$ and $\{r_i\}_{i=0}^{k}$ same as in $\mathsf{VIP-POPE}$.

For different values of the degree of the polynomial $f(x)$, we ran each scheme for 100 new instances and each instance for 10 times. We then averaged out the total time for the verification equation in each scheme. In Fig. 5.7, we observe that $\mathsf{VIP-POPE}$ takes almost constant time while the cost of verification equation in PIPE and Xia's scheme increases linearly with respect to the degree k. Moreover, our scheme takes only around $5 - 6$ milliseconds for verification equation even for $k = 100$ which makes it practically feasible for real applications.

5.4 conclusion

In this chapter, we gave a formal definition of another primitive called VPOPE (for Verifiable and Private Oblivious Polynomial Evaluation). This primitive allows a company to delegate the computation of a secret polynomial $f(\cdot)$ to an external server on the client's encrypted data in a verifiable way. In other terms, a client sends an encrypted query to a server associated with her secret data x using her public key pk. Then, the client receives d with proof that $d = \mathcal{E}_{\mathsf{pk}}(f(x))$. We defined the required security properties for VPOPE: Client's Privacy - Indistinguishability (CPI), Indistinguishability against Chosen Function Attack (IND-CFA), Query Soundness (QS), and Unforgeability. The query soundness notion ensures that a user cannot send a malicious query to learn more about the polynomial. The CPI notion ensures that the server does not learn anything about the secret data x.

In PIPE, the user sends data to the server in plain form and the server computation over plain data. For the delegation of computation over encrypted data, we designed a scheme called Verifiable $\mathsf{IND\text{-}CFA}$ Paillier based Private Oblivious Polynomial Evaluation ($\mathsf{VIP} - \mathsf{POPE}$). Using the security properties of Private Polynomial Evaluation (PPE) schemes and Oblivious Polynomial Evaluation (OPE) schemes, we prove that our scheme is *proof unforgeability*, *indistinguishability against chosen function attack*, and *client privacy*-secure under the Decisional Composite Residuosity assumption in the random oracle model.

CHAPTER 6

Privacy-Preserving Verifiable Computation

6.1 Introduction

The $\mathsf{VIP-POPE}$ scheme considers privacy of the user's data but doesn't consider privacy of user's identity during identity verification. Based on the application, a system can use any privacy preserving identity verification scheme along with $\mathsf{VIP-POPE}$ to have data privacy as well as user's privacy. Alternatively, it is good to have a single scheme which provides both the features. We propose a design for verifiable polynomial computation scheme along with privacy preserving authentication. In the proposed system model, we assume that the cloud service provider is a semi-trusted party. We trust the cloud for secrecy of the polynomial, but we do not trust the server for computation on polynomial function. The system model consists of the following entities.

- Cloud Service Provider (CSP): CSP verifies authenticity of message and evaluates a function $f(x)$ over a user input m. The CSP also computes a proof of computation for $f(m)$.
- Users: Users are consumers of the company who get services from CSP on their requests.
- Company: The company provide appropriate service in terms of evaluation of a function $f(x)$ with the help of CSP. The company also generates and securely distributes public and private parameters to all involved entities.

The PriVC scheme has following goals - (i) verifiable computation on encrypted data; (ii) keeping the computation logic on data hidden from the user (secrecy of the prediction function); (iii) not letting the server know for whom the computation is intended (privacy of the user); and (iv) not denying of usage of the services by the user (undeniability). Once a user sends a message to the CSP, the user cannot deny the transaction and moreover, the CSP is able to prove that the transaction was indeed done by that particular user. The PriVC scheme works as follows. The company possesses a prediction function $f(x)$ and it hires CSP for computational power. The $f(x)$ is a confidential polynomial function and known to the company and CSP only. A user first gets registered with the company and receives a pseudo-identity. To avail the service, the user sends encryption of data m and the proof of authenticity to the CSP. To encrypt the data, we use a symmetric version of the extended DGHV scheme [58]. The CSP verifies the authenticity of the data and then generates an encrypted form of $f(m)$ along with the proof of computation. The user decrypts and verifies the computation of $f(m)$. Furthermore, a user cannot deny a valid transaction and at the same time, the CSP cannot generate a valid transaction for any user. The system model of PriVC is depicted in Figure 6.1.

Any private polynomial evaluation scheme should be designed in such a way that the proof of computation and other public parameters must not give any advantage to an attacker. In our proposed PriVC scheme, we use the security notion of indistinguishability against chosen function attack (IND-CFA) and UNF as defined in chapter 4. We first define the PriVC scheme as follows.

Definition 25 (Privacy-Preserving Verifiable Computation)**.** *Let f be a polynomial in $\mathbb{Z}_q^+[X]$. A Privacy - Preserving Verifiable Computation* (PriVC) *is a 8-tuple of algorithms* (Setup, Init, KeyGen, EvalRequest, VerifyAuth, Eval, VerifyResult, VerifyTrans) *defined as:*

- $\mathsf{Setup}(\lambda)$*: It takes security parameter λ as input and the public parameters* pub*.*
- $\mathsf{Init}(f, \mathsf{pub})$*: It take polynomial function f and public parameters* pub *as input and outputs a secret key for the function,* $\mathsf{sk_f}$*, and verification key for individual user u,* $\mathsf{vk_f^u}$*.*
- $\mathsf{KeyGen}(\lambda, \mathsf{pub})$*: For a user u, it generates extended DGHV private key* $\mathsf{k_u}$ *and parameters ρ, γ.*

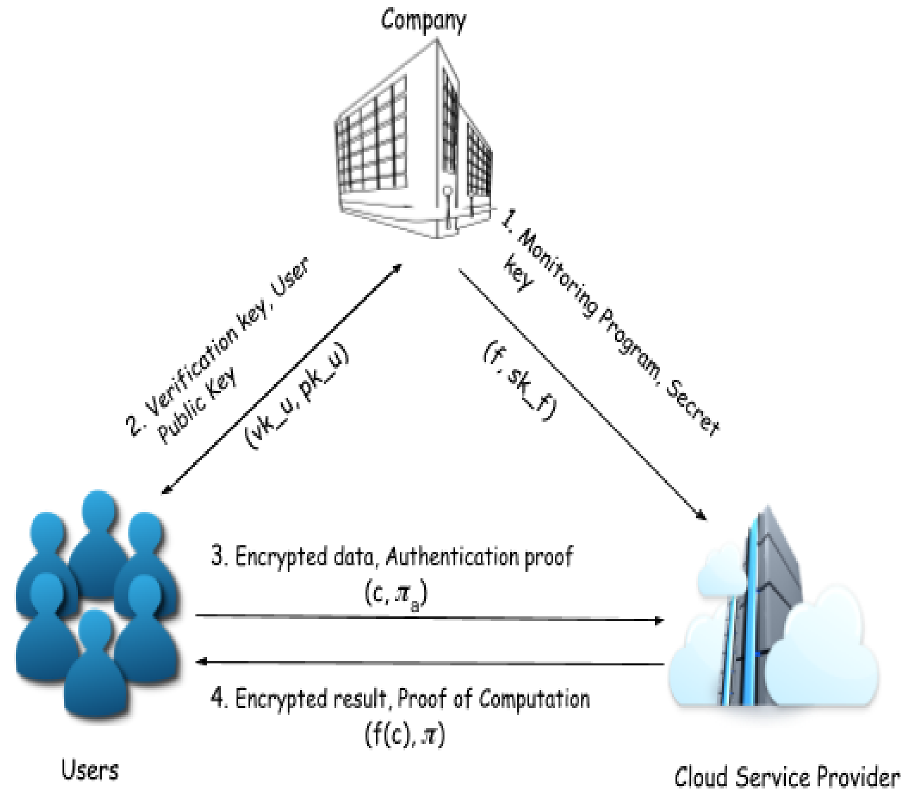

Company
1. Monitoring Program, Secret key
(f, sk_f)
2. Verification key, User Public Key
(vk_u, pk_u)
3. Encrypted data, Authentication proof
(c, π_a)
4. Encrypted result, Proof of Computation
(f(c), π)
Users
Cloud Service Provider

Figure 6.1: PriVC scheme for monitoring system

- $\mathsf{EvalRequest}(m, \mathsf{k_u}, \mathsf{pub}, \mathsf{vk_f^u})$: *It takes a message m, user key k_u, public parameters* pub*, verification key* $\mathsf{vk_f^u}$ *as input and outputs an encrypted query (c, π_a).*

- $\mathsf{VerifyAuth}(c, \pi_a, \mathsf{sk_f}, f, \mathsf{pub})$: *It takes an encrypted query (c, π_a), secret key* $\mathsf{sk_f}$*, the polynomial f and public parameters* pub *as input. It outputs c if (c, π_a) is accepted; else, aborts.*

- $\mathsf{Eval}(c, f, \mathsf{sk_f}, \mathsf{pub})$: *It takes an encrypted message c, secret key* $\mathsf{sk_f}$ *and public parameters* pub *as input. It outputs (y, π) where $y = f(c)$ is an encryption of $f(m)$ and π is* proof of computation *for $f(m)$.*

- $\mathsf{VerifyResult}(y, \pi, \mathsf{vk_f^u}, \mathsf{k_u}, m, \mathsf{pub})$: *It decrypts (y, π) and verifies the computation of $f(m)$ using verification key* $\mathsf{vk_f^u}$.

- $\mathsf{VerifyTrans}(c, \pi_a, \mathsf{k_u}, \mathsf{vk_f^u}, \mathsf{pub})$: *It takes a transaction (c, π_a), user key k_u, verification key* $\mathsf{vk_f^u}$ *and public parameters* pub *as input. It outputs* 1 *if (c, π_a) is a valid transaction otherwise* 0.

6.1.1 Adversarial Assumptions

The polynomial $f(x)$ used in the PriVC scheme is a secret polynomial and users should not be able to learn anything about $f(x)$.

An adversary (we note that a legitimate user can also act as adversary) can choose two polynomials and then tries to guess which polynomial is used by the CSP. More concretely, given two polynomial f_0 and f_1 , the adversary should not be able to distinguish which polynomial is used by the CSP. We prove that the PriVC scheme is secure under IND-CFA model.

Definition 26 (Oracle for CFA, $\mathcal{O}_{\mathsf{CFA}}$)**.** *In this oracle, the adversary has an access to* Eval *algorithm. The adversary can query the server at most one time. The input of each query is an encrypted value c of data m and output of each query is (y, π) where $y = f_b(c) + r * (f_0(c) - f_1(c))$, r is a random integer and π is* proof *of computation. The oracle returns (y, π).*

```
$\mathcal{O}_{\mathsf{CFA}}$ $(c, b, f_0, f_1, \mathsf{sk}_f, \mathsf{pub})$:
    $r_1, r_2 \leftarrow \{0,1\}^*$:
    $f' \leftarrow f_b + r_1 * (f_0 - f_1)$:
    $\mathsf{sk}'_f \leftarrow \mathsf{sk}_f + r_2 * (f_0 - f_1)$:
    $(y, \pi) \leftarrow \mathsf{Eval}(c, f', \mathsf{sk}'_f, \mathsf{pub})$;
    Return $(y, \pi)$.
```

In IND-CFA model, the adversary tries to guess which polynomial is used by a PriVC scheme. The adversary chooses two polynomials (f_0, f_1). The server randomly selects one polynomial f_b where $b \in \{0, 1\}$ and random integers r_1, r_2. It sets $f' = f_b + r_1 * (f_0 - f_1)$, $sk' = sk + r_2 * (f_0 - f_1)$ and evaluates and proves the computation of $y = f'(c)$, where c is given by the adversary. Note that if $f_0(m) = f_1(m)$, then decryption of y gives $f_b(m)$; else, the adversary gets garbage value after decryption. In the definition of IND-CFA, $\mathbb{Z}_q^+[X]_n$ represents a set of all polynomials of degree at most n with coefficients in $\mathbb{Z}_q^+$.

Definition 27 (IND-CFA)**.** *Let Π be a* PriVC*, $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a two-party PPT adversary and n be an integer. The n-IND-CFA experiment for $\mathcal{A}$ against Π is defined as* *follows:*

```
$\mathsf{Exp}^{n\text{-}IND\text{-}CFA}_{\Pi,\mathcal{A}}(\lambda)$:
    $b \leftarrow \{0,1\}$;
    $\mathsf{pub} \leftarrow \mathsf{Setup}(\lambda)$;
    $(f_0, f_1, st) \leftarrow \mathcal{A}_1(\mathsf{pub}, n)$;
    $(\mathsf{sk}_f, \mathsf{vk}^u_f) \leftarrow \mathsf{Init}(f_b, \mathsf{pub})$;
    $\mathsf{k}_u \leftarrow \mathsf{KeyGen}(\lambda, \mathsf{pub})$;
    $(c, \pi_a) \leftarrow \mathsf{EvalRequest}(st, \mathsf{k}_u, \mathsf{pub}, \mathsf{vk}^u_f)$
    $b_* \leftarrow \mathcal{A}_2^{\mathcal{O}_{CFA}(\cdot)}(\mathsf{k}_u, \mathsf{vk}^u_f, \mathsf{pub}, f_0, f_1, c)$;
    If $f_0 \notin \mathbb{Z}_q^+[X]_n$ or $f_1 \notin \mathbb{Z}_q^+[X]_n$:
        Then return $\perp$;
    Else return $(b = b_*)$.
```

$\mathcal{A}$ has access to the server oracle $\mathcal{O}_{CFA}(\cdot)$. The advantage of the adversary $\mathcal{A}$ against the n-IND-CFA experiment is defined by:

$$\mathsf{Adv}^{n\text{-}IND\text{-}CFA}_{\Pi,\mathcal{A}}(\lambda) = \left| \frac{1}{2} - \Pr\left[1 \leftarrow \mathsf{Exp}^{n\text{-}IND\text{-}CFA}_{\Pi,\mathcal{A}}(\lambda)\right] \right|.$$

A scheme Π is n-IND-CFA secure if this advantage is negligible in λ for any polynomial-time adversary $\mathcal{A}$.

We note that in the above experiment, n is the degree of the polynomial $f(x)$ and st is a testing point. Using public parameters, the adversary chooses two functions f_0, f_1 and a point st.

The second security model is regarding unforgeability property. In verifiable computation scheme, it is essential to prove the unforgeability. A PriVC scheme is said to be unforgeable when the server cannot produce valid proof for a wrong computation.

Definition 28 (Unforgeability). *Let Π be a* PriVC *scheme and $\mathcal{A} = (\mathcal{A}_1.\mathcal{A}_2)$ be a two-party adversary. The Unforgeability (UNF) experiment for $\mathcal{A}$ against Π is defined as follows:*

```
$\mathsf{Exp}^{UNF}_{\Pi,\mathcal{A}}(\lambda)$:
  $\mathsf{pub} \leftarrow \mathsf{Setup}(\lambda)$;
  $(f, st) \leftarrow \mathcal{A}_1(\mathsf{pub})$;
  $(\mathsf{sk}_f, \mathsf{vk}^u_f) \leftarrow \mathsf{Init}(f, \mathsf{pub})$;
  $\mathsf{k}_u \leftarrow \mathsf{KeyGen}(\lambda, \mathsf{pub})$;
  $(c, \pi_a) \leftarrow \mathsf{EvalRequest}(st, \mathsf{k}_u, \mathsf{pub}, \mathsf{vk}^u_f)$
  $(y, \pi) \leftarrow \mathsf{Eval}(c, f, \mathsf{sk}_f, \mathsf{pub})$
  $(x_*, y_*, \pi_*) \leftarrow \mathcal{A}_2(\mathsf{sk}_f, f, c, y, \pi, \mathsf{pub})$;
  If
          $f(x_*) \neq y_*$ and
          $\mathsf{VerifyResult}(y_*, \pi_*, \mathsf{vk}^u_f, \mathsf{k}_u, x_*, \mathsf{pub})$ :
        Then return 1;
  Else return 0.
```

The advantage of the adversary $\mathcal{A}$ against the UNF experiment is defined by:

$$\mathsf{Adv}^{UNF}_{\Pi,\mathcal{A}}(\lambda) = \Pr\left[1 \leftarrow \mathsf{Exp}^{UNF}_{\Pi,\mathcal{A}}(\lambda)\right].$$

A scheme Π is UNF secure if the above advantage is negligible for any polynomial-time adversary $\mathcal{A}$.

6.2 Construction of the PriVC

The Privacy-Preserving Verifiable Computation (PriVC) is a 8-tuple of algorithms (Setup, Init, KeyGen, EvalRequest, VerifyAuth, Eval, VerifyResult, VerifyTrans) as defined in Section 3 (Definition 6). The detailed working principle of each algorithm of the PriVC is as follows.

- $\mathsf{Setup}(\lambda)$: This algorithm generates a prime q and a multiplicative group G of order q. Let g be a generator of the group G. It randomly selects a secret key $sk_c \in \mathbb{Z}_q^\star$ for the company and sets $pk_c = g^{sk_c}$. It also selects a cryptographically secure hash function $H : \{0,1\}^\star \longmapsto \mathbb{Z}_q$. It sets $\mathsf{pub} = (q, pk_c, G, g, H)$.

- $\mathsf{Init}(f, \mathsf{pub})$: For the polynomial $f \in \mathbb{Z}_q[x]^n$, the company picks $(n+1)$ random numbers $\{\gamma_i \in \mathbb{Z}_q\}_{i=0}^n$ and sets $\Gamma(x) = \sum_{i=0}^n \gamma_i x^i$. For each user u in the system, it picks a secret sk_u randomly from $\mathbb{Z}_q^\star$ and sets $V_u(x) = sk_u^{-1}(\Gamma(x) - sk_c f(x))$. It sends the secret key $\mathsf{sk_f} = \Gamma(x)$ along with f to the CSP and the verification key $\mathsf{vk_f^u} = (V_u(x), pk_u = g^{sk_u})$ to user u.

- $\mathsf{KeyGen}(\lambda, \mathsf{pub})$: This algorithm is run by each user separately to generate extended DGHV private key. This algorithm picks a prime p in $[2^{\eta-1}, 2^\eta]$ where $\eta = \mathcal{O}(\lambda^2)$. It sets parameters $\rho = \mathcal{O}(\lambda)$ and $\gamma = \mathcal{O}(\lambda^5)$. The user's private key is $\mathsf{k}_u = p$.

- $\mathsf{EvalRequest}(m, \mathsf{k}_u, \mathsf{pub}, \mathsf{vk_f^u})$: This algorithm generates an authentic encrypted ciphertext query for the data m. We assume that the function f is a polynomial of degree n with coefficients $0 \le a_i \le q$ for $0 \le i \le n$. This algorithm first computes a tuple $c = (c_0, c_1, \ldots, c_n)$ as follows:

$$c_i = (m^i \bmod q) + s_i q + t_i p$$

where each $s_i \in (-2^\rho, 2^\rho)$ and $t_i \in (0, 2^\gamma/p)$ are randomly chosen integers. It then computes $\mathsf{proof\ of\ authenticity}$ $\pi_a = (R, S)$ as follows:

$$\theta = H(c)$$
$$R = V_u(\theta)$$
$$S = pk_u.$$

The user sends (c, π_a) to the CSP.

- $\mathsf{VerifyAuth}(c, \pi_a, \mathsf{sk_f}, f, \mathsf{pub})$: This algorithm verifies authenticity of c. The CSP computes $f(\theta)$, $\Gamma(\theta)$ where $\theta = H(c)$, and checks validity of the equation 6.1:

$$S^R pk_c^{f(\theta)} \stackrel{?}{=} g^{\Gamma(\theta)} \tag{6.1}$$

  The algorithm outputs (c, π_a) if the above equation holds true; else, aborts.

- $\mathsf{Eval}(c, f, \mathsf{sk_f}, \mathsf{pub})$: This algorithm computes $f(c)$ and $\Gamma(c)$ as follows:

$$f(c) = \sum_{i=0}^{n} a_i c_i\,, \qquad \Gamma(c) = \sum_{i=0}^{n} \gamma_i c_i.$$

  It sends $(y = f(c), \pi = \Gamma(c))$ to the user.

- $\mathsf{VerifyResult}(y, \pi, \mathsf{vk_f^u}, \mathsf{k_u}, m, \mathsf{pub})$: This algorithm computes $f(m)$ and $\Gamma(m)$ from (y, π) as follows:

$$f(m) = (y \bmod p) \bmod q$$

$$\Gamma(m) = (\pi \bmod p) \bmod q.$$

  It then verifies the equation 6.2:

$$pk_u^{V_u(m)} pk_c^{f(m)} \stackrel{?}{=} g^{\Gamma(m)} \tag{6.2}$$

- $\mathsf{VerifyTrans}(c, \pi_a, \mathsf{k_u}, \mathsf{vk_f^u}, \mathsf{pub})$: Using this algorithm, the user can verify whether a specific query (c, π_a) was sent by him/her. From c and $\pi_a = (R, S)$, if $S = pk_u$, then it computes $\theta = H(c)$. The user then verifies the equation 6.3:

$$R \stackrel{?}{=} V_u(\theta) \tag{6.3}$$

Correctness: We provide proof of correctness for equation in VerifyResult.

$$\begin{aligned} pk_u^{V_u(m)} pk_c^{f(m)} &= (g^{sk_u})^{sk_u^{-1}(\Gamma(m)-sk_c f(m))}(g^{sk_c})^{f(m)} \\ &= g^{\Gamma(m)-sk_c f(m)}(g^{sk_c})^{f(m)} \\ &= g^{\Gamma(m)}. \end{aligned}$$

6.3 Security Analysis

We first show that PriVC is IND-CFA secure. Note that for proof generation, we are using only one additional polynomial $\Gamma(x)$ and no other parameter. We show that this additional polynomial $\Gamma(x)$ doesn't leak any additional information about the polynomial f.

6.3.1 IND-CFA security

Theorem 14. $\forall n \in \mathbb{N}$, *PriVC is unconditionally n-IND-CFA secure.*

Proof. Let $\mathcal{A}$ be an IND-CFA adversary for PriVC. We show that there exist a polynomial time algorithm $\mathcal{B}$ which can simulate the experiment $\mathsf{Exp}^{\text{n-IND-CFA}}_{\mathsf{PriVC},\mathcal{A}}(\lambda)$ to $\mathcal{A}$. The algorithm $\mathcal{B}$ works as follows:

- $\mathcal{B}$ picks $b \leftarrow \{0,1\}$.
- $\mathcal{B}$ generates $\mathsf{pub} = (q, pk_c, G, g, H) \leftarrow \mathsf{Setup}(\lambda)$ where $pk_c = g^{sk_c}$ and $sk_c \in \mathbb{Z}_q^\star$.
- $\mathcal{B}$ runs $(f_0, f_1, st) \leftarrow \mathcal{A}_1(\mathsf{pub}, n)$.
- $\mathcal{B}$ generates $(\mathsf{sk}_f, \mathsf{vk}_f^u) \leftarrow \mathsf{Init}(f_b, \mathsf{pub})$ where $mathsfsk_f = \Gamma(x) = \sum_{i=0}^{n} \gamma_i x^i$, $\mathsf{vk}_\mathsf{f}^\mathsf{u} = (V_u(x), pk_u)$, $pk_u = g^{sk_u}$ and $V_u(x) = sk_u^{-1}(\Gamma(x) - sk_c f_b(x))$.
- $\mathcal{B}$ generates $\mathsf{k_u} \leftarrow \mathsf{KeyGen}(\lambda, \mathsf{pub})$ where $\mathsf{k_u} = p$ is extended DGHV encryption key.
- $\mathcal{B}$ generates $(c, \pi_a) \leftarrow \mathsf{EvalRequest}(st, \mathsf{k_u}, \mathsf{pub}, \mathsf{vk}_f^u)$ where $c = (c_0, c_1, ..., c_n)$, c_i is encryption of x^i and $\pi_a = (V_u(H(c)), pk_u)$.

- $\mathcal{B}$ runs $b_\star \leftarrow \mathcal{A}_2^{\mathcal{O}_{CFA}(\cdot)}(\mathsf{k_u}, \mathsf{vk_f^u}, \mathsf{pub}, f_0, f_1, c)$. To simulate the oracle $\mathcal{O}_{CFA}(\cdot)$ on c to $\mathcal{A}$ on c, $\mathcal{B}$ picks $r_1, r_2 \leftarrow \{0,1\}^\star$ and set $f' = f_b + r_1(f_0 - f_1)$ and $sk'_f = \Gamma'(x) = sk_f + r_2(f_0 - f_1)$ and computes $y = f'(c)$ and $\pi = \Gamma'(c)$. It returns (y, π).
- Finally, $\mathcal{B}$ outputs $b_\star$.

We note that if $f_0(st) = f_1(st)$ then we have $f_b(st) = (f'(st) \bmod p) \bmod q$ and $\Gamma(st) = (\Gamma'(st) \bmod p) \bmod q$. Finally, $pk_u^{V_u(st)} pk_c^{f_b(st)} = (g^{sk_u})^{sk_u^{-1}(\Gamma(st) - sk_c f_b(st))}(g^{sk_c})^{f_b(st)} = g^{\Gamma(st)}$.

We conclude that the n-IND-CFA experiment is successfully simulated for $\mathcal{A}$. Hence, the success of the adversary $\mathcal{A}$ is equal to the randomly guessing the value b. This gives

$$\Pr\left[1 \leftarrow \mathsf{Exp}_{\mathsf{PriVC},\mathcal{A}}^{\text{n-IND-CFA}}(\lambda)\right] = \frac{1}{2}$$

and the $\mathsf{Adv}_{\mathsf{PriVC},\mathcal{A}}^{\text{n-IND-CFA}}(\lambda)$ is negligible. Hence, the PriVC scheme is n-IND-CFA secure. □

6.3.2 Unforgeability

We show that our scheme is unforgeable under discrete logarithm assumption.

Theorem 15. *The proof of computation in PriVC is UNF-secure under the discrete logarithm assumption.*

Proof. We show that if an adversary $\mathcal{A}$ can successfully forge the proof of computation then it can break the discrete logarithm assumption by computing $\log_g pk_c$. Let $(x_\star, y_\star, \pi_\star)$ be a forged result with $\mathsf{VerifyResult}(y_\star, \pi_\star, \mathsf{vk}_f^u, \mathsf{k_u}, x_\star, \mathsf{pub}) = 1$ and $f(x_\star) \neq y_\star$.

$\mathcal{A}$ can compute $\Gamma' = (\pi_\star \bmod p) \bmod q$. We have

$$pk_u^{V_u(x_\star)} pk_c^{y_\star} = g^{\Gamma'} \qquad \text{and} \qquad pk_u^{V_u(x_\star)} pk_c^{f(x_\star)} = g^{\Gamma(x_\star)}.$$

From this, the adversary $\mathcal{A}$ can deduce that

$$pk_c = g^{\dfrac{\Gamma' - \Gamma(x_\star)}{y_\star - f(x_\star)}}.$$

Finally, $\mathcal{A}$ computes $\log_g pk_c = \dfrac{\Gamma' - \Gamma(x_\star)}{y_\star - f(x_\star)}$. Based on discrete logarithm assumption, there cannot exist an adversary $\mathcal{A}$ such that the advantage $\mathsf{Adv}^{UNF}_{\mathsf{PriVC},\mathcal{A}}(\lambda)$ is non-negligible. $\square$

User non-repudiation: Assume that $(c, \pi_a = (R, S))$ be a valid query for a user u and the server creates a fake query $(c', \pi'_a = (R', S))$ for the user u. If $H(c) = H(c')$, then the server can set $\pi'_a = \pi_a$ and (c', π'_a) becomes a valid query for the user u. Since the hash function H is assumed to be a collision resistant, it is computationally difficult to find c' such that $H(c) = H(c')$. For any (c', R'), the query (c', π'_a) is valid for the user u if only if $V_u(H(c')) = R'$. Since the verification function $V_u(x)$ is not public and $V_u(x) \in \mathbb{Z}_q[x]$, the probability of $V_u(H(c')) = R'$ is $1/q$. For sufficient value of q, this probability is negligible.

The algorithm $\mathsf{VerifyAuth}$ ensures that an unauthorized user cannot access the service. Since verification key's of each user is kept secret, the probability that an unauthorized user generates a valid query (c, π_a) where $\pi_a = (R, S)$ is equal to the probability of $R = V_u(H(c))$ for any user u. Since $V_u(x)$ is in $\mathbb{Z}_q[x]$, the probability of $R = V_u(H(c))$ for any random R is $1/q$.

Security of Encryption: For encryption of user data, we are using a symmetric version of the extended DGHV scheme proposed by Pedro *et al.* [58]. They have proved that their scheme is semantically secure under the assumption of approximate GCD problem. However, we note that there are two attacks on DGHV scheme, and we show that the extended DGHV scheme is secure under both the attacks. First is key recovery attack [59] and another is lattice attack [60].

In the key recovery attack, the adversary has access to the decryption oracle, and it can successfully find the key p after a few communications with the decryption oracle. The attack was successful because B is equal to 2 in the DGHV scheme, and therefore, the adversary was able to reduce the search space in half after every communication with the decryption oracle. In our scheme, B is a prime q of $\mathcal{O}(\lambda)$

bits. Therefore, the adversary can reduce the search space down to the size of an order of 2^λ, and it still has to do brute force for 2^λ many numbers. Therefore, the extended DGHV scheme is secure under a key recovery attack. In the lattice attack, the adversary considers a vector of several ciphertexts together and reduces the search space of vectors of plaintexts to the size of B^2. For $B = 2$, this is just 4, and if the plaintexts are binary, then it is easy to find the correct message out of 4. We have used $B = q$ where q is $\mathcal{O}(\lambda)$ bit prime. Therefore, the lattice attack can reduce the search space of vectors of plaintexts to the size of $B^2 \approx 2^{2\lambda}$. Hence, the extended DGHV scheme is secure under lattice attack as well.

We note that if a user collude with the CSP, the CSP can provide the polynomial to the user. This problem is inherent and cannot be prevented. Client can share verification key with the server but it does not help the server to forge the proof of computation for other users as each user has different verification key.

6.4 Experimental Results

To check the practicality of the scheme, we have implemented the PriVC scheme with realistic parameters.

Table 6.1: Performance level for various test instances

Instance	**EvalRequest**	**VerifyAuth**	**Eval**	**VerifyResult**	**VerifyTrans**
Tiny	3.25	0.176	0.169	0.154	0.122
Small	20.8	0.645	0.884	0.739	0.587
Medium	125	3.13	4.92	4.48	3.08
Large	651	14.2	32.3	25.3	14.1

All times in the Table 6.1 are in milliseconds.

In this section, we describe the implementation of PriVC and parameters selection. To prevent key recovery attack and lattice attacks on extended DGHV scheme, we must have $\rho = \lambda$, $\eta = \mathcal{O}(\lambda^2)$, $\gamma = \mathcal{O}(\lambda^5)$. Throughout our experiment, the degree of the polynomial $f(x)$ and $\Gamma(x)$ is 10. The coefficients of all three polynomials are randomly selected from $\mathbb{Z}_q$ where q is a 256 bit integer.

We considered four test instances with different security level as described in the Table 6.2. We have implemented PriVC scheme using SageMath version 8.1

on a quad-core desktop computer with Intel Core i5 − 6500 at 3.2 GHz and 4 GB RAM.

Table 6.2: Parameters used for various test instances

Instance	λ[a]	ρ[b]	η[b]	γ[b]
Tiny	42	27	1026	150000
Small	52	41	1558	830000
Medium	62	56	2128	4200000
Large	72	71	2698	19350000

[a] λ denotes security level.
[b] ρ, η and γ denotes bit-size of parameters in extended DGHV.

Table 6.1 summarizes the performance of our implementation of PriVC. We considered four instances for our experiment, namely Tiny, Small, Medium, and Large. For each instance, we use parameters as described in the Table 6.2. The time in the Table 8.5 represents the running time (in milliseconds) of a single query averaged over 10000 iterations. When we increase the security level, the size of the encrypted query increases, this leads to more computational cost during query generation, and function evaluation. The encrypted query generation depends on the extended DGHV parameters. The size of the parameters for the extended DGHV scheme increases as we increase the security level. As expected, the query generation time increases as we increase the security of the underlying encryption scheme. Even for high-security level, Large instance, the computational cost for each process is a fraction of a second.

The result verification involves mainly 3 exponentiation and polynomial evaluation. The algorithm VerifyResult first decrypts the result and then verify it using the verification equation. The time for VerifyResult in Table 6.1 is the total time required for decrypting the result and then verifying it. The time required for result verification is reasonably small compared to actual polynomial evaluation time. A quick look at the execution times in Table 6.1 shows that our scheme is efficient and suitable for the practical purpose.

6.5 Conclusion

In this chapter, we provide a construction of a privacy-preserving verifiable polynomial evaluation scheme. This scheme allows a company to delegate computation of a secret polynomial to the CSP so that a user can verify the computation done by the cloud server. In any delegation of computation schemes, a user must communicate with the cloud server, and therefore, the cloud server needs to authenticate the user before providing any service to the user. The proposed scheme also allows CSP to authenticate the user in a privacy-preserving manner. Moreover, once the transaction is completed, the user cannot deny it later, as each query involves a user-specific verification key. Using formal security models for IND-CFA and UNF, we prove that our scheme PriVC is IND-CFA and UNF secure. We implemented the proposed scheme using SageMath version 8.1 and observed that the computation step's verification takes less time than an actual polynomial evaluation done by the CSP.

CHAPTER 7

DeDuplication with Cross-server Ownership

7.1 Introduction

Cloud infrastructure has been widely used for managing data and services of applications owned by industries, organizations, and individuals. The cloud-based storage services like Dropbox[46], Google Drive [47], and NetApp [48] are prime choices for most of the people for storing data. The advantages of public cloud services are data availability, reliability, and ease of data sharing at a reasonable price [61]. It is in the best interest of cloud service providers to utilize the available limited storage efficiently. The data deduplication is an integral part of the cloud storage system. If several users upload the same data on the same server, it is beneficial for the server to detect it and keep only a few copies of the data. If two users upload the same data on different storage servers, the ownership of the data remains with the first uploader, and the second server should be able to detect it inform the first uploader about it. The problem becomes more difficult when the data are in encrypted form. The users may encrypt their data either to preserves data privacy or due to company policy or to abide by legal regulations. If each user uses their own key to encrypt the data, then the same data produce different ciphertexts when encrypted by different users. In such a case, the cloud server cannot identify whether two ciphertexts are encryption of the same data, and hence cross-user deduplication is not possible. At the same time, sharing the same encryption key among all the clients compromises the whole purpose of encryption.

There are several data deduplication schemes in the literature which considers ownership issue [83, 84, 85]. In all such schemes, the a user is considered an owner of the data if the user can prove the possession of the data to the cloud server. This certainly does not mean that the user is the creator of the data. Consider the scenario where there are several cloud data storage servers. Let user u creates a file F and uploads it to the cloud storage server CSS_1. If another user obtain this file F somehow and uploads it to CSS_1, then the server CSS_1 does not notify about it to the user u as per the existing ownership protocols. Moreover, the user u does not know if any other user has uploaded similar file to any other cloud storage servers. To address this ownership issue, we need corroboration among the servers, which is implausible. Hence we introduced a new entity called the trusted third party (TTP), which is assumed to be trusted by everyone in the system. Each server registers with the TTP and forwards the data to it. The root server checks for ownership of the data and alerts the server of duplicate data, if any. Now, as the workload of the TTP is computationally intense, and it would not be wise to upload the ciphertext to the TTP. Hence, we used the concept of message locked encryption, which in turn uses tags(which are again message derived hashes) to do the task.

In 2002, Douceur *et al.* introduced a convergent encryption scheme, which is a base technique to ensure data deduplication. For any given data M, a user derives encryption key $K = H(M)$ where H is a cryptographic hash function and then encrypts data M as $C = E(K, M)$. If two different users encrypt the same data, then the encryption key still remains the same, and it is easy to check for deduplication. Convergent encryption is susceptible to an offline brute-force dictionary attack. If it is known that the ciphertext C is an encryption of one of the messages from the dictionary of size n, the attacker can use brute-force over the dictionary and recover the message in the time of n offline encryptions.

In any deterministic scheme, identical plaintexts will be mapped to one ciphertext. Bellare *et. al*[25] introduced message-locked encryption scheme $MLE = (P, K, E, D, T)$ where P is the public parameter, K is the key generation algorithm used to compute the message derived key, E is the encryption algorithm, T is the tag generation algorithm while D is the decryption algorithm. All these are deterministic, and hence same tags imply identical plaintext. They later improvised [25, 26] to make a more efficient method that reduces the number of passes to just one as well as counters the tag consistency attack. In the standard MLE hash-and-CE scheme, the user can alter the message to M' without affecting the tag generation algorithm and hence get away with a fake upload. This is the tag consistency attack. In the improvised version, they recomputed the tag at the decryption algorithm to counter this TC attack. They further randomized the encryption scheme to ensure all the steps take place in a single pass. This new scheme, called Randomized Convergent Encryption, is the basis of this project. Abadi *et. al*[27] improvised this and provided stronger security measures. Their scheme avoided tags that are calculated deterministically from the message. The tags, in addition to the encryption algorithm, were also computed randomly here as $\mathcal{T}=(g^r, g^{rh(m)})$ where r is a random number and h is a strong collision-resistant hash function. This approach is inefficient for operations involving tag searching. Moreover, randomized tags does not provide indistinguishability property as it is possible to distinguish two data by comparing corresponding tags. For the proposed scheme, we consider deterministic tags.

In this chapter, we introduce a cross-user data deduplication and cross-server ownership scheme, DeDOP. The proposed DeDOP scheme achieves the goals of cross-user data deduplication and cross-server ownership issues using the Randomized Convergent Encryption(RCE). We show that the proposed schemes ensure tag consistency, which detects data faking attacks at the client level. We further ensure that the scheme is secure against a replay attack.

7.2 Preliminaries

The proposed scheme uses randomized convergent encryption to encrypt the data. The randomized convergent encryption provides security against chosen distribution attack where an adversary can identify whether a ciphertext is an encryption of a message which comes from a chosen distribution.

Definition 29 (Randomized Convergent Encryption). *A Randomized Convergent*

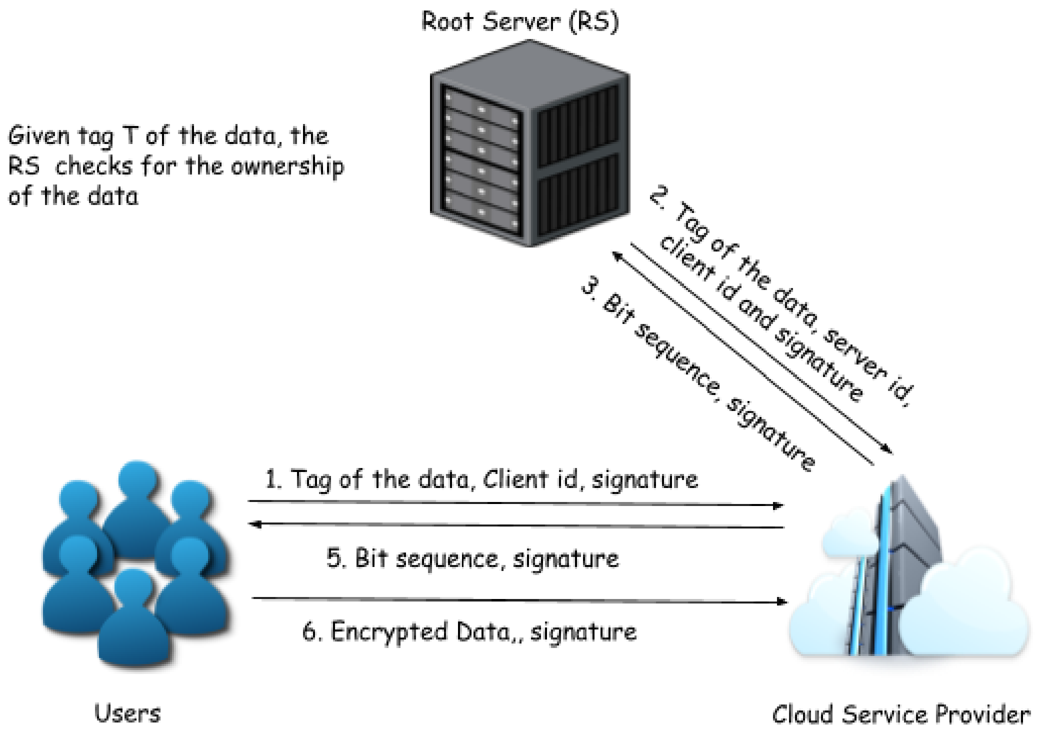

Root Server (RS)
Given tag T of the data, the RS checks for the ownership of the data
2. Tag of the data, server id, client id and signature
3. Bit sequence, signature
1. Tag of the data, Client id, signature
5. Bit sequence, signature
6. Encrypted Data,, signature
Users
Cloud Service Provider

Figure 7.1: Illustration of deduplication and cross-server Ownership

Encryption scheme is a five-tuple of algorithms: parameter generation, key generation, encryption, decryption, tag generation.

- *ParamGen(λ): Given security parameter λ, it generates public parameters P. It selects a cryptographically secure hash function H and secure symmetric encryption scheme $SE = (E, D)$.*

- *KeyGen((m, P)): It returns message derived key $K = H(P, m)$.*

- *Encrypt(m, K): It picks random $L \in 0, 1^k$ and computes $c_1 = E(L, m), c_2 = L \oplus K$ where E is a symmetric encryption algorithm. The ciphertext is $c = c_1 || c_2$.*

- *Decrypt(c, K): It first gets c_2 from $c = c_1 || c_2$ and computes the key $L = c_2 \oplus K$. It decrypts c_1 using L and returns $m = D(L, c_1)$ where D is a symmetric decryption algorithm.*

- *Tag(K, P): It returns tag $T = H(P, K)$.*

Definition 30 (Tag Consistency (TC))**.** *A MLE scheme is said to be TC secure if a client can verify correctness of a tag corresponding to a retrieved file.*

Suppose an adversary uploads a fake file for a new tag T. Later if an honest user tries to upload the correct file for the tag T, then the server doesn't upload the new file as there is already one file for the tag T. The user assumes that a copy of the file is already stored in the server. When the user retrieves the file using tag T, he/she will receive a fake file. TC security prevents this as the user can check tag consistency. The RCE is a TC secure scheme.

7.3 Scheme Description

Our system model for DeDOP scheme mainly contains three entities: Clients, Cloud Storage servers (CSS) and a trusted third party (TTP).

- Clients: A client is an individual or an organization that has large files and stores it on the public cloud.
- Cloud Storage Server: Cloud storage server (CSS) is a data storage server maintained by a cloud service provider who has a vast amount of storage space. For efficient use of the storage space, the CSS performs data deduplication over the stored data. Each cloud storage server in the system is registered with the trusted third party, TTP.
- Trusted Third Party: The trusted third party maintains tags for all the data stored on the cloud storage servers. The TTP is responsible for ownership and duplicate check at the time of data upload.

Each client and thr storage server receives unique identity from the TTP. For each data block stored in any of the registered server, the TTP keeps its tag value and corresponding owner id (id of the user who uploads it first time), server ids (ids of DBS where it is stored) and user ids (ids of other users who uploaded the same data). The TTP maintains a hash-map dictionary with tag as key and a set of owner id, server ids and other users ids as value. A storage server keeps list of user ids along with tag sequences of files stored by each client. For each tag stored in the DBS, it keeps corresponding ciphertext. Each storage server maintains two hash-map dictionaries. First is D_1 with client id as key and corresponding list of tag sequences as a value. Second is D_2 with tag as key and corresponding ciphertext as value.

The proposed DeDOP scheme has following goals: (i) cross-user data deduplication for encrypted data; (ii) tackle cross-server ownership issues. The convergent encryption scheme provides confidentiality of user's data and cross-user data deduplication. A convergent encryption with tag consistency allows cross-user deduplication check and also prevents data faking attacks. The DeDOP scheme works as follows. Consider that a user u_{id} wants to upload a file F to the server S_j. It first breaks the file F in fixed sized blocks. Let $F = B_1 B_2 \dots B_n$. The user then computes tag for τ_k for each block B_k and sends the tag sequence $T = \tau_1 \tau_2 \tau_3 \dots \tau_n$ to the server S_j. The server forwards it to the TTP along with the user id, u_{id}. The RS breaks the tag sequence in individual tags and looks up in the table for duplicate check. The TTP sends a binary sequence $b_1 b_2 b_3 \dots b_n$ to the server S_j. For each $1 \leq i \leq n$, $b_i = 0$ if the i-th tag is already stored in the server S_j else $b_i = 1$. The server forwards the binary sequence to the client and asks for only those ciphertexts for which $b_i = 0$. At the time of tag search by the TTP, if more than α fraction of tags are already present in the system, then it checks whether there is any other user who owns at least α fraction of these tags. It then informs the corresponding storage server about possible cross-server almost duplicate file. Once the user u_{id} receives the bit sequence b from the server, it encrypts only those blocks for which $b_i = 1$ and sends it to the server for storage.

7.3.1 Assumptions

We assume that each cloud storage server is registered with the TTP. Each server has a private key sk_s known only to them and corresponding public key pk_s which is known to every registered user. Each user is registered with at least one storage server and has a private key sk_u known only to them and a public key pk_u which is known to all involved parties. Similarly, the trusted third party also has a private key sk_{tp} whose public key pk_{tp} is known to every other registered entities.

We also assume that a registered user can be malicious. A registered user may act as data faking adversary who tries to upload fake data for a valid tag. The cloud storage servers and TTP are trusted entities.

7.3.2 Security Model

Any data deduplication scheme should be secure against duplicate data faking attacks. In duplicate data faking attacks, an attacker replaces valid data by fake data without being detected. Consider a scenario where an attacker uploads a fake data F' with some tag T. Later, an honest client tries to upload a file F with the correct tag T on the server. Since the tag T is already stored on the server, the server keeps only one copy of the data, and it is F'. When the client downloads the file for tag T, it receives F' instead of F and cannot check whether it is the original file. Therefore, it is necessary to check whether the tag and the file are related at the time of download. To prevent duplicate faking attacks, we consider the Tag Consistency security model. In TC, we assume that the adversary has access to public parameters as well as a valid target file and corresponding tags. The goal of the adversary is to create a fake file that can pass through the file upload process.

Definition 31 (Tag Consistency (STC)). *Let Π be a data deduplication scheme and $\mathcal{A}$ be a PPT adversary. The Tag Consistency (TC) experiment of $\mathcal{A}$ against Π is defined as follows:*

```
$\mathsf{Exp}^{\mathsf{TC}}_{\Pi,\mathcal{A}}(\lambda)$:
    $\mathsf{pub} \leftarrow \mathtt{Setup}(\lambda)$ ;
    $(B, B') \leftarrow \mathcal{A}(\mathsf{pub})$ ;
    $(u_{id}, t, sign_{sk_u}\{t\}) \leftarrow \mathtt{TagGen}(u_{id}, B, \mathsf{pub}, sk_u)$ ;
    $K \leftarrow \mathtt{KeyGen}(B, \mathsf{pub})$ ;
    $C \leftarrow \mathtt{Encrypt}(B, K)$ ;
    If $(B \neq B')$ and $\mathtt{FileCheck(C,t,pub)}$:
      return $\mathsf{true}$ ;
    Else return $\mathsf{false}$.
```

The advantage of $\mathcal{A}$ against the STC experiment is given by:

$$Adv^{STC}_{\Pi,\mathcal{A}}(\lambda) = \Pr\left[Exp^{STC}_{\Pi,\mathcal{A}}(\lambda) = true\right] .$$

A scheme Π is STC secure if this advantage is negligible for any $\mathcal{A} \in \text{POLY}(\lambda)^2$.

7.4 The proposed scheme, DeDOP

The DeDOP contains three main phases: Setup phase, Deduplication phase and File upload and file check phase. At the time of registration, a storage server receives a server id (s_{id}) from the TTP and a user receives user id (u_{id}) from the database server. The details of each phase is as follows:

- $\mathsf{Setup}(\lambda)$: Using the security parameter λ, this algorithm first generates the public parameters P for the encryption scheme RCE. It also selects a cryptographically secure hash function H, a secure symmetric key encryption scheme $SE = (E, D)$ and a secure signature algorithm $\text{sign}_{sk}(.)$. It sets β as block length and α as threshold fraction for ownership check. Finally, it outputs $\mathsf{pub} = (\alpha, \beta, H, P, SE, \text{sign}_{sk}(.))$.

- Deduplication Phase.

  - $\mathsf{TagGen}(u_{id}, F, \mathsf{pub}, sk_u)$: Let $F = B_1||B_2||\ldots||B_n$ where each B_i is a block of a length β. Using RCE, the client computes the tag sequence T for the file F as follows: For each $1 \leq i \leq n$, it computes
    1. $K_i \leftarrow H(P, B_i)$
    2. $\tau_i \leftarrow H(P, K_i)$.

    The tag sequence for the file F is $T = \tau_1||\tau_2||\ldots||\tau_n$. The client signs the tag sequence using secret key and sends $(u_{id}, T, \text{sign}_{sk_u}\{T\})$ to the storage server S_j. The user stores encryption keys $\{K_i\}_{i=1}^{n}$ locally.

  - $\mathsf{DupCheck}(u_{id}, T, \mathsf{pub}, \text{sign}_{sk_u}\{T\})$: The server S_j verifies the signature and stores (u_{id}, T). It forwards the tag sequence to the TTP along with the signature $\text{sign}_{sk_s}\{T\}$.

For the tag sequence T, the TTP creates a bit string $b = b_1 b_2 \ldots b_n$ where $b_i = 1$ initially. For each tag t_i, the TTP checks whether it already exist in the system. If there is a match, then it checks whether the duplicate tag is stored on the same storage server, S_j. The TTP sets $b_i = 0$ if the duplicate tag is stored on the same server. For each new tag t_k, the TTP stores $(t_k, (u_{id}, s_{id}))$. For each existing tag, it appends (u_{id}, s_{id}) to the entry of the tag. if the number of duplicate tags is more than the threshold, then for each duplicate tag, the TTP compares the user id with the owner id of the tag. If the user id is not same as owner id, then it computes total number of tags that matches with tags of a particular owner id. If the proportion of matches is more than a threshold, then it notifies the owner through a storage server. The TTP sends b to the storage server along with the signature $\text{sign}_{sk_r}\{b\}$. The storage server forwards it to the client.

- File upload and file check Phase.

  - FileUpload$(b = b_1 || b_2 || \ldots || b_n, \text{sign}_{sk_r}\{b\})$:
    If $b_i = 1$, then user encrypts the block B_i as follows:

    1. $L_i \leftarrow \{0, 1\}^k$
    2. $c_{1i} \leftarrow E(L_i, B_i)$
    3. $c_{2i} \leftarrow L_i \oplus K_i$.

    The ciphertext for the word B_i is $c_i = c_{1i} || c_{2i}$. The user sends $(c = \{c_i\}_{i=1}^{m}, \text{sign}_{sk_u}(c))$ to the server.

  - FileCheck$(C, T, \text{pub}, \{K_i\}_{i=1}^{n})$: Whenever a user downloads a file C from the server, it first verifies the corresponding tag T for the file. It first breaks C as $C = c_1 || c_2 || c_3 || \ldots || c_n$ and $T = \tau_1 || \tau_2 || \tau_3 || \ldots || \tau_n$. For each $1 \leq i \leq n$, if $\tau_i = H(P, K_i)$, then proceed; else aborts. Then it computes

$L_i = c_{2i} \oplus K_i$ and decrypts c_{1i} as $B'_i = D(L_i, c_{1i})$. If $K_i = H(P, B'_i)$, then accept the file, else abort.

7.5 Security Analysis

A client encrypts data to keep the data confidential from everyone else. We need to make sure that the encryption method used in the scheme ensures data privacy.

Theorem 16. *Let H be a cryptographically secure hash function and $SE = (E, D)$ be a secure symmetric encryption scheme. Then the* RCE *scheme is Privacy against chosen distribution attack (PRV-CDA) secure.*

Bellare *et. al* [26] provided proof of the above theorem. Since the underlying encryption scheme is PRV-CDA secure, the proposed scheme provides data privacy.

If a fake file upload attack, an attacker uploads a fake file for a tag of the genuine file. Later, an honest client tries to upload an original file for the same tag. Since the tag already exists in the system, the client does not upload the file, and the DBS links a fake file to the client. At the time of retrieval, the client receives a fake file. TC security prevents this attack as a client can always check the tag consistency. Since the RCE scheme is TC secure, the proposed DeDOP scheme is also TC secure.

In a public channel, an attacker may get the tag sequence uploaded by another client. Later, the attacker sends the tag sequence to the server. The server follows the protocol and forwards it to the TTP. Since the corresponding file is already stored in the server, the TTP returns bit sequence $b = b_1 b_2 \ldots b_n$ with all $b_i = 0$. The attacker does not have to upload any block of the file, and the attcker will be able to access the file. However, the file is encrypted using massaged derived keys. Without the keys, the attacker does not learn anything about the data and does not gain any information.

7.6 Conclusions

In this chapter, we discussed the importance of ownership of data in a multi-server scenario. We reviewed several existing ownership and data deduplication schemes. It considers a user as an owner of the data if the user possesses some meta-data in all those schemes. However, it does not check whether the user is the creator of the data in the system. We proposed an approach that can tackle the ownership issue and data deduplication in multiple storage server scenarios. We have used the randomized convergent encryption scheme of the message locked encryption with deterministic tags. We note that the proposed scheme provides tag consistency at the client level but does not provide tag consistency at the server level.

CHAPTER 8

Data accountability in cloud storage

8.1 Introduction

The main advantages of storing data on the public cloud are data availability, reliability, and ease of data sharing at a reasonable price [61]. However, storing data, in particular, sensitive data, on public cloud storage, becomes a potential target of an attacker. If sensitive data gets leaked or compromised, then the data owner and service provider could face problems in protecting user data from malicious intent. Recently, a freelance photographer Dave Cooper lost around $100,000$ video clips due to a bug in Adobe Premiere Pro [52]. He was able to detect deleted files as it is significant in numbers. Now imagine only one of the $100,000$ clips gets removed due to a bug. Will Dave be able to detect it? Clients (People or organizations) store their data on the cloud without keeping any local copy of it, and this makes it difficult for the client to detect a slight modification or deletion in the data as it is hard to remember complete data. If the cloud service provider voluntarily discloses the data modification, then it may affect their reputation, and it is in their best interest not to reveal any data damage that could lead to potential harm to the company as well as an individual. To keep the data intact, the system must have a mechanism to verify the integrity of the data stored on the public cloud.

Consider a scenario of a healthcare application where a healthcare company provides predictions based on a client's medical history. In such a situation, clients of the healthcare company may want to keep a history of all their data by storing it on the cloud without keeping any local copy. The decision-making process in the medical field also depends on past data of a patient. A partial or full modification or deletion of previous data stored on the cloud may affect a health-related decision-making process of the patient. It is necessary to have a specific mechanism to verify the integrity of the data stored in the cloud. Once a user uploads data on the cloud storage, will it remain as it is overtime? Without saving the complete data locally, how does a user make sure that the file is not modified? The trivial way to ensure this is to download the whole file and compare the hash value of the file with a locally stored hash value. If the file size is large and the user has constrained computational resources, then the trivial method is inefficient as it involves large data transmission. This method is not suitable for a system where the integrity of data is checked frequently [82]. To provide data integrity with minimum communication cost, researches introduced several techniques like Proof of Storage (PoS), Provable Data Possession (PDP), Proof of Retrievability (POR), etc.

We introduce a new Proof of Storage with data deduplication scheme (DPoS). The DPoS scheme has following goals: (i) cross-user data deduplication at block-level; (ii) provide proof of data stored. We use a variant of VIP-POPE scheme proposed in previous chapter. In previous chapter, we trusted the cloud with the polynomial f but not with the computation using f. The scheme provided proof of computation. In this scenario, we don't trust the cloud with data. Using same VIP-POPE scheme, we can generate proof of data by treating the file F as polynomial by considering blocks of the file as coefficients of the the polynomial. In the proposed scheme, there is a single tag for each block which can act as tag during data deduplication and meta data during proof of storage process. It is also possible to verify tag and block relation at the time of file upload. Without tag consistency check, a user can submit a fake file for a tag. The DPoS scheme works as follows. We assume that each client is registered with the cloud server. To upload a file, a client first breaks the file in blocks and generates tag sequence for the file. It sends tag sequence to the cloud storage server CSS. The CSS breaks the tag sequence in individual tags and looks up in the dictionary for each tag. The CSS sends a binary sequence $b_1 b_2 b_3 \dots b_n$ to the client. For each $1 \leq i \leq n$,

$b_i = 1$ if the i-th tag is already stored in the CSS; else, $b_i = 0$. The client send data only for those blocks for which $b_i = 0$. For each uploaded block, the CSS checks whether the tag is consistent with block. For proof of storage, a client picks a random number x and an index set for challenged blocks. The CSS treats those challenged blocks as coefficients of a polynomial and evaluate the polynomial at x along with the proof. The client can verify the integrity of the data using secret value and proof provided by the CSS.

8.2 System Model and Assumptions

8.2.1 System Model

Several previous proof of storage schemes considers trusted third party for auditing data stored on the server [36, 40, 39]. In DPoS, we do not require any trusted third party auditor. The system model of DPoS has mainly two entities:

- Clients: A client is an individual or an organization that has large files and stores it on the public cloud. Whenever required, the client can demand proof of the integrity of the data from the cloud.

- Cloud Storage Server: Cloud storage server (CSS) is a data storage server maintained by a cloud service provider who has a vast amount of storage space. For efficient use of the storage space, the CSS performs data deduplication over the stored data.

A client stores large files on the CSS and keeps a local copy of only metadata related to the files. At the time of the file upload, the CSS performs data deduplication for efficient storage management. If two data are the same, then the CSS keeps only one copy of the data. As a client does not have any local copy of the data, the client should be able to check the integrity of the data stored on the CSS. To perform a data integrity check, a client sends a challenge to the CSS for a specific file. In response, the CSS provides proof for the integrity of the data using a challenge and the file stored on the server. Using metadata related to the file and verification parameters, the client verifies the proof.

In the proposed DPoS scheme, a tag, which is used during the data deduplication and the proof of the storage process, is computed for each data block. It is also possible to verify the consistency of the tag at the time of file upload. Without a tag consistency check, a user can submit a fake file for a valid tag. We assume that each client is registered with the cloud server. To upload a file, a client first breaks the file in blocks and generates a tag sequence for the file. It sends the tag sequence to the cloud storage server (CSS). The CSS breaks the tag sequence in individual tags and looks up in the dictionary for each tag. The CSS sends a binary sequence $b_1b_2b_3\dots b_n$ to the client. For each $1 \leq i \leq n$, $b_i = 1$ if the i-th tag is already stored in the CSS; else, $b_i = 0$. The client send data only for those blocks for which $b_i = 0$. For each uploaded block, the CSS checks whether the tag is consistent with block. For proof of storage, a client picks a random number x and an index set for challenged blocks. The CSS treats those challenged blocks as coefficients of a vector and evaluates the inner product with the challenge vector along with the proof. The client can verify the integrity of the data using secret value and proof provided by the CSS.

8.2.2 Assumptions

We assume that the cloud storage server is a semi-trusted. We do not trust the cloud with the data stored on the server as data are prone to human or machine error. It can act as a forgery adversary who tries to hide the data corruption by forging a valid proof using corrupted data. A forgery adversary has access to the valid data corresponding to the corrupted data.

We also assume that a client can also be malicious. A registered client may act as data faking adversary who tries to upload fake data along with valid tag in the system. A data faking adversary has an access to the valid file and is registered on the CSS.

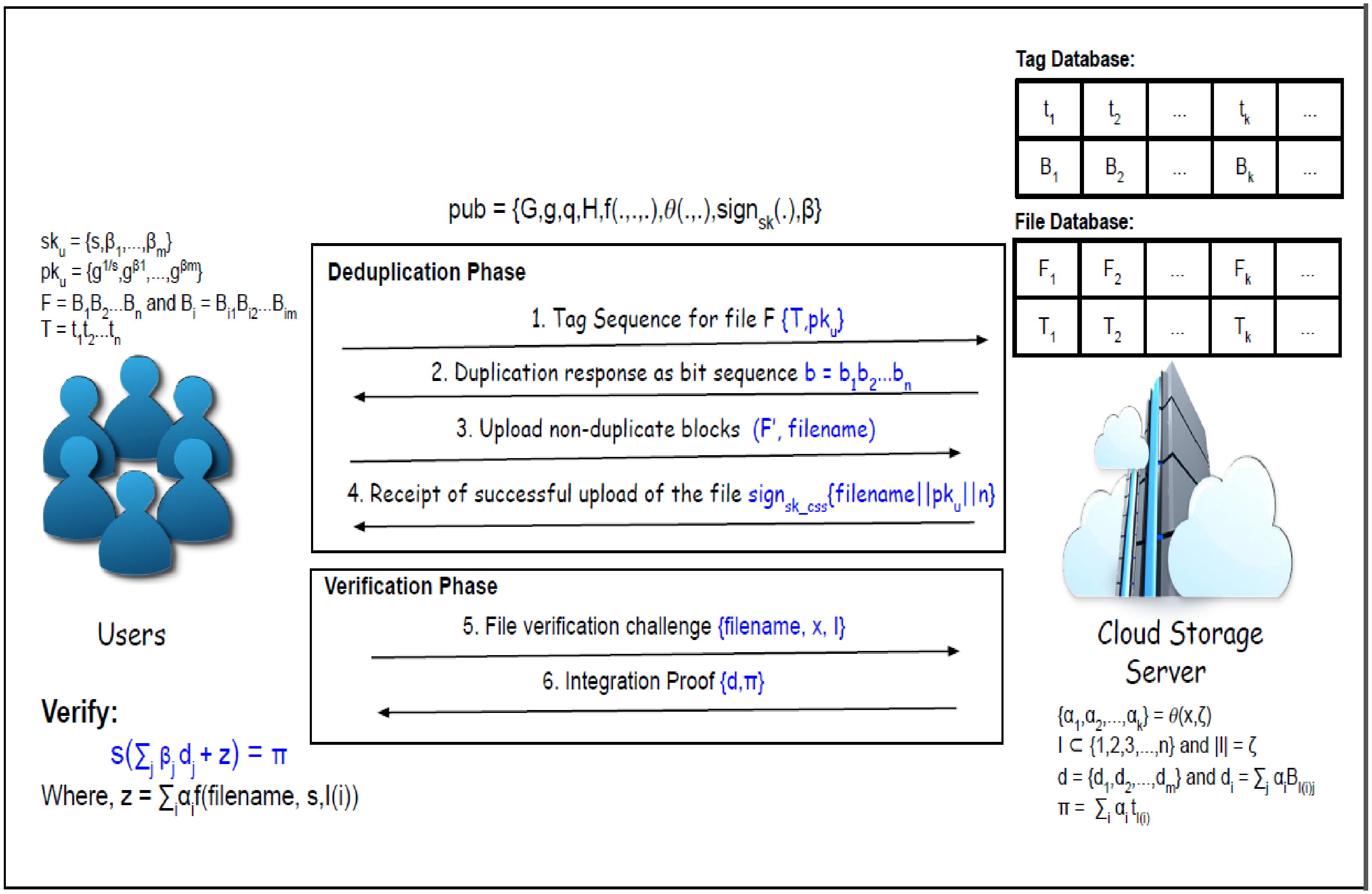

pub = {G,g,q,H,f(.,.,.),θ(.,.),sign_sk(.),β}
sk_u = {s,β_1,…,β_m}
pk_u = {g^{1/s},g^{β1},…,g^{βm}}
F = B_1B_2…B_n and B_i = B_i1B_i2…B_im
T = t_1t_2…t_n
Users
Verify:
s(∑_j β_j d_j + z) = π
Where, z = ∑_i α_i f(filename, s,I(i))
Deduplication Phase
1. Tag Sequence for file F {T,pk_u}
2. Duplication response as bit sequence b = b_1b_2…b_n
3. Upload non-duplicate blocks (F', filename)
4. Receipt of successful upload of the file sign_sk_css{filename||pk_u||n}
Verification Phase
5. File verification challenge {filename, x, I}
6. Integration Proof {d,π}
Tag Database:
t_1 | t_2 | … | t_k | …
B_1 | B_2 | … | B_k | …
File Database:
F_1 | F_2 | … | F_k | …
T_1 | T_2 | … | T_k | …
Cloud Storage Server
{α_1,α_2,…,α_k} = θ(x,ζ)
I ⊂ {1,2,3,…,n} and |I| = ζ
d = {d_1,d_2,…,d_m} and d_i = ∑_j α_i B_I(i)j
π = ∑_i α_i t_I(i)

Figure 8.1: The proposed DPoS scheme DPoS

8.2.3 Security Model

Any proof of storage with data deduplication scheme should be secure against duplicate data faking attacks as well as forgery attack. In duplicate data faking attacks, an attacker replaces valid data by fake data without being detected. Consider a scenario where an attacker uploads a fake data F' with some tag T. Later, an honest client tries to upload a file F with the correct tag T on the server. Since the tag T is already stored on the server, the server keeps only one copy of the data, and it is F'. When the client downloads the file for tag T, it receives F' instead of F. Therefore, it is necessary to check whether the tag and the file are related at the time of upload. To prevent duplicate faking attacks, we consider the Stronger Tag Consistency security model. In STC, we assume that the adversary has access to public parameters as well as a valid target file and corresponding tags. The goal of the adversary is to create a fake file that can pass through the file upload process along with the tags of the target file. Note that a malicious user can act as an STC adversary.

Definition 32 (Stronger Tag Consistency (STC))**.** *Let Π be a data deduplication scheme and $\mathcal{A}$ be a PPT adversary. The Stronger Tag Consistency (STC) experiment of $\mathcal{A}$ against Π is defined as follows:*

```
$\mathbf{Exp}^{\mathbf{STC}}_{\Pi,\mathcal{A}}(\lambda)$:
    $\mathsf{pub} \leftarrow \mathtt{Setup}(\lambda)$ ;
    $(sk_u, pk_u) \leftarrow \mathtt{Init}(\mathsf{pub})$ ;
    $(B, B') \leftarrow \mathcal{A}(\mathsf{pub})$ ;
    $(t, \gamma, pk_u) \leftarrow \mathtt{TagGen}(B, \mathsf{pub}, sk_u, pk_u)$ ;
    If $(B \neq B')$ and $\mathtt{FileCheck}(B', t, pk_u, \mathsf{pub})$:
      return true ;
    Else return false.
```

The advantage of $\mathcal{A}$ against the STC experiment is given by:

$$Adv^{STC}_{\Pi,\mathcal{A}}(\lambda) = \Pr\left[Exp^{STC}_{\Pi,\mathcal{A}}(\lambda) = true\right] .$$

A scheme Π is STC secure if this advantage is negligible for any $\mathcal{A} \in \text{POLY}(\lambda)^2$.

In a forgery attack, an adversary has access to the target file along with corresponding tags. The goal of the forgery adversary is to provide valid data integrity proof with a modified target file. A malicious cloud server can act as a forgery adversary. To prevent this attack, we consider Unforegeability in the PoS security model.

Definition 33 (Unforgeability in PoS (UNF-PoS))**.** *Let Π be a Proof of Storage (PoS) scheme and $\mathcal{A}$ be a PPT adversary. The Unforgeability in PoS (UNF-PoS) experiment of $\mathcal{A}$ against Π is defined as follows:*

```
$\mathsf{Exp}^{\mathsf{UNF\text{-}PoS}}_{\Pi,\mathcal{A}}(\lambda)$:
    $\mathsf{pub} \leftarrow \mathtt{Setup}(\lambda)$ ;
    $(sk_u, pk_u) \leftarrow \mathtt{Init}(\mathsf{pub})$ ;
    $(\mathit{filename}, F, T) \leftarrow \mathcal{A}(\mathsf{pub}, pk_u)$ ;
    $x \leftarrow \mathbb{Z}_q^\star$ ;
    $I \subset \{1, 2, \cdots, n\}$ ;
    $(F', (d', \pi')) \leftarrow \mathcal{A}(x, I, \mathit{filename}, F, T, pk_u, \mathsf{pub})$ ;
    If $(F \neq F')$ and $\mathtt{PrivateProofVerify}(x, I, d', \pi', \mathit{filename}, sk_u)$:
      return $1$ ;
    Else return $0$.
```

The advantage of $\mathcal{A}$ against the UNF-PoS experiment is given by:

$$Adv^{UNF\text{-}PoS}_{\Pi,\mathcal{A}}(\lambda) = \Pr\left[Exp^{UNF\text{-}PoS}_{\Pi,\mathcal{A}}(\lambda) = 1\right] .$$

A scheme Π is UNF-PoS secure if the advantage $\mathsf{Adv}^{\mathsf{UNF\text{-}PoS}}_{\Pi,\mathcal{A}}(\lambda)$ is negligible for any $\mathcal{A} \in \text{POLY}(\lambda)^2$.

8.3 Construction of DPoS

We provide detailed construction of our proposed scheme. The Data Deduplication with Proof of Storage (DPoS) scheme is a 9-tuple algorithm Setup, Init, TagGen, DupCheck, FileUpload, FileCheck, ProofGen, PrivateProofVerify, PublicProofVerify. The proposed scheme DPoS is illustrated in the Figure 8.1. The detailed working of each of the algorithm is as follows:

- $\texttt{Setup}(\lambda)$: This algorithm is run by the CSS. This algorithm first generates a prime number q such that the complexity of DLP in $\mathbb{Z}_q$ is at least λ bits. It selects a cryptographically secure hash function $H : \{0,1\}^\star \to \mathbb{Z}_q$, a secure signature algorithm $\text{sign}_{sk}(.)$, a multiplicative group $G = \langle g \rangle$ of order q and a pseudo-random function $f : \{0,1\}^\star \times \mathbb{Z} \times \mathbb{Z}_q \to \mathbb{Z}_q$. It also selects another pseudo-random function $\theta(\cdot,\cdot)$ which takes an element of $\mathbb{Z}_q^\star$ and an integer m as an input and outputs m many random elements of $\mathbb{Z}_q^\star$. It outputs $\mathsf{pub} = (G, g, q, H, f, \theta, \text{sign}_{sk}(.), \beta)$ where β is a bit length of each block in a file.

- $\texttt{Init}(\mathsf{pub})$: This algorithm is run by the client. It randomly pick $s \in \mathbb{Z}_q^\times$ and computes $g^{1/s}$. It further selects m random numbers $\{\beta_i \in \mathbb{Z}_q^\times\}_{i=1}^m$, where $m = \frac{\beta}{\log_2(q)}$. It sets the client's secret key $\mathsf{sk}_u = \{s, \{\beta_i\}_{i=1}^m\}$ and the client's public key $\mathsf{pk}_u = \left\{g^{1/s}, \{g^{\beta_i}\}_{i=1}^m\right\}$.

- $\texttt{TagGen}(F, \mathsf{pub}, \mathsf{sk}_u, \mathsf{pk}_u)$: Let $F = B_1||B_2||\ldots||B_n$ where each $B_i = B_{i1}\, B_{i2} \ldots B_{im}$ is a block of a length β bits and B_{ij} is j-th sector of i-th block. The client first assign a name $\mathsf{filename}$ to the file. The client computes the tag sequence T for the file F as follows. For each $1 \le i \le n$, it computes

  1. $r_i = f(\mathsf{filename}, s, i)$
  2. $\gamma_i = g^{r_i}$
  3. $t_i = \left(\left(\sum_{j=1}^m \beta_j \cdot B_{ij}\right) + r_i\right) s.$

  The tag sequence for the file F is $T = \{t_i, \gamma_i, H(B_i)\}_{i=1}^n$. The client sends (T, pk_u) to the CSS.

- $\texttt{DupCheck}(T, \mathsf{pk}_u, \mathsf{pub})$: For each $1 \le i \le n$, the CSS takes $H(B_i)$ and compare it with the tags of already stored data. If there is a match found for some $i = k$, then it retrieves corresponding stored block B_k and verifies the equation 8.1:

$$\left(g^{1/s}\right)^{t_k} = \gamma_k \cdot \prod_{j=1}^{m} \left(g^{\beta_j}\right)^{B_{kj}} \tag{8.1}$$

For the file F, it creates a bit string $b = b_1 b_2 \cdots b_n$ where $b_i = 1$ if the tag t_i is already stored in the system; else, $b_i = 0$. The CSS sends b to the client.

- `FileUpload`(b, F) : The client creates a file $F' = B'_1 B'_2 \cdots B'_n$ where $B'_i = B_i$ if $b_i = 1$; else, $B'_i = Null$. The client sends $(F', \mathsf{filename})$ to the CSS and stores $(\mathsf{filename}, s, n)$.

- `FileCheck`$(b, F', \mathsf{filename}, T, \mathsf{pk}_u, \mathsf{pub})$: For each $1 \le i \le n$, if $b_i = 1$, then the CSS checks whether B'_i and $(t_i, \gamma_i, H(B_i))$ are related using equations 8.2 and 8.3:

$$[H(B'_i) = H(B_i) \tag{8.2}$$

$$\left(g^{1/s}\right)^{t_i} = \gamma_i \cdot \prod_{j=1}^{m} \left(g^{\beta_j}\right)^{B_{ij}} \tag{8.3}$$

If it does not holds true for at least one i, $1 \le i \le n$, then it returns 0 and aborts; else it returns 1 and proceeds further. If $b_i = 0$, then it links B'_i to already stored B_i. It stores F along with $\{\mathsf{filename}, T, \mathsf{pk}_u\}$. It sends $\mathrm{sign}_{sk_{css}}(\mathsf{filename}||\mathsf{pk}_u||n)$ as receipt to the client where sk_{css} is the signing key of the CSS.

- `ProofGen`$(\mathsf{filename}, x, I, F, T)$: For the target file, client picks a random number $x \in \mathbb{Z}_q^\star$ and a set of indices $I \subset \{1, 2, \cdots, n\}$ and sends it to the CSS. The CSS generates $\{\alpha_1, \alpha_2, \ldots, \alpha_\zeta\} = \theta(x, \zeta)$ where $\zeta = |I|$ and for $1 \le j \le m$, it computes

$$d_j = \sum_{i=1}^{\zeta} \alpha_i \cdot B_{I(i)j} \quad \text{and} \quad \pi = \sum_{i=1}^{\zeta} \alpha_i \cdot t_{I(i)}.$$

It sends $\left(d = \{d_j\}_{j=1}^{m}, \pi, \{\gamma_{I(i)}\}_{i=1}^{\zeta}\right)$ to the client.

- `PrivateProofVerify`$(x, I, d, \pi, \mathsf{filename}, \mathsf{sk}_u)$: The client first generates $\{\alpha_1, \alpha_2, \ldots, \alpha_\zeta\} = \theta(x, \zeta)$ where $\zeta = |I|$ and computes

$$z = \sum_{i=1}^{\zeta} \alpha_i \cdot f(\mathsf{filename}, s, I(i)).$$

  If $s\left(\sum_{j=1}^{m} \beta_j \cdot d_j + z\right) = \pi$, then the algorithm returns 1; else, it returns 0.

- `PublicProofVerify`$(x, I, d, \pi, \{\gamma_{I(i)}\}_{i=1}^{\zeta}, \mathsf{pk}_u)$: The public verifier first generates $\{\alpha_1, \alpha_2, \ldots, \alpha_\zeta\} = \theta(x, \zeta)$ where $\zeta = |I|$ and computes

$$Z = \prod_{i=1}^{\zeta} \gamma_{I(i)}^{\alpha_i}.$$

  If $\prod_{j=1}^{m}\left(g^{\beta_j}\right)^{d_j} \cdot Z = \left(g^{1/s}\right)^{\pi}$, then the algorithm returns 1; else, it returns 0.

Correctness:

- `FileCheck`:

$$\begin{aligned}
\left(g^{1/s}\right)^{t_i} &= \left(g^{1/s}\right)^{\left(\left(\sum_{j=1}^{m} \beta_j \cdot B_{ij}\right) + r_i\right)s} \\
&= g^{\left(\sum_{j=1}^{m} \beta_j \cdot B_{ij}\right) + r_i} \\
&= g^{\sum_{j=1}^{m} \beta_j \cdot B_{ij}} \gamma_i \\
&= \gamma_i \cdot \prod_{j=1}^{m} \left(g^{\beta_j}\right)^{B_{ij}}.
\end{aligned}$$

- `PrivateProofVerify`:

$$
\begin{aligned}
s\left(\sum_{j=1}^{m}\beta_j\cdot d_j+z\right) &= s\left(\sum_{j=1}^{m}\beta_j\sum_{i=1}^{\zeta}\alpha_i B_{I(i)j}+\sum_{i=1}^{\zeta}\alpha_i r_{I(i)}\right)\\
&=\sum_{i=1}^{\zeta}\alpha_i(\sum_{j=1}^{m}\beta_j B_{I(i)j}+r_{I(i)})s\\
&=\sum_{i=1}^{\zeta}\alpha_i t_{I(i)}\\
&=\pi.
\end{aligned}
$$

- `PublicProofVerify`:

$$
\begin{aligned}
\prod_{j=1}^{m}\left(g^{\beta_j}\right)^{d_j}\cdot Z &= \prod_{j=1}^{m} g^{\left(\beta_j\sum_{i=1}^{\zeta}\alpha_i B_{I(i)j}\right)}\cdot\prod_{i=1}^{\zeta}\gamma_{I(i)}^{\alpha_i}\\
&=\prod_{j=1}^{m}\prod_{i=1}^{\zeta} g^{\left(\beta_j\alpha_i B_{I(i)j}\right)}\cdot\prod_{i=1}^{\zeta} g^{r_{I(i)}\alpha_i}\\
&=\prod_{i=1}^{\zeta} g^{\alpha_i\sum_{j=1}^{m}\beta_j B_{I(i)j}}\cdot\prod_{i=1}^{\zeta} g^{r_{I(i)}\alpha_i}\\
&=\prod_{i=1}^{\zeta} g^{\alpha_i\left(\sum_{j=1}^{m}\beta_j B_{I(i)j}+r_{I(i)}\right)}\\
&=\prod_{i=1}^{\zeta}\left(g^{1/s}\right)^{\alpha_i t_{I(i)}}\\
&=\left(g^{1/s}\right)^{\sum_{i=1}^{\zeta}\alpha_i t_{I(i)}}\\
&=\left(g^{1/s}\right)^{\pi}.
\end{aligned}
$$

8.4 Security Analysis

We show that the proposed scheme DPoS is STC secure. This provides security against duplicate data faking attacks where a malicious client tries to successfully upload a fake file for a valid tag.

Theorem 17. *DPoS is STC secure if the hash function H is collision resistant.*

Table 8.1: Theoretical comparison of proposed scheme with existing schemes

	Client's cost		Server's cost
Scheme	**Tag generation**	**Proof Verification**	**Proof generation**
Zheng *et al.* [38]	4nE + 2sn$M$ + 2nH	3mE + mM + 2P	3mM + mE
Yuan *et al.* [40]	nH + sn$E$ + 2sn$M$	mH + mM + 4P	sE + (2m+s)M + mE
DPoS	nH + nE + nM	mH + mM	mH + 2mM

1 n: number of blocks in the file, s: number of sectors in each block, m: number of challenge blocks

2 E: cost of one modular exponentiation, M: cost of one modular multiplication, H: cost of one hash computation, P: cost of one pairing operation

Proof. We show that if the adversary $\mathcal{A}$ in $\mathsf{Exp}^{\mathsf{STC}}_{\mathsf{DPoS},\mathcal{A}}(\lambda)$ is successful in duplicate data faking attack with non-negligible probability then it can find a collision for the hash function H with same non-negligible probability. Let $(t, \gamma, H(B))$ be a valid tag for the block B and an adversary $\mathcal{A}$ finds another block B' such that $B' \neq B$ and the $\mathsf{Exp}^{\mathsf{STC}}_{\mathsf{DPoS},\mathcal{A}}(\lambda)$ returns true for the block B'. This means $H(B) = H(B')$ and

$$\begin{aligned}\left(g^{1/s}\right)^t &= \gamma \cdot \prod_{j=1}^{m} \left(g^{\beta_j}\right)^{B'_j} \\ g^{\sum_{j=1}^{m}\beta_j B_j + r} &= g^r \cdot g^{\sum_{j=1}^{m}\beta_j B'_j} \\ \sum_{j=1}^{m} \beta_j B_j &= \sum_{j=1}^{m} \beta_j B'_j.\end{aligned}$$

Since $\{\beta_j\}_{j=1}^m$ are randomly selected from $\mathbb{Z}_q^\star$, the probability of $\sum_{j=1}^m \beta_j B_j = \sum_{j=1}^m \beta_j B'_j$ and $H(B) = H(B')$ for any two blocks is less than the probability of getting a collision for the hash function H. As the adversary $\mathcal{A}$ has non-negligible success probability, we have a collision for the hash function H with non-negligible probability. Since the hash function H is collision resistant, we conclude that there does not exist such adversary $\mathcal{A}$ and DPoS is STC secure. □

Now, we show that DPoS is unforgeable. In the proposed scheme, the probability of the CSS producing a valid integrity proof using a corrupt file is negligible. Any error in the file gets detected during integrity proof verification.

Theorem 18. *DPoS is UNF secure under DL assumption.*

Proof. We show that if the adversary $\mathcal{A}$ in $\mathsf{Exp}^{\mathsf{UNF\text{-}PoS}}_{\mathsf{DPoS},\mathcal{A}}(\lambda)$ is successful in modifying the file $F = B_1 B_2 \dots B_n$ to $F' = B'_1 B'_2 \dots B'_n$ with non-negligible probability then it can successfully find the secret value s corresponding to the file F. The $\mathsf{Exp}^{\mathsf{UNF\text{-}PoS}}_{\mathsf{DPoS},\mathcal{A}}(\lambda)$ returns 1 only if $F \neq F'$ and the proof is valid for the file F. Let F' be the modified file, (d', π') is integrity proof generated using F' for the challenge (x, I) and (d, π) is integrity proof generated using F for the challenge (x, I).

Case I: $d = d'$ or $\pi = \pi'$.

If $d = d'$, then for each $1 \leq j \leq m$, we have

$$\sum_{i=1}^{\zeta} \alpha_i \cdot B_{I(i)j} = \sum_{i=1}^{\zeta} \alpha_i \cdot B'_{I(i)j}$$

for all $\{\alpha_1, \alpha_2, \dots, \alpha_m\} \in (\mathbb{Z}_q^\star)^m$ and all $I \subset \{1, 2, \dots, n\}$. This gives $B_{I(i)j} = B_{I(i)j}$ for all $1 \leq i \leq n$ and $1 \leq j \leq m$. This gives $F = F'$ which contradicts the assumption that $F \neq F'$. Hence, $d = d'$ cannot be true.

If $\pi = \pi'$, then we have

$$\sum_{i=1}^{\zeta} \alpha_i t_{I(i)} = \sum_{i=1}^{\zeta} \alpha_i t'_{I(i)}$$

for all $\{\alpha_1, \alpha_2, \dots, \alpha_m\} \in (\mathbb{Z}_q^\star)^m$ and all $I \subset \{1, 2, \dots, n\}$. This gives $t_i = t'_i$ for all $i \in \{1, 2, \dots, n\}$. As $F \neq F'$, there is at least one $i \in \{1, 2, \dots, n\}$ such that $B_i \neq B'_i$. Since $t_i = t'_i$, we have $\sum_{j=1}^{m} \beta_j B_j = \sum_{j=1}^{m} \beta_j B'_j$. The probability of $\sum_{j=1}^{m} \beta_j B_j = \sum_{j=1}^{m} \beta_j B'_j$ for two different blocks B_i, B'_i is negligible, $1/q$, and hence, we cannot have $\pi = \pi'$ with non-negligible probability.

Case II: $d \neq d'$ and $\pi \neq \pi'$.

Since both (d, π) and (d', π') are valid proofs for the file F, we have

$$s \left(\sum_{j=1}^{m} \beta_j \cdot d_j + z \right) = \pi \text{ and } s \left(\sum_{j=1}^{m} \beta_j \cdot d'_j + z \right) = \pi'$$

where $z = \sum_{i=1}^{m} \alpha_i \cdot f(\mathsf{filename}, s, I(i))$. By subtracting one from other, we get

$$s \left(\sum_{j=1}^{m} \beta_j \cdot d_j - \sum_{j=1}^{m} \beta_j \cdot d'_j \right) = \pi - \pi'$$

$$s = \frac{\pi - \pi'}{\left(\sum_{j=1}^{m} \beta_j \cdot d_j - \sum_{j=1}^{m} \beta_j \cdot d'_j\right)}.$$

We note that the denominator can still be zero even though $d \neq d'$. However, the probability of $\sum_{j=1}^{m} \beta_j \cdot d_j = \sum_{j=1}^{m} \beta_j \cdot d'_j$ is $1/q$ which is negligible for a large prime q. Since the adversary $\mathcal{A}$ knows both (d, π) and (d', π'), it can easily compute the secret s with non-negligible advantage. Given public value $g^{1/s}$, finding the secret s is equivalent to solving discrete logarithm problem (DLP). Since it is computationally hard to solve DLP as per DL assumption, we conclude that there does not exist such adversary $\mathcal{A}$ and DPoS is UNF secure. □

8.5 Performance Analysis

To analyze our scheme analytically, we compare Tag generation, Proof generation, and Proof verification parts of our scheme with existing schemes. To make the comparison authentic and as close as possible, we consider only those papers which proposed a proof of storage with deduplication schemes. Table 8.1 illustrates the comparison between the proposed scheme, Zheng *et al.*' scheme, and Yuan *et al.* 's scheme. From the Table 8.1, it is evident that the proposed

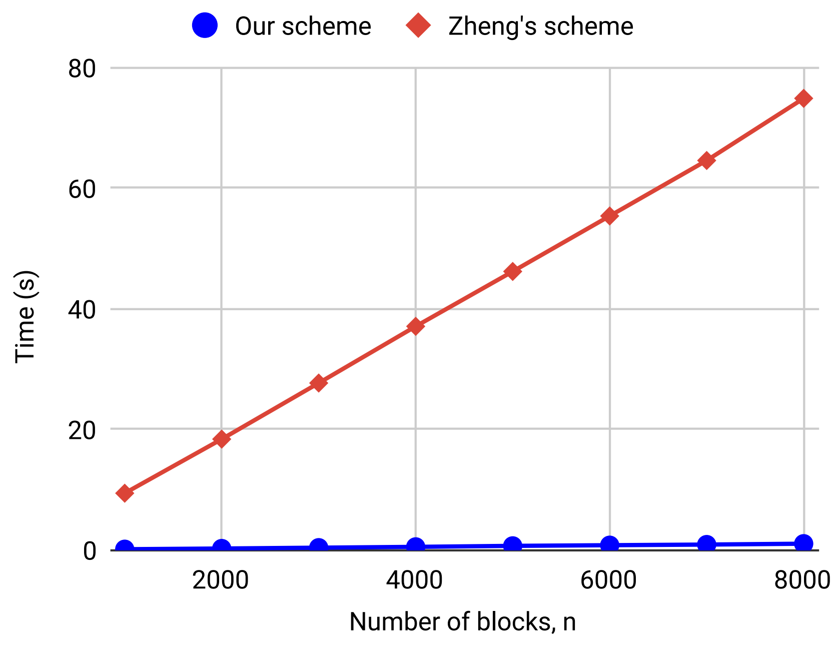


Figure 8.2: Tag generation cost vs number of blocks

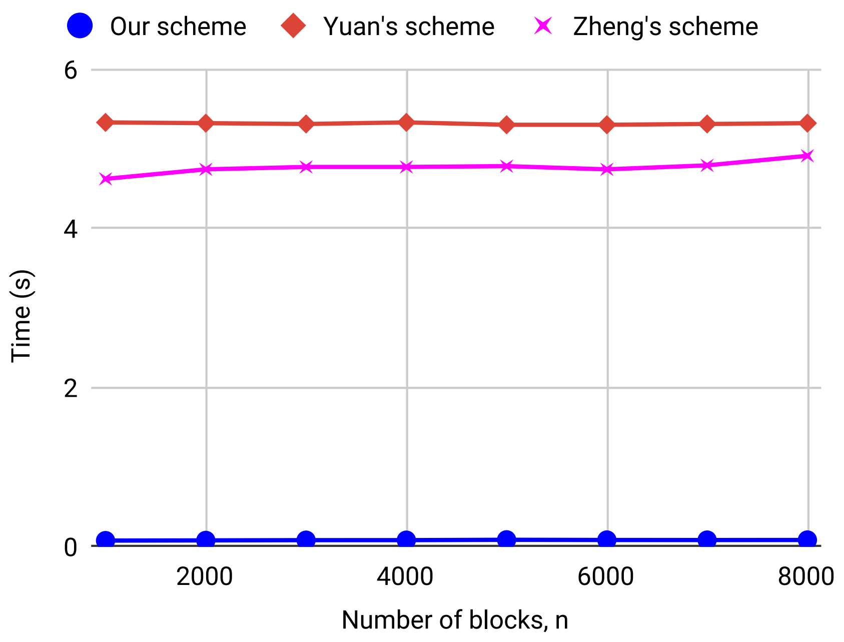


Figure 8.3: Proof generation cost vs number of blocks

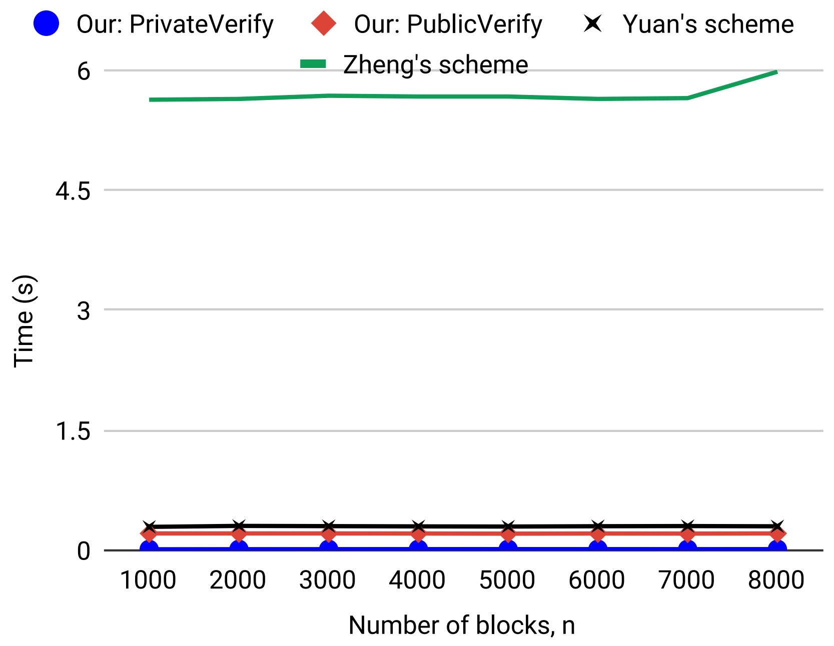


Figure 8.4: Verification cost vs number of blocks

scheme takes much less time for verification of the proof as it does not require any pairing operation. The verification is usually done by the client, which may have a resource-constrained device. It is important to have the verification cost as low as possible. The client also does tag generation for a file. In our proposed scheme, the theoretical cost of the tag generation is on par with other schemes and can be analyzed more accurately by implementing all these schemes in the same environment. The proof generation part in our scheme does not require any exponentiation as compared to other schemes. This means the cost of proof generation is also lesser than that of the existing schemes. We further compare these schemes more accurately by implementing all on the same platform with the same parameters.

8.5.1 Experimental Results

We implement DPoS scheme along with Yuan's scheme and Zheng's scheme in a pc with following specifications: Ubuntu 18.04.2 LTS operating system, 4 GB memory, i5 − 6500 processor @ 3.2 GHz and Sage 8.1 [66].

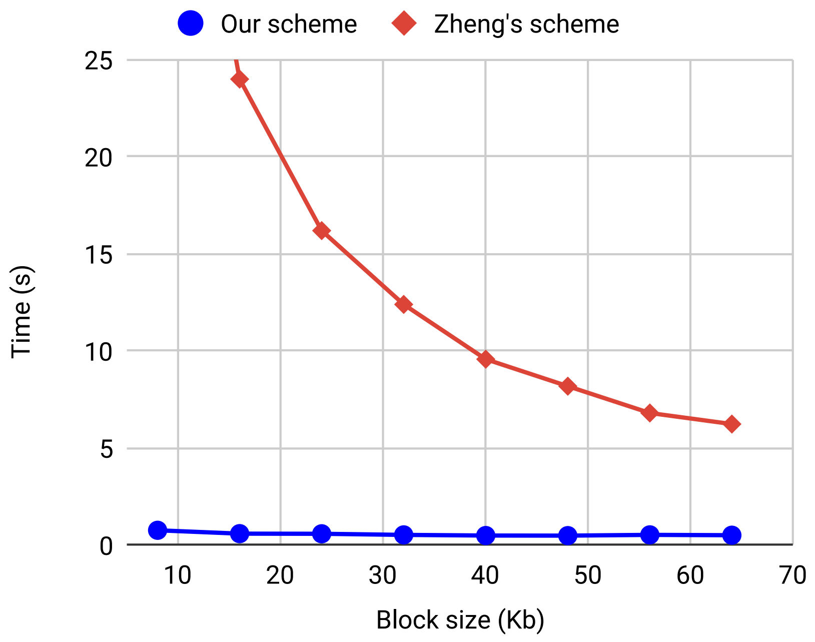


Figure 8.5: Tag generation cost vs block size

We have taken two scenarios. In the first scenario, we kept the block size fixed while changing the number of blocks. This eventually changes the size of the file. We have considered a block size of 8 Kb, and the number of blocks varies between 1000 to 8000. In our scheme, we are breaking our blocks into sectors of size 1024 bits. For other two schemes, the size of each sector depends on the size

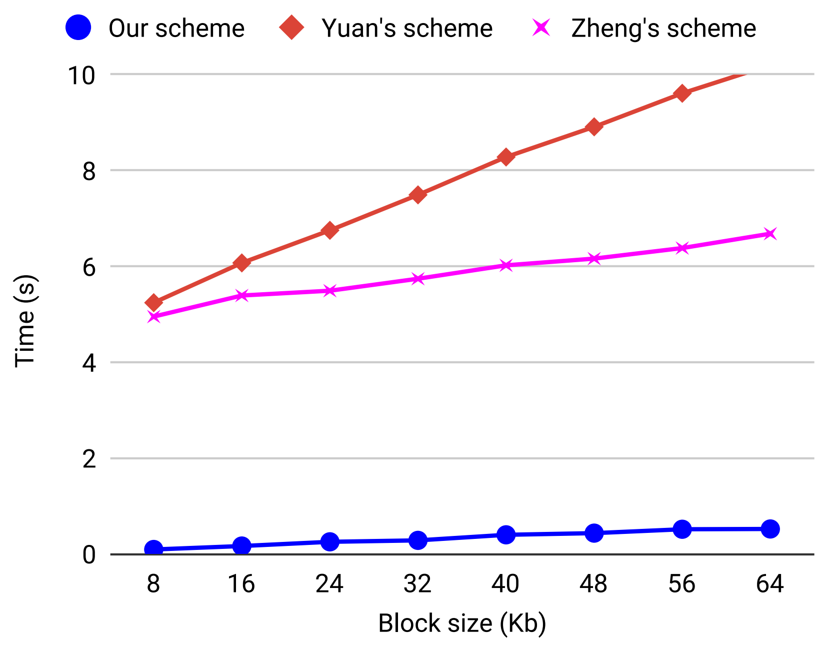


Figure 8.6: Proof generation cost vs block size

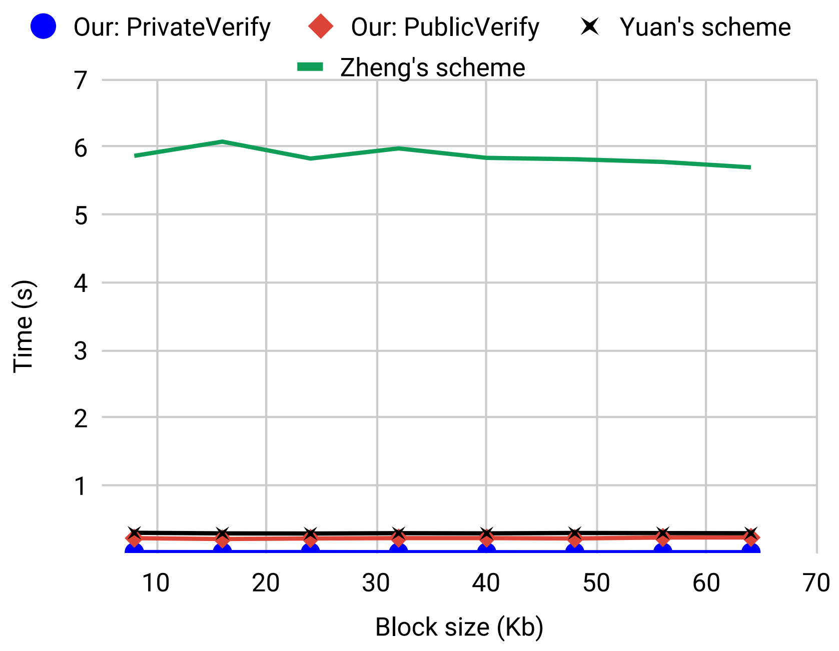


Figure 8.7: Verification cost vs block size

of the elliptic curve group used. We used standard elliptic curve `secp256k1` from Standards for Efficient Cryptography 2 (SEC 2) [64]. This sets the size of each sector to be 256 bits and 32 sectors in each block for the other schemes. For $\lambda = 80$ bit security, we selected a 1024 bit prime as q for our scheme.

Figure 8.2, 8.3, 8.4 shows cost comparison of all three schemes for tag generation, proof generation and proof verification process. Since Tag generation for Yuan's scheme is taking more than 500 seconds for each case, we have not included it in the graphical representation. Moreover, the tag generation is a one time cost for each file but data integrity check are often done frequently. The proof generation and verification is required to be an efficient. The tag generation and proof verification are done on the client-side while proof generation is done on the server-side. In all the three processes, our scheme takes less time as compared to the other two schemes. For tag, generation, our scheme takes less than 1 seconds for up to 8000 blocks. For the challenge process, it is considered that 460 blocks are enough for 99% detection rate when there is enough blocks and the probability of data corruption is 0.01 [40, 32]. In our experiment, the number of challenge blocks is 460. As the number of challenege blocks is fixed, the proof generation and verification cost for each scheme remains constant as the file size increases. However, the cost for proof generation in our scheme is less than 0.1 second and for other two schemes, it is around 5 seconds. For the proof verification process, private verification of our scheme is better than Zheng's scheme by a factor of at least 1000. This is significant and important as verification of proof is done by the client. The public verification cost of our scheme is comparable but less than the verification cost of Yuan's scheme.

In the second scenario, we fixed a file size of 5 MB with different block size. We kept the rest of the parameters the same as in the first experiment. Since the file size is the same and the block size is different, it changes the number of blocks as well. For this experiment, we kept the number of challenge blocks equal to 460. We considered block sizes of 8 Kb to 64 Kb. The cost of tag generation for Yuan's scheme is more than 1000 seconds for each case. We did not put it on the graph as it is already very high. As shown in the Figure 8.5, the cost of tag generation decreases as block size increases. However, it is negligible for our scheme as compared to the other two schemes. As shown in the Figure 8.6, the cost of proof generation increases as we increase the block size, and it is significantly lower for our scheme for all block sizes. Even though the cost of proof verification is constant and lower for our scheme and Yuan's scheme as shown in the Figure 8.7, private verification of our scheme is still better than Yuan's scheme by a factor of around 25.

8.6 Conclusion

This chapter presented an efficient proof of storage with data deduplication at the block-level scheme, DPoS. The design of the DPoS is based on the idea of the $\mathsf{VIP} - \mathsf{POPE}$ scheme. In the $\mathsf{VIP} - \mathsf{POPE}$ scheme, we are verifying the computation over encrypted data. In DPoS, we consider the computation of proof of storage as computation over data. If data gets modified, then the verification of computation fails. Therefore, verification of the integrity of the data is equivalent to verification of computation.

The DPoS scheme does not require any additional parameters for the data deduplication process. The verification parameters used in the proof of storage is used for the data deduplication process. Moreover, it is secure against duplicate faking attacks and forgery attacks. We implemented DPoS with existing similar schemes and observed that the proposed scheme DPoS is efficient and practical compared to other schemes.

CHAPTER 9

Conclusion and Future Work

Verification of computation and integrity of stored data are important security concerns in cloud-based services. Verification of computation is challenging, especially when the logic of computation is hidden from users. While designing schemes related to these problems, one needs to consider several other features like data privacy, user's privacy, efficient storage system, etc. based on the application. In this thesis, we presented our contribution to verifiable cloud computing and storage services. We have formally defined verifiable private polynomial evaluation and IND-CFA, polynomial protection PP, and unforgeability security models. We presented the first verifiable private polynomial evaluation scheme PIPE, which is secure under standard security assumptions. We have provided detailed security proofs for PIPE.

We provided a verifiable oblivious polynomial evaluation $\mathsf{VIP}-\mathsf{POPE}$ scheme for healthcare-related applications, where users encrypt data before sending it to the cloud. We formally proved IND-CFA, CPI, and unforgeability properties of *pipetwo*. We compared the proposed scheme with existing related schemes by implementing all the schemes on the same platform. We observed that our proposed scheme $\mathsf{VIP}-\mathsf{POPE}$ outperforms other schemes significantly. To provide privacy-preserving identification and verifiable computation in a single scheme, we designed PriVC. The PriVC scheme verifies computation over encrypted data, privacy-preserving user authentication, and user undeniability. We provided IND-CFA, unforgeability, and non-repudiation security proofs. We have shown that the proposed scheme PriVC is efficient by implementing our scheme with realistic parameters.

We presented an efficient proof of storage scheme with block-level data deduplication for data integrity in cloud storage service. We have shown that the proposed scheme is secure against duplicate faking attacks and proof forgery attacks. We proved our scheme's efficiency by implementing our scheme and existing related schemes on the same platform. We observed that our proposed scheme is significantly better than the rest of the scheme.

The thesis work opens up some future scope, verifying computation with a multi-variable polynomial in a similar scenario. Inputs to the multi-variable polynomial may come from the same user or different users. It will be more challenging to preserve data privacy and verify the computation simultaneously when inputs to the polynomial come from two different users. Consider a scenario with two variable polynomial $f(x, y)$. One user sends x_0, and another user sends y_0 to the cloud. The cloud computes $f(x_0, y_0)$ along with two separate proof of computation, π_1 and π_2. The cloud sends $(f(x_0, y_0), \pi_1)$ to first user and $(f(x_0, y_0), \pi_2)$ to second user. Each user does not know anything about the input of other users, however, they still need to verify the computation. Another scope is in the area of proof of storage with data deduplication over encrypted data. Our proposed scheme DPoS considers plain data. Data deduplication over encrypted data requires a convergent encryption method to ensure that each data owner can decrypt the data. However, existing proof of storage schemes does not work in such a scenario. The challenge process is proof of storage involves a homomorphic authenticator. Convergent encryption does not have homomorphic property as it encrypts each data block with a different key, which can be challenging to address.

References

[1] Gantz, J., Reinsel, D.: The digital universe decade–are you ready(2010). http://www.emc.com/collateral/analyst-reports/idc-digital-universe-are-you-ready.pdf, last accessed on 16th August, 2019.

[2] Baudron, O., Fouque, P., Pointcheval, D., Stern, J., Poupard, G.: Practical multi-candidate election system. In Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing (PODC), Newport, Rhode Island, USA, pp. 274–283, 2001.

[3] Bellare, M., Boldyreva, A., Micali, S.: Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements. In Proceedings of Advances in Cryptology - EUROCRYPT, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, pp. 259–274, 2000.

[4] Bultel, X., Das, M.L., Gajera, H., Gerault, D., Giraud, M., Lafourcade, P.: Verifiable Private Polynomial Evaluation. In Proceedings of Provable Security - 11th International Conference (ProvSec), Xi'an, China, pp. 487–506, 2017.

[5] Canetti, R., Riva, B., Rothblum, G.N.: Two Protocols for Delegation of Computation. In Proceedings of Information Theoretic Security - 6th International Conference (ICITS), Montreal, QC, Canada, pp. 37–61, 2012.

[6] Choi, S.G., Katz, J., Kumaresan, R., Cid C.: Multi-Client Non-interactive Verifiable Computation. In Proceedings of Theory of Cryptography - 10th Theory of Cryptography Conference (TCC), Tokyo, Japan, pp. 499–518, 2013.

[7] Diffie, W., Hellman, M.E.: New directions in cryptography. In IEEE Transactions on Information Theory, 22(6), pp. 644–654, November 1976.

[8] Feldman, P.: A Practical Scheme for Non-interactive Verifiable Secret Sharing. In Proceedings of 28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, pp. 427–437, 1987.

[9] Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In Proceedings of Advances in Cryptology - CRYPTO, Santa Barbara, California, USA, pp. 186–194, 1986.

[10] Fiore, D., Gennaro, R.: Publicly verifiable delegation of large polynomials and matrix computations, with applications. In Proceedings of the ACM Conference on Computer and Communications Security (CCS), Raleigh, NC, USA, pp. 501–512, 2012.

[11] Freedman, M.J., Ishai, Y., Pinkas, B., Reingold, O.: Keyword Search and Oblivious Pseudorandom Functions. In Proceedings of Second Theory of Cryptography Conference (TCC), Cambridge, MA, USA, pp. 303–324, 2005.

[12] Freedman, M.J., Nissim, K., Pinkas, B.: Efficient Private Matching and Set Intersection. In Proceedings of Advances in Cryptology- EUROCRYPT, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, pp. 1–19, 2004.

[13] Gennaro, R., Gentry, C., Parno, B.: Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In Proceedings of Advances in Cryptology - CRYPTO, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, pp. 465–482, 2010.

[14] Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-Size Commitments to Polynomials and Their Applications. In Proceedings of Advances in Cryptology - ASIACRYPT - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, pp. 177–194, 2010.

[15] Lindell, Y., Pinkas, B.: Privacy Preserving Data Mining. In Proceedings of Advances in Cryptology - CRYPTO, 20th Annual Cryptology Conference, Santa Barbara, CA, USA, pp. 36–54, 2000.

[16] Naor, M., Pinkas, B.: Oblivious Transfer and Polynomial Evaluation. In Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, Atlanta, Georgia, USA, pp. 245–254, 1999.

[17] Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Proceedings of Advances in Cryptology - EUROCRYPT, 17th international conference on Theory and application of cryptographic techniques, Prague, Czech Republic, pp. 223–238, 1999.

[18] Papamanthou, C., Shi, E., Tamassia, R.: Signatures of Correct Computation. In Proceedings of Theory of Cryptography - 10th Theory of Cryptography Conference (TCC), Tokyo, Japan, pp. 222–242, 2013.

[19] Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly Practical Verifiable Computation. In Proceedings of IEEE Symposium on Security and Privacy (SP), Berkeley, CA, USA, pp. 238–252, 2013.

[20] Parno, B., Raykova, M., Vaikuntanathan, V.: How to delegate and verify in public: Verifiable computation from attribute-based encryption. In Proceedings of Theory of Cryptography - 9th Theory of Cryptography Conference (TCC), Taormina, Sicily, Italy, pp. 422–439, 2012.

[21] Pedersen, T.P.: Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In Proceedings of Advances in Cryptology - CRYPTO, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, pp. 129–140, 1991.

[22] Fersh, M., Kiltz, E., Poettering, B.: On the Provable Security of (EC)DSA Signatures. In Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS), Vienna, Austria, pp. 1651–1662, 2016.

[23] Xia, Z., Yang, B., Zhang, M., Mu, Y.: An Efficient and Provably Secure Private Polynomial Evaluation Scheme. In Proceedings of Information Security Practice and Experience - 14th International Conference (ISPEC) Tokyo, Japan, pp. 595–609, 2018.

[24] Douceur, J.D., Adya, A., Bolosky, W.J., Simon, D., Theimer, M.: Reclaiming space from duplicate files in a server-less distributed file system. In Proceedings of IEEE International Conference on Distributed Computing System, Vienna, Austria, pp. 617–624, 2002.

[25] Bellare, M., Keelveedhi, S., Ristenpart, T.: DupLESS: Server-aided encryption for deduplicated storage. In Proceedings of USENIX Security Symposium, Washington, DC, USA, pp. 179–194, 2013.

[26] Bellare, M., Keelveedhi, S., Ristenpart, T.: Message-locked encryption and secure deduplication. In Proceedings of Advances in Cryptology — EUROCRYPT, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, pp. 296–312, 2013.

[27] Abadi, M., Boneh, D., Mironov, I., Raghunathan, A., Segev, G.: Message-locked encryption for lock-dependent messages. In Proceedings of Advances in Cryptology — CRYPTO, 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, pp. 374–391, 2013.

[28] Bellare, M., Keelveedhi, S.: Interactive message-locked encryption and secure deduplication. In Proceedings of 18th IACR International Conference on Practice and Theory in Public-Key Cryptography (PKC), Gaithersburg, MD, USA, pp. 516–538, 2015.

[29] Jiang, T., Chen, X., Wu, Q., Ma, J., Susilo, W., Lou, W.: Secure and Efficient Cloud Data Deduplication With Randomized Tag. In IEEE transactions on information forensics and security, 12(3), pp. 532–543, October 2016.

[30] Garfinkel, S.L.: Public key cryptography. In IEEE Journal of Computer, 29(6), pp. 101–104, June 1996.

[31] Storer, M.W., Greenan, K., Long, D.D.E., Miller, E.L.: Secure data deduplication. In Proceedings of 4th ACM International Workshop on Storage security and survivability, Alexandria, Virginia, USA, pp. 1–10, 2008.

[32] Ateniese, G., Burns, R., Herring, J.: Provable data possession at untrusted stores. In proceeding of the 14th ACM Conference on Computer and Communications Security (CCS), pp. 598–609, 2007.

[33] Mukundan, R., Madria, S., Linderman, M.: Efficient integrity verification of replicated data in cloud using homomorphic encryption. In Springer Journal of Distributed and Parallel Databases, 32(4), pp. 507–534, December 2014.

[34] Lin, C., Shen, Z., Chen, Q., Sheldon, F.: A data integrity verification scheme in mobile cloud computing. In journal of Network and Computer Applications, 77, pp. 146–151, January 2017.

[35] Wang, Y., Wu, Q., Qin, B., Tang, S., Susilo, W.: Online/Offline Provable Data Possession. In IEEE Transactions of Information Forensic and Security, 12(5), pp. 1182–1194, January 2017.

[36] Li, J., Tan, X., Chen, X., Wong, D.S., Xhafa, F.: OPoR: Enabling Proof of Retrievability in Cloud Computing with Resource-Constrained Devices. In IEEE Transactions on Cloud Computing, 3(2), pp. 195–205, October 2014.

[37] Tan, C.B., Hanafi, M., Hijazi, A., Lim, Y., Gani, A.: A survey on Proof of Retrievability for cloud data integrity and availability: Cloud storage state-of-the-art, issues, solutions and future trends. In Journal of Network and Computer Applications, 110, pp. 75–86, May 2018.

[38] Zheng, Q., Xu, S.: Secure and efficient proof of storage with deduplication. In Proceedings of the second ACM conference on Data and Application Security and Privacy (CODASPY), San Antonio, Texas, USA, pp. 1–12, 2012.

[39] Shin, Y., Koo, D., Hur, J., Yun, J.: Secure proof of storage with deduplication for cloud storage systems. In Multimedia Tools and Applications, Springer, 76(19), pp. 19363–19378, October 2017.

[40] Yuan, J., Yu, S.: Secure and Constant Cost Public Cloud Storage Auditing with Deduplication. In Proceedings of IEEE conference on Communications and Network Security, National Harbor, MD, USA, pp. 145–153, 2013.

[41] Vasilopoulos, D., Onen, M., Elkhiyaoui, K., Molva, R.: Message-Locked Proofs of Retrievability with Secure Deduplication. In Proceedings of the 2016 ACM on Cloud Computing Security Workshop, Vienna, Austria, pp. 73–83, 2016.

[42] He, K., Chen, J., Du, R., Wu, Q., Xue, G., Zhang, X.: DeyPoS: Deduplicatable Dynamic Proof of Storage for Multi-User Environments. In IEEE Transactions on Computers, 65(12), pp. 3631–3645, April 2016.

[43] Juels, A., Kaliski, B.S.: Pors: proofs of retrievability for large files. In Proceedings of the 14th ACM conference on Computer and communications security (CCS), Alexandria, Virginia, USA, pp. 584–597, 2007.

[44] Shawish, A., Salma, M.: Cloud Computing: Paradigms and Technologies. In Inter-cooperative Collective Intelligence: Techniques and Applications, 496, pp. 39–67, August 2013.

[45] Guo, L., Zhang, C., Yue, H., Fang, Y.: PSaD: A privacy-preserving social-assisted content dissemination scheme in DTNs. In IEEE Transactions on Mobile Computing, 13(12), pp. 2903–2918, February 2014.

[46] Dropbox, accessed on Apr. 15, 2020. [Online]. Available: https://www.dropbox.com/

[47] Google. Google Drive, accessed on Apr. 15, 2020. http://drive.google.com

[48] NetApp. Universal Storage System, accessed on Apr. 15, 2020. [Online]. Available: http://www.netapp.com/us/products/platform-os/dedupe.aspx

[49] Guo, L., Fang, Y., Li, M., Li, P.: Verifiable Privacy-preserving Monitoring for Cloud-assisted mHealth Systems. In Proceedings of IEEE Conference on Computer Communications (INFOCOM), Kowloon, Hong Kong, pp. 1026–1034, 2015.

[50] Pollard, J.: A monte carlo method for index computation (mod p). In *Mathematics of Computation*, volume 32, pp. 918–924. Springer, 1978.

[51] Gajera, H., Naik, S., Das M.L.: On the security of "Verifiable privacy-preserving monitoring for cloud-assisted mHealth systems". In Proceedings of International Conference on Information Systems Security (ICISS), Jaipur, India, pp. 324–335, 2016.

[52] Jones, R.: Nasty Adobe Bug Deleted $250,000-Worth of Man's Files, Lawsuit Claims. https://gizmodo.com/nasty-adobe-bug-deleted-250-000-worth-of-mans-files-l-1830405390, last accessed on August 18, 2019.

[53] Raczkowski Paruch: Cloud computing in Poland. https://www.lexology.com/library/detail.aspx?g=396a301b-6fea-49b5-ab68-abf3c5041707, last accessed on August 18, 2019.

[54] Mohan, P., Sultan, S.: MediNet: A mobile healthcare management system for the Caribbean region. In Proceedings of International Conference of Mobile and Ubiquitous Systems: Networking & Services, Toronto, ON, Canada, pp. 1–2, 2009.

[55] Liu, C.H., Wen, J., Yu, Q., Yang, B., Wang, W.: HealthKiosk: A family-based connected healthcare system for long-term monitoring. In Proceedings of IEEE Conference on Computer Communications Workshops, Shanghai, China, pp. 241–246, 2011.

[56] Klasnja, P., Pratt, W.: Healthcare in the pocket: Mapping the space of mobile-phone health interventions. In Journal of Biomedical Informatics, 45(1), pp. 184–198, February 2012.

[57] Chiarini, G., Ray, P., Akter, S., Masella, C., Ganz, A.: mHealth technologies for chronic diseases and elders: A systematic review. In IEEE Journal on Selected Areas in Communications, 31(9), pp. 6–18, August 2013.

[58] Pisa, P. S., Abdalla, M., Duarte, O. C. M. B.: Somewhat homomorphic encryption scheme for arithmetic operations on large integers. In Proceedings of Global Information Infrastructure and Networking Symposium (GIIS), Choroni, Venezuela, pp. 1–8, 2012.

[59] Chenal, M., Tang, Q.: On Key Recovery Attacks Against Existing Somewhat Homomorphic Encryption Schemes. In Proceedings of 3rd International Conference on Cryptology and Information Security in Latin America (LATINCRYPT 2014), Florianópolis, Brazil, pp. 239–258, 2014.

[60] Lepoint, T., Tibouchi, M.: Cryptanalysis of a (Somewhat) Additively Homomorphic Encryption Scheme Used in PIR. In Proceedings of International Conference on Financial Cryptography and Data Security, San Juan, Puerto Rico, pp. 184–193, 2015.

[61] Knorr, E.: What is cloud computing? Everything you need to know now. https://www.infoworld.com/article/2683784/what-is-cloud-computing.html, last accessed on August 18, 2019.

[62] Pearson, S., Benameur, A.: Privacy, Security and Trust Issues Arising from Cloud Computing. In Proceedings of IEEE Second International Conference on Cloud Computing Technology and Science, Indianapolis, IN, USA, pp. 693–702, 2010.

[63] https://aws.amazon.com/agreement/, last accessed on August 18, 2019.

[64] Certicom Research: Standards for Efficient Cryptography 2 (SEC 2). http://www.secg.org/sec2-v2.pdf, last accessed on August 18, 2019.

[65] Li, Y., Yu, Y., Min, G., Susilo, W., Ni, J., Choo, K. K. R.: Fuzzy identity-based data integrity auditing for reliable cloud storage systems. In IEEE Transactions on Dependable and Secure Computing, vol 16(1), pp. 72–83, Feb 2017.

[66] https://www.sagemath.org/, last accessed on August 18, 2019.

[67] Amin, R., Islam, S. K. H., Biswas, G. P., Khan, M. K., Kumar, N.: A robust and anonymous patient monitoring system using wireless medical sensor networks. In Future Generation Computer Systems, 80, pp. 483–495, March 2018.

[68] De Muth, J. E.: Basic statistics and pharmaceutical statistical applications. Chapman and Hall/CRC, 2014.

[69] Personal info of 1.5m SingHealth patients, including PM Lee, stolen in Singapore's worst cyber attack. https://www.straitstimes.com/singapore/personal-info-of-15m-singhealth-patients-including-pm-lee-stolen-in-singapores-most, last accessed on August 2018, 2019.

[70] ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In IEEE Transactions on Information Theory, 31(4), pp. 469–472, July 1985.

[71] Online Statistics Education: A Multimedia Course of Study (http://onlinestatbook.com/). Project Leader: David M. Lane, Rice University.

[72] Van Dijk, M., Gentry, C., Halevi, S., and Vaikuntanathan, V.:Fully homomorphic encryption over the integers. In proceedings of 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), Monaco and Nice, French Riviera, pp. 24–43, 2010.

[73] Basu, A., Sengupta, I., and Singh, J. K.: Secured cloud storage scheme using ECC based key management in user hierarchy. In Proc. of the International Conference on Information Systems Security (ICISS 2011), LNCS 7093, Springer, pp. 175–189, 2011.

[74] Lin, H., Shao, J., Zhang, C., and Fang,Y.: CAM: Cloud-assisted privacy preserving mobile health monitoring. In IEEE Transactions on Information Forensics and Security, 8(6):985–997, 2013.

[75] Wang, W., Li, Z., Owens, R., and Bhargava, B.: Secure and efficient access to outsourced Data. In Proceedings of the ACM workshop on Cloud Computing Security (CCSW 2009), pp. 55–66, 2009.

[76] Katz, J., Yehuda, L.: Introduction to Modern Cryptography. CRC Press, 2007.

[77] Bellare, M.: A Note on Negligible Functions. https://eprint.iacr.org/1997/004.pdf

[78] Chaum, D., Pedersen, T. P.: Wallet Databases with Observers. In Proceedings of Annual International Cryptology Conference (CRYPTO), Santa Barbara, California, USA, pp. 89–105, 1992.

[79] Meadows, C.A.: Formal verification of cryptographic protocols: A survey. In Proceedings of Advances in Cryptology — ASIACRYPT, 4th International Conferences on the Theory and Applications of Cryptology Wollongong, Australia, pp. 133–150, 1994.

[80] Blanchet, B.: CryptoVerif: A computationally-sound security protocol verifier, 2017, [online] Available: http://cryptoverif.inria.fr/cryptoverif.pdf.

[81] Blanchet, B., Abadi, M., Fournet, C.: Automated verification of selected equivalences for security protocols. In The Journal of Logic and Algebraic Programming, 75(1), pp. 3–51, Feb-Mar 2008.

[82] Ateniese, G., Di Pietro, R., Mancini L. V., and Tsudik, G.: Scalable and efficient provable data possession. In proceeding of the 4th International Conference on Security and privacy in communication networks, Istanbul, Turkey, pp. 1–10, 2008.

[83] Hur, J., Koo, D., Shin, Y., and Kang, K.: Secure Data Deduplication with Dynamic Ownership Management in Cloud Storage. In IEEE Transactions on Knowledge and Data Engineering, 28(11), pp. 3113–3125, Jun 2016.

[84] Ng, W. K., Wen, Y., and Zhu, H.,: Private data deduplication protocols in cloud storage. In Proceeding of the 27th Annual ACM Symposium on Applied Computing (SAC'12), Trento, Italy, pp. 441–446, 2012.

[85] Ding, W., Yan, Z., and Deng, R. H.: Secure Encrypted Data Deduplication with Ownership Proof and User Revocation. In Proceeding of the International Conference on Algorithms and Architectures for Parallel Processing, Helsinki, Finland, pp. 297–312, 2017.

[86] Okayama, T.: Future gardening systemsmart garden. In Journal of Developments in Sustainable Agriculture, 9(1), pp. 47–50, 2014.

[87] Lloret, J., Garcia, M., Bri, D., and Sendra, S.: A wireless sensor network deployment for rural and forest fire detection and verification. In Sensors, 9(11), pp. 8722–8747, 2009.

Appendix A : PolyCommit$_{\mathsf{Ped}}$ Scheme

We recall the PolyCommit$_{\mathsf{Ped}}$ construction presented by Kate *et al.* [14].

Definition 34. $\mathsf{PolyCommit}_{\mathsf{Ped}} = (\mathsf{setup}, \mathsf{init}, \mathsf{compute}, \mathsf{verif})$ *is a* PPE *scheme defined as follows:*

$\mathsf{setup}(\lambda)$: *Using the security parameter λ, it generates two groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p (providing λ-bit security) such that there exists a symmetric bilinear pairing $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Moreover, it chooses two generators g and h of $\mathbb{G}$ and picks $\alpha \leftarrow \mathbb{Z}_p^*$. It sets $F = \mathbb{Z}_p^*$, $\mathsf{pub} = (\mathbb{G}, \mathbb{G}_T, p, e, g, h, (g^{\alpha}, \ldots, g^{\alpha^k}), (h^{\alpha}, \ldots, h^{\alpha^k}))$ and returns (pub, F).*

$\mathsf{init}(\mathsf{pub}, f)$: *Using $f(x) = \sum_{i=0}^{k} a_i \cdot x^i$, this algorithm chooses a random polynomial of degree k, $r(x) = \sum_{i=0}^{k} r_i \cdot x^i \in \mathbb{Z}_p[x]$ and sets $\mathsf{sk} = r(x)$. It computes $\mathcal{C} = \prod_{i=0}^{k} (g^{\alpha^i})^{a_i} (h^{\alpha^i})^{r_i} = g^{f(\alpha)} h^{r(\alpha)}$ and sets $\mathsf{vk} = \mathcal{C}$. Finally, it returns $(\mathsf{sk}, \mathsf{vk})$.*

$\mathsf{compute}(\mathsf{pub}, \mathsf{vk}, x_i, \mathsf{sk}, f)$: *This algorithm computes $\psi_i(x) = (f(x) - f(x_i))/(x - x_i)$ and $\hat{\psi}_i(x) = (r(x) - r(x_i))/(x - x_i)$. Let $(\gamma_0, \ldots, \gamma_k)$ and $(\hat{\gamma}_0, \ldots, \hat{\gamma}_k)$ be such that $\psi_i(x) = \sum_{j=0}^{k} \gamma_j \cdot x^j$ and $\hat{\psi}_i(x) = \sum_{j=0}^{k} \hat{\gamma}_j \cdot x^j$. It computes $w_i = \prod_{j=0}^{k} (g^{\alpha^j})^{\gamma_j} (h^{\alpha^j})^{\hat{\gamma}_j} = g^{\psi_i(\alpha)} h^{\hat{\psi}_i(\alpha)}$. It sets $\pi = (x_i, r(x_i), w_i)$ and returns $(f(x_i), \pi)$.*

$\mathsf{verif}(\mathsf{pub}, \mathsf{vk}, x_i, f(x_i), \pi)$: *If $e(\mathcal{C}, g)$ equals to $e(w_i, (g^{\alpha})^{-x_i}) e(g^{f(x_i)} h^{r(x_i)}, g)$, the algorithm outputs 1, else it outputs 0.*

Appendix B : Publications from the thesis

Published:

1. Gajera, H., Naik, S., Das, M. L.: On the security of "verifiable privacy-preserving monitoring for cloud-assisted mhealth systems". In Proceedings of International Conference on Information Systems Security (ICISS), pp. 324–335, 2016.

2. Bultel, X., Das, M. L., Gajera, H., Gerault, D., Giraud, M., Lafourcade, P.: Verifiable private polynomial evaluation. In Proceedings of International Conference on Provable Security (ProvSec 2017), pp. 487–506, 2017.

3. Gajera, H., Naik, S., Das, M. L.: MedCop: Verifiable Computation for Mobile Healthcare System. In Proceedings of International Symposium on Security in Computing and Communication (SSCC), pp. 471–482, 2018.

4. Gajera, H., Giraud, M., Gerault, D., Das, M. L., Lafourcade, P.: Verifiable and Private Oblivious Polynomial Evaluation. In Proceedings of International Conference on Information Security Theory and Practice (IFIP 2019), pp. 49–65, 2019.

5. Gajera, H., Das, M. L.: Privc: Privacy Preserving Verifiable Computation. In Proceedings of International Conference on COMmunication Systems & NETworkS (COMSNETS), pp. 298–305, 2020.

6. Gajera, H., Das, M. L.: DeDOP: Deduplication with cross-server Ownership over encrypted data. In Proceedings of Third ISEA Conference on Security and Privacy (ISEA-ISAP), pp. 36–40, 2020.

7. Gajera, H., Das, M. L.: Fine-grained Data Deduplication and proof of storage Scheme in Public Cloud Storage. In Proceedings of International Conference on COMmunication Systems & NETworkS (COMSNETS), pp. 237–241, 2021.