JAMES COOK UNIVERSITY AUSTRALIA
College of Science and Engineering

Pre-Distributed Information in Secure Multi-Party Computation

by

Amirreza Hamidi

A Thesis Submitted
in Fulfillment of the
Requirements for the Degree

Doctor of Philosophy

Townsville, Australia

Dissertation directed by Associate Professor Hossein Ghodosi
Discipline of Information Technology

2023

Certificate of Authorship/Originality

I certify that the work in this thesis has not been previously submitted for a degree nor has it been submitted as a part of the requirements for other degree except as fully acknowledged within the text.

I also certify that this thesis has been written by me the author. Any help that I have received in my research and in the preparation of the thesis itself has been fully acknowledged. In addition, I endorse that all information sources and literature used are quoted in the thesis.

Dedication

To the last rider, may the glory and victory be with you.

Acknowledgements

First of all, I would like to thank my primary supervisor, Hossein Ghodosi, for his guidance, advice and support during my PhD program at James Cook University. He helped me improve my writing which was not possible to achieve without his assistance.

I also would like to express my gratitude to the other member of my advisory panel, Shaun Belward, who has kindly watched my progress.

Thanks to Louis Cianciullo, a senior PhD graduate, who sometimes gave me helpful technical support.

Last but not least, I would like to express many thanks to my Mum and Dad who always remotely have been encouraging and supporting me to finish this long PhD journey while they were too far from me!

List of Publications and Contribution of Others

The contributions of the following publications correspond to the chapters 2-7 of this thesis. The author contributed to the bulk of this research, with help and guidance from associate professor Hossein Ghodosi.

[1] **Hamidi, Amirreza**, and Hossein Ghodosi. "Secure Multi-Party Computation Using Pre-distributed Information from an Initializer." Security and Privacy: Second International Conference, ICSP 2021, Jamshedpur, India, November 16–17, 2021, Proceedings 2. Springer International Publishing, 2021.

Based on Chapter 2.

[2] **Hamidi, Amirreza**, and Hossein Ghodosi. "Unconditionally Fast Secure Multi-party Computation with Multi-depths Gates Using Pre-computed Information." Proceedings of Seventh International Congress on Information and Communication Technology: ICICT 2022, London, Volume 2. Singapore: Springer Nature Singapore, 2022.

Based on Chapter 3.

[3] **Hamidi, Amirreza**, and Hossein Ghodosi. "Efficient Distributed Keys Generation of Threshold Paillier Cryptosystem." International Conference on Information Technology and Communications Security. Cham: Springer Nature Switzerland, 2022.

Based on Chapter 4.

[4] **Hamidi, Amirreza**, and Hossein Ghodosi. "Verifiable DOPE from Somewhat Homomorphic Encryption, and the Extension to DOT." International Conference on Science of Cyber Security. Cham: Springer International Publishing, 2022.

Based on Chapter 5.

[5] **Hamidi, Amirreza**, and Hossein Ghodosi. "Outsourcing Verifiable Distributed Oblivious Polynomial Evaluation from Threshold Cryptography." International Conference on Information and Communications Security. Singapore: Springer Nature Singapore, 2023.
Based on Chapter 6.

[6] **Hamidi, Amirreza**, and Hossein Ghodosi. "Fair Distributed Oblivious Polynomial Evaluation via Bitcoin Deposits: Compute-as-a-Service." Nordic Conference on Secure IT Systems. Cham: Springer Nature Switzerland, 2023.
Based on Chapter 7.

The research carried out in this thesis was supported by a James Cook University Postgraduate Research Scholarship (JCUPRS). Moreover, the publications fees were funded by the College of Science & Engineering of James Cook University.

ABSTRACT

Secure multi-party computation (MPC) is a fundamental concept in distributed cryptography where a set of n participants with their private inputs wish to compute any given function without revealing any information about their inputs. The given function ranges from a simple summation to in-depth multiplication which can simulate many problems in real-world cryptography where the parties need to maintain privacy of their inputs (e.g., private auction, distributed keys generation of cryptosystems, privacy-preserving machine learning, set intersection, electronic voting and etc.). So, this system is the building block of many cryptographic models.

It is usually assumed that the security of this system is threatened by two types of adversaries which can corrupt a subset of the total participants: either passive (semi-honest) or active (malicious). A passive adversary can read the inputs of the corrupt parties, trying to obtain information about the honest players' inputs. An active adversary can additionally make the corrupt parties deviate from the protocol in an arbitrary fashion to change the outcome of the computation.

Multi-Party Computation from Pre-Processed Information

In this study, we investigate on arithmetic MPC using pre-processed values which can be generated and computed in an offline pre-processing phase at any time before implementing the actual online computation in MPC primitives. These values are uniformly random and can be generated and distributed either by a trusted third party (initializer) or the participants themselves. Here, the point of having a pre-processing phase is that the parties do not need to know/hold their private inputs in this phase while they are able to evaluate a given function much faster than normal MPC protocols with information-theoretic security using the pre-processed random values in the online computation phase.

Having said about the idea of pre-distributed random values, we first employ

this technique to propose two general MPC protocols and an application scheme of MPC in distributed keys generation of a cryptosystem. Namely, our contributions for general MPC are as follows:

1. *Secure Multi-Party Computation Using Pre-distributed Information from an Initializer*: An initializer is a type of third party which distributes some random values in the pre-processing phase and does not get involved in the actual online computation. Our protocol needs only one round of online computation for both addition and multiplication gates (operations). The protocol is unconditionally secure against a passive adversary corrupting a coalition of t parties, and the communication complexity for a multiplication is linear.

2. *Unconditionally Fast Secure Multi-party Computation with Multi-depths Gates*: The evaluation of n-inputs multiplication in just one communication round of online computation is achieved in this contribution. This results a fast computation technique of any given function with multi-depth multiplications. The protocol is unconditionally secure against t participants corrupted by a passive adversary.

3. *Efficient Distributed Keys Generation of Threshold Paillier Cryptosystem*: As an important application of MPC in real-world cryptography, an efficient scheme of distributed keys generation of Paillier cryptosystem is presented. The efficiency is in terms of the required participants in online computation and the communication overhead. The model is also secure against t parties corrupted by an active adversary.

Distributed Oblivious Polynomial Evaluation

Oblivious polynomial evaluation (OPE) is a special case of two-party computation where a sender party P_1 holds a polynomial $f(x)$ of degree k and a receiver party P_2 has a value α. They wish to perform a secure computation such that P_2 obtains the value $f(\alpha)$ while neither party gets any information about the other party's private input. Distributed oblivious polynomial evaluation (DOPE) is a variant of

OPE where the main two parties communicate with a set of t distributed cloud servers instead of direct communication with each other. The advantage is that this system is more decentralized and secure compared to the traditional OPE since it is less prone to the single point of failure attack. We also give three contributions of DOPE models as follows:

4. *Verifiable DOPE from Somewhat Homomorphic Encryption, and the Extension to DOT*: The servers employ additive homomorphic encryption to compute the encrypted shares of the outcome. This protocol preserves the security against an active adversary corrupting a coalition of $t-1$ cloud servers. The scheme is also extended to a protocol of secure $\binom{1}{2}$ distributed oblivious transfer.

5. *Outsourcing Verifiable DOPE from Threshold Cryptography*: A lightweight DOPE scheme is presented where the main expensive computation of secure decryption procedure is outsourced to the cloud servers using threshold cryptography. This technique makes it possible that a receiver party P_2 with a low-computation device (e.g., a mobile or a laptop) gains the output. The security is maintained against a coalition of corrupt $t-1$ servers and the opposed party, and it offers a linear communication complexity.

6. *Fair Distributed DOPE via Bitcoin Deposits: Compute-as-a-Service*: We present the first fair and robust DOPE protocol using Bitcoin deposit transactions. The fairness property ensures that an honest cloud server gains the reward for conducting a computation service while a corrupt server has to pay some penalty amount to an honest party. A non-interactive fault-detection technique is proposed to detect a corrupt server/party. Thus, the perfect security is maintained against an active adversary.

Note that all the contributions above are based on peer-reviewed articles, which have been already published, in order to improve on the cutting-edge filed of MPC and the related area DOPE. To summarize, the aim of this thesis is to provide new schemes and practical tools for the improvement of the mentioned fields in terms of the efficiency and security.

Contents

List of Figures

Chapter 1

Introduction

It is quite an interesting fact that various quantitative and even qualitative problems can be expressed by arithmetic functions. A very common case is when some parties want to solve a problem by evaluating the function while they need to preserve their privacies. This is where multi-party computation (MPC) can help. MPC is a broad and fundamental concept in distributed cryptography where n parties $P_1, \ldots, P_n$ with their private inputs $x_1, \ldots, x_n$ wish to obtain the output of an agreed function using their inputs without revealing any information about the inputs. The function can contain several addition and multiplication operations. Indeed, MPC is an important building block for building secure distributed system and reducing the required trust between the participants. It has various practical applications such as but not limited to private auction [18], satellite collision [67], privacy-preserving machine learning [79], secure statistical analysis of personal information [17], distributed keys generation of cryptosystems [59], confidential benchmarking [32] and protection against side-channel attacks in hardware [13, 85, 93]. Figure 1.1 illustrates a MPC system with n participants.

1.0.1 General Security Model

It is assumed that an adversary can corrupt a subset of total parties in this system where there are two types of adversaries: either passive (semi-honest) or active (malicious).

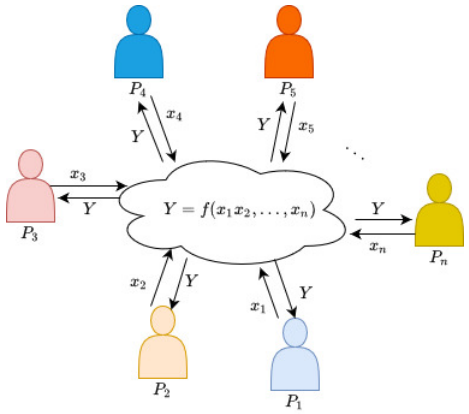

Figure 1.1 : A MPC system with n participants

A passive adversary can read the internal information of corrupt players in the attempt to obtain as much as possible information about the honest parties' inputs. In addition to this ability, an active adversary can make the corrupt players deviate from the protocol in an arbitrary fashion trying to forge the outcome of the function without being detected. Moreover, the adversarial parties are assumed to be either computationally bounded or computationally unbounded. This assumption suggests that an adversary can have limited time and computation power (computationally bounded) or unlimited time and computation power (computationally unbounded). Based on this assumption, the security of any MPC protocols is divided into two categories: either information-theoretic (unconditional) security or computational (conditional) security. An unconditional secure MPC scheme preserves the security against a computationally unbounded adversary, while a computational secure MPC protocol holds the security against a computationally bounded adversary. It is worth to mention that computationally secure protocols are less efficient and usually need some form of public-key cryptography. Generally speaking, the security of an unconditionally secure protocol is stronger as it does not depend on the assumptions of the computational intractability of the adversary. However, this is achieved at the cost of higher communication complexity compared to a computational secure pro-

tocol. Hence, depending on the requirements and available tools, one may consider a balance between security and efficiency to choose a suitable MPC system.

Any MPC system must satisfy two main security conditions: *privacy* and *correctness*:

- **Privacy** ensures that any information about the inputs of honest participants cannot be leaked to a passive adversary, except what the parties can obtain from the computation outcome after running the protocol.
- **Correctness** guarantees that the participants are able to compute the correct outcome at the end of the protocol or detect any malicious behaviour if an active adversary is present.

After all, we can have a better definition of a MPC system as:

Definition 1. *A secure MPC system enables a set of n participants $P_1, \ldots, P_n$, with their inputs $x_1, \ldots, x_n$, to compute any given function $f(x_1, \ldots, x_n)$ without revealing any information about the inputs in the presence of an adversary $\mathcal{A}$, such that $\mathcal{A}$ can corrupt a threshold number of t parties.*

MPC systems are based on two building blocks: either secret sharing (for arithmetic functions) or garbled circuit (for boolean functions). Since the scope of this thesis is about arithmetic circuits, we first explain a very popular secret sharing scheme.

1.1 Secret Sharing

We employ the Shamir's secret sharing scheme which is a threshold method and was first introduced by [96]. A secret holder (dealer) distributes a secret s among n parties using a random polynomial $f(x) = s + a_1 x + \ldots + a_t x^t \mod q$ where q is a prime number with the condition $q > n$ and the coefficients a_j (for $j = 1, \ldots, t$) are

chosen randomly in the field $\mathbb{F}_q$. Namely, the dealer sends privately a share f_i to the party P_i where $f_i \leftarrow f(i)$. In order to obtain the secret, a set of $t+1$ qualified parties (access structure) pools their shares and can reconstruct the free constant (secret) with no private interaction using the Lagrange interpolation method as follows:

$$f(0) = \sum_{i=1}^{t+1} f_i \cdot l_{0,i}$$

Where $l_{0,i}$ is the Lagrange coefficient of the party P_i calculated as:

$$l_{0,i} = \prod_{j \neq i} \frac{-j}{i-j}$$

Where i is the index of P_i and j is the index of every other party P_j except P_i. Clearly, the security of this secret sharing scheme is unconditional against up to t semi-honest participants (called t-private). More formally:

Definition 2. *A threshold secret sharing scheme for distributing the secret s is t-private if a passive adversary, corrupting any subset of up to t participants, cannot gain any information about the secret s.*

In this thesis, we denote the t-sharings $[s]_t$ as a set of $t+1$ shares of a random polynomial $f(x)$ with the degree/threshold t and the secret s.

As an important homomorphic property, the Shamir's secret sharing is linear where the participants can reconstruct any linear function with no interaction [11]. Namely, given two sets of t-sharings $[s_1]_t$ and $[s_2]_t$, a party is able to locally compute a share of the addition $s_1 + s_2$ simply by adding the two shares together as $[s_1 + s_2]_t \leftarrow [s_1]_t + [s_2]_t$. Also, a share of multiplication by a constant factor a can be calculated as $[a \cdot s_1]_t \leftarrow a \cdot [s_1]_t$ for any $a \in \mathbb{F}_q$.

It was first shown by [98] that a malicious party, who does not follow the protocol, can change the real secret in the reconstruction phase without being detected by giving false information as their real share. This is where **verifiable secret sharing**

(VSS) techniques are required to prevent these types of attacks in the secret sharing. These techniques are categorized as either *fault-detection* or *cheater-detection*. In the secret sharing with fault-detection, even one honest party is enough to detect any inconsistency in the system and abort the protocol implying that the majority can be dishonest. Generally speaking, these protocols are more efficient (lower communication and computation overheads) but less accurate with low probability of error as the honest participants are unable to find the cheater parties. On the contrary, the schemes with cheater-detection work with honest majority in the system while the perfect security is achieved. This term means that the honest majority have the ability to detect the cheaters at the cost of less efficiency.

1.2 MPC

Multi-party computation (MPC) was first introduced by Yao [100] in 1982 for the case of two-party computation and later it was extended by [78] to the case of n parties using cryptographic intractability assumptions. The first information-theoretic MPC systems were independently presented by [10] and [24]. In particular, [10] proposed an information-theoretic MPC protocol with the perfect security against at most $t < n/3$ parties corrupted by an active adversary. Later, [92] showed that if broadcast channels are available, unconditional security with low probability of error is achievable in the presence of up to $t < n/2$ corrupt players. Also, a MPC protocol secure against dishonest majority using cut and choose technique was given by [39]. Generally speaking, the bilateral channels between every pair of players in unconditionally secure MPC are assumed to be private and secure.

The functions (circuits) in MPC can include inputs *addition* and *multiplication*. Note that the communication complexity of a MPC protocol is measured as the maximum number of the elements in the field sent and received by the honest parties in the protocol. As discussed in section 1.1, the secret sharing scheme is usually

employed in unconditional secure MPC protocols. Here, a participant can calculate a share of an addition gate with no interaction. For instance, given two t-sharings $[x_i]_t$ and $[x_j]_t$ of the two inputs x_i and x_j, a party can locally compute the new share $[x_i + x_j]_t = [x_i]_t + [x_j]_t$ of the inputs addition, and at least $t + 1$ honest participants obtain the output. However, a problem arises for a multiplication gate which reduces the level of security and efficiency. Namely, a party can compute a share of the two-inputs multiplication $x_i \times x_j$ by multiplying their respective shares together as $[x_i \times x_j]_{2t} = [x_i]_t \times [x_j]_t$. It is trivial to show that the new shares $[x_i \times x_j]_{2t}$ belong to the multiplication polynomial of degree $2t$. It implies that at least $2t + 1$ honest parties are required to gain the output $x_i \times x_j$. Moreover, the free coefficients of the resulted polynomial are not random which contradicts the information-theoretic security definition [10]. In summary, the computation of the addition gate is straightforward whilst the multiplication gate raises a bottleneck in a MPC system.

Beaver's Scheme

Some solutions have been proposed to deal with the multiplication problem and perform the degree reduction of the resulted polynomial. [53] and, later, [68] presented methods for reducing the degree of a polynomial using linear feature of secret sharing to redistribute new shares which result the same old secret (pro-active secret sharing). A very popular and efficient technique is the Beaver's scheme [6] where a share of a multiplication gate is computed at the cost of distributing some random shares and reconstructing two random numbers. Namely, three t-sharings $[a]_t$, $[b]_t$ and $[c]_t$ of a correlated random triple $a \cdot b = c$ are distributed among the participants. Each party locally calculates two random shares of degree t as $[\epsilon]_t = [x_i]_t - [a]_t$ and $[\delta]_t = [x_j]_t - [b]_t$ and the parties reconstruct the random values ϵ and δ. Now, each

party can calculate a share of the multiplication $y = x_i \cdot x_j$ of degree t as follows:

$$[y]_t = \epsilon \cdot \delta + \epsilon \cdot [b]_t + \delta \cdot [a]_t + [c]_t \tag{1.1}$$

Clearly, this equation holds the linear feature of secret sharing and only $t+1$ parties are required to obtain the output. It is trivial to prove the correctness of the equation 1.1 which can be written as:

$$\begin{aligned} [y]_t &= ([a]_t + \epsilon) \cdot ([b]_t + \delta) \\ &= ([a]_t + (x_i - a)) \cdot ([b]_t + (x_j - b)) \\ &= [x_i \cdot x_j] \end{aligned}$$

The communication complexity of this technique for each multiplication gate is linear $O(n)$.

1.2.1 MPC from Pre-Processed Information

Perhaps the most efficient MPC protocols are the ones with pre-processed information. The idea is that some random values are computed and distributed among the participants in a phase before the implementation of the actual circuit computation. The generation of the random values usually requires expensive computation power while the point is that the function evaluation stage is lightweight and can be conducted much faster than the normal MPC systems. These protocols include two phases:

- **Offline Pre-Processing Phase**: This phase is offline meaning that the participants do not hold their inputs, and it can be executed at any time before the computation phase. The parties conduct some expensive computation to generate some random values (e.g., the Beaver's triples) which can be used in the computation phase. The generation of the pre-processed random values can be based on some techniques including homomorphic encryption of public

key cryptosystems [12, 35], oblivious transfer [71, 46] or using cloud servers (commodity-based cryptography) [7, 95, 97].

- **Online Computation Phase**: In this online phase the participants have their inputs and distribute them among the parties. The parties perform a fast and cheap evaluation of the given function using the random values they have obtained in the pre-processing phase.

This technique has recently caught the most attention as the heavy expensive parts of the computation are transferred to the offline pre-processing phase while the parties can enjoy the efficient and cheap information-theoretic secure computation in the actual online phase.

1.3 Oblivious Polynomial Evaluation

Secure two-party computation is a setting of multi-party computation where two parties with their private inputs want to execute a function computation while the privacy of their inputs is preserved. In detail, parties P_1 and P_2, holding the respective inputs x and y, jointly compute some function $f(x, y)$ such that P_1 and P_2 obtain the outputs $f_1(x, y)$ and $f_2(x, y)$, respectively. This system is denoted by the functionality $(x, y) \rightarrow (f_1(x, y), f_2(x, y))$. Oblivious polynomial evaluation (OPE) is a variant of two-party computation where a sender party P_1 holds a polynomial of degree k as $f(x) = a_0 + a_1x + \ldots + a_kx^k$ and a receiver party P_2 has a value α. They wish to conduct a secure computation such that the receiver party P_2 gains the value $f(\alpha)$ while the sender party P_1 gets nothing. The security model must ensure that neither party obtains any information relating to the other party's private input except the output P_2 gains, i.e. $f(\alpha)$. The inputs and the output are over a pre-determined field $\mathbb{F}_q$ and the system can be denoted by the functionality $(f(x), \alpha) \rightarrow (\perp, f(\alpha))$. Figure 1.2 shows an OPE system schematic. More formally:

Definition 3. *In a secure OPE system, two parties with their private inputs participate in the system over the field* $\mathbb{F}_q$ *where the sender party* P_1 *holds a polynomial* $f(x)$ *of degree* k *and the receiver party* P_2 *has a value* α*. They execute a computation procedure such that* P_2 *obtains the value* $f(\alpha)$*. The system is said to be securely implemented if:*

- P_1 *cannot distinguish* α *from a value* α' *randomly chosen over the field.*
- P_2 *cannot gain any information in relation to the polynomial* $f(x)$ *except the output* $f(\alpha)$*.*

Figure 1.2 : An OPE system schematic

Background

OPE, which was first introduced by [81], is a notable building block of many cryptographic primitives and security models such as RSA keys generation [55], data mining [1], oblivious neural networking [23], oblivious keyword search [47], scalar product [56], metering the number of visitors to a website [81], set intersection [48] and electronic voting [87]. In secure information-comparison protocols, two parties with their private inputs, say x and y, want to know whether $x > y$ without leaking any additional information of the inputs which can be used in password comparison, online auction and benchmarking [41]. It also plays an important role in privacy-preserving machine learning where a client wishes to execute a secure protocol with a server/company to gain private information in the classification phase of a machine

learning algorithm [41]. These algorithms usually have two phases: training and classification where OPE plays a secure tool to obtain the output in the classification phase. For instance, in healthcare system a patient intends to obtain a prediction of his health status from a healthcare company holding a trained model without revealing any information about his personal health records [57, 50].

1.3.1 Distributed Oblivious Polynomial Evaluation

The recent development of cloud computing has made it possible to outsource main expensive computations to a number of cloud servers. distributed oblivious polynomial evaluation (DOPE) is a special case of OPE where the main two parties communicate with a number of t designated cloud servers to outsource their OPE computations instead of direct interactions with each other. This system offers higher flexibility as the main two parties do not communicate directly and, also, they can remain anonymous to each other. However, the main challenge is that this approach incurs obvious security and privacy breaches as the system must maintain the security conditions against more parties (the servers) compared to an OPE system. One may think of using multi-party computation solutions in DOPE systems, nevertheless, these solutions are generic and are very inefficient, especially when large inputs are involved [1]. Figure 1.3 illustrates a DOPE system schematic.

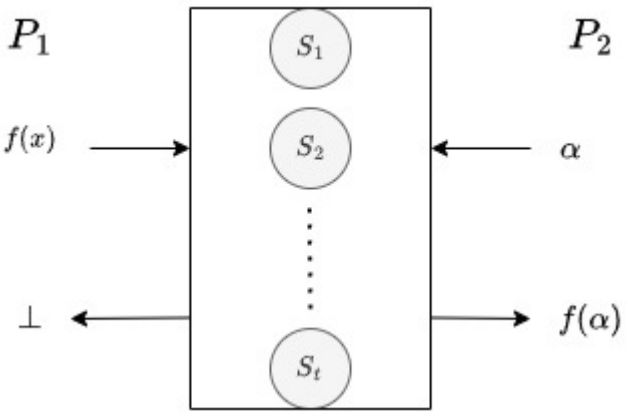

Figure 1.3 : A DOPE system schematic

In the literature, some studies have investigated the idea of employing either one cloud third party [64, 57, 72, 20, 50] or a set of distributed ($t \geq 2$) servers [75, 27] in their protocols. With regards to DOPE using just one third party, [57] presented a verifiable privacy-preserving monitoring scheme for mobile health systems with one cloud-assisted server. Also, [50] proposed a verifiable and private OPE protocol, by employing homomorphic encryption feature and a trusted server, to record medical datasets. [20] formally defined the notion of private polynomial evaluation with a designated server.

Using just one cloud server offers lower communication complexity, however, the serious drawback is that corrupting only one server causes a central point of failure breaking the whole security of the system. Therefore, the DOPE protocols which outsource the computation to a number of t distributed cloud servers ($t \geq 2$) are potentially more decentralized and secure, since corrupting the t servers is less likely. The number of t cloud servers can be denoted as the *security parameter*.

The first information-theoretic DOPE protocols were studied by [75]. The main problem of their protocol is that the privacy of the parties' inputs is imperfect (i.e., it is not maintained against the maximum required number of $t-1$ servers). To deal with this problem, they introduced some publicly known information that increases the communication overhead. As a result, their protocol does not seem to be suitable and efficient for a system with large datasets. [26] proposed an unconditionally secure DOPE protocol where the main two parties P_1 and P_2 are required to communicate directly which does not meet the security condition that the two parties are allowed to interact only with the cloud servers. We will discuss the security model of a DOPE system later in Chapter 5.

1.4 Motivation

As discussed earlier in this Chapter, MPC is an important structure of many distributed cryptographic primitives. Recently, the idea of MPC from pre-processed information has caught lots of attentions by renowned researchers due to its efficiency. Furthermore, with the recent advancement in the decentralized blockchain technology, efficient MPC systems can be a powerful tool to add security and reduce the trust between the nodes in it. That is why we aim to give improvements on MPC system.

DOPE is relatively a new research field and just a few number of studies has been conducted in this field. Moreover, with the recent development of machine learning, DOPE (as a variant of OPE) has appeared as a vital component in privacy-preserving machine learning. Therefore, more researches are required in this field to make DOPE systems more efficient and practical in the applications of real-world cryptography.

1.5 Organization and Contributions

In the rest of content in this thesis, each Chapter is based on a published peer-reviewed conference article. It should be noted that each Chapter contains the required definitions, preliminaries, security models and evaluations. That is why we include only general definitions and background in the Introduction Chapter. Namely, the rest of the thesis is designed as follows:

- **Chapter 2**: An unconditionally MPC protocol is proposed using a third party initializer where the initializer is just involved in the offline pre-processing phase [58]. In terms of the efficiency, this protocol can be considered as an alternative technique to the Beaver's scheme.

- **Chapter 3**: We give an unconditionally secure MPC scheme where the eval-

uation of a function with n-inputs multiplication is conducted in only one round of communication in the online computation phase which improves on the communication round complexity in this field. This ability results a fast computation of any given function in just one online communication round [60].

- **Chapter 4**: We present an efficient and secure scheme of distributed keys generation of threshold Paillier cryptosystem using the MPC from pre-processed information [59]. It gives improvements on the required participants and the communication complexity in the online computation phase for the keys generation of this cryptosystem in the threshold fashion.

- **Chapter 5**: The first verifiable DOPE system is proposed using the homomorphic feature of the Paillier encryption system. The protocol is also extended to a system of secure $\binom{1}{2}$ distributed oblivious transfer [61].

- **Chapter 6**: We give a secure lightweight DOPE scheme where the main expensive computation of the decryption procedure is outsourced to the cloud servers using the idea of threshold cryptography [63]. This technique makes it possible that a receiver party with a low computational power device can participate in the DOPE and obtain the output.

- **Chapter 7**: The first fair and robust DOPE system is presented by employing Bitcoin deposit transactions [62]. The fairness property ensures that an honest cloud server gains the reward for performing a computation service while a corrupt server compensates some penalty amount to an honest party.

- **Chapter 8**: Finally, this Chapter gives the conclusion of this thesis and some potential suggestions for future research.

Chapter 2

Secure Multi-Party Computation Using Pre-Distributed Information from an Initializer

2.1 Introduction

In this Chapter, we propose an information-theoretic secure MPC protocol by employing an initializer party in the pre-processing phase. Note that, as it was previously discussed in the section 1.2.1, we utilize the technique of MPC from pre-processed information in the scope of this thesis. An initializer is a third party which is just involved in the offline pre-processing phase. In fact, it does not need to know the function to be computed, nor the players' inputs, it just distributes some raw materials used for the actual computation. This idea allows to have an efficient online computation phase where cheap information-theoretic primitives are used.
In detail, our protocol is a threshold $(t+1, n)$ MPC system based on the Shamir's secret sharing which was discussed in section 1.1. More formally:

Definition 4. *A threshold MPC system enables a set of* $(t+1, n)$ *(called* $t+1$ *out of* n*) participants* $P_1, \ldots, P_n$*, with their private inputs* $x_1, \ldots, x_n$*, to compute any given function* $f(x_1, \ldots, x_n)$ *without revealing any information about the inputs.*

Note that this definition implies that the system is t-private where it must hold the privacy condition against at most t parties corrupted by a passive adversary. Indeed, it defines the security level of a MPC protocol where the privacy of the honest participants' inputs and the actual protocol outcome is preserved against the maximum t semi-honest parties [54].

2.1.1 Background

In the literature, some renowned MPC studies have utilized the method of random pre-processed information before the actual computation in both boolean circuit [14, 77] and arithmetic circuit [12, 40, 35]. Commodity-based cryptography was first introduced by [7]. The idea of employing an initializer server in a pre-processing phase was first suggested by Rivest (1999) [94] to propose efficient unconditionally secure commitment and 1-out of-2 Oblivious Transfer protocols. Later, it was also used in the MPC studies of [39] and [30] to generate random triples in an efficient way before the actual computation. Recently, [25] presented a protocol where some pre-computed information is distributed by an initializer in the pre-processing phase and the online phase is conducted using oblivious linear evaluation in the presence of a passive adversary. With regards to employing more than one server, [97] proposed a MPC protocol where a set of raw triples is distributed by cloud service providers in a pre-processing phase. An important drawback of using more than one server is that it causes much higher communication complexity in the pre-processing phase since the participants must communicate with multiple servers to receive the required information.

2.1.2 Our Contribution

We present an arithmetic information-theoretic secure MPC protocol which computes a share of inputs addition and multiplication gates with the same multiplicative depth in parallel with no interaction by employing an initializer server [58]. It also can be extended to compute a share of multiplication gates with different multiplicative levels. Our protocol needs only $n \geq t + 1$ participants in the both offline and online phases and it is unconditionally secure against a coalition of t participants controlled by a passive adversary. The protocol requires only one round of secret sharing in the online phase that is used for both the inputs addition and the

same-level multiplication gates. Hence, both of these computation gates can be conducted in parallel. This can be achieved at the cost of two rounds of reconstruction in the online phase and with the linear communication complexity $O(n)$. Therefore, our protocol can be considered as an alternative to the Beaver's scheme in terms of the efficiency. Furthermore, the share of multiplication gates with different multiplicative depths can be computed by implementing it in different communication rounds.

2.2 Outline

Our model employs an independent initializer in the pre-processing phase, to distribute random values among the participants in a probabilistic functionality. Suppose, two input holders P_i and P_j with their corresponding inputs, x_i and x_j, are given two random numbers r_i and r_j, respectively. The idea is that P_i can calculate two numbers, α_{i1} and α_{i2}, such that $f(\alpha_{i1}) \simeq r_i$ and $f(\alpha_{i2}) \simeq r_i$ while $f(x_i) \not\simeq r_i$ where the symbols $\simeq$ and $\not\simeq$ show the dependence and independence notations, respectively. The same method can be proven for P_j with his corresponding random number r_j such that $f(x_j) \not\simeq r_j$. Thus, the multiplication gate $f(x_i, x_j) \leftarrow x_i \times x_j$ can be expressed as $f(x_i, x_j) \not\simeq r_i, r_j$ meaning that the outcome doesn't depend on the pseudo-random numbers. In fact, the random numbers just ensure the privacy condition of our protocol. This trick achieves the possibility of computing a share of the both addition and multiplication gates in parallel. Moreover, this computation phase can be conducted using just one round of online secret sharing by the input holders.

2.3 The Protocol

Our protocol consists of one offline pre-processing phase and an online phase. It can be executed in parallel for addition gates and multiplication gates with the same

multiplicative depth, with the capability of being extended to the multiplication gates with intermediate levels in any given function $f(x_1, \ldots, x_n)$. It is assumed that private channels between the players are secure and synchronous.

2.3.1 Pre-Processing Phase

This offline phase can be executed by an initializer T at any time before the online computation phase and it is operated in a probabilistic assumption. Note that this pre-computed information can be given to the players or purchased by them before the actual computation. Let the inputs multiplication gate be $x_i \cdot x_j$ belonging to the two input holders, P_i and P_j, and also a n-inputs addition gate be $\sum_{i=1}^{n} x_i$ in the circuit. Suppose n participants want to compute a share of these gates in parallel. Fig 2.1 illustrates the detail of the offline pre-processing phase.

- The initializer T generates n random numbers r_i (for $i = 1, \ldots, n$) and distributes them by the t-sharings $[r_i]_t$ among the participants.
- For the multiplication gate $x_i \cdot x_j$, T calculates the value $R = r_i \cdot r_j$ and distributes the t-sharings $[R]_t$ among the participants.
- T leaves the protocol and takes no step further.

Figure 2.1 : The offline pre-processing phase of our proposed protocol

The communication upper bound of this phase for each multiplication gate $x_i \cdot x_j$ is $O(n)$.

2.3.2 Computation Phase

This is a deterministic online phase where the input holders and the participants are involved to compute the given function. They employ the random pre-computed information distributed by the initializer in the previous offline phase. This phase

is divided into two sub-phases: *sharing* and *actual computation*.

Sharing

This sub-phase is implemented once for both addition and multiplication gates. Figure 2.2 shows the online sharing sub-phase.

Inputs: Each input holder P_i (for $i = 1, \ldots, n$) has the input x_i.

- Each input holder P_i distributes the t-sharings $[x_i]_t$ of his input among the participants.
- Each participant P_k $(1 \leq k \leq n)$ computes two new shares, using the corresponding random share r_i distributed by T in the pre-processing phase, as follows:

$$[z_i]_t = [x_i]_t + [r_i]_t$$

$$[y_i]_t = [x_i]_t - [r_i]_t$$

Figure 2.2 : The online sharing in the computation phase

Addition

Here, the actual online computation of the addition gate takes place where each participant is able to compute a share of an inputs addition gate with no interaction. This is as efficient as the homomorphic secret sharing method for the addition gate. Figure 2.3 presents the online computation of the inputs addition gate.

Output: Each participant P_k among a set of $t+1$ participants $(1 \leq t+1 \leq n)$ computes a share of $f(x_1, \ldots, x_n) = x_1 + \ldots + x_n = \sum_{i=1}^{n} x_i$.

- Each P_k computes the new share of the inputs addition as follows:

$$[\sum_{i=1}^{n} x_i]_t = \frac{\sum_{i=1}^{n}([z_i]_t + [y_i]_t)}{2}$$

Figure 2.3 : online computation sub-phase of the inputs addition gate

Now, we evaluate the security conditions of the proposed inputs addition gate based on the general MPC security conditions described in section 1.0.1.

Correctness

Theorem 1. *Each participant has the correct share of the function* $f(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i$ *at the end of the inputs-addition sub-phase.*

Proof. Based on homomorphic nature of the Shamir's secret sharing [11], if a set of $n \geq t+1$ participants pools their shares, they can reconstruct the addition gate with the following free term:

$$\begin{aligned} f &= \frac{\sum_{i=1}^{n}[(x_i + r_i) + (x_i - r_i)]}{2} \\ &= x_1 + \ldots + x_n \end{aligned}$$

Privacy

Theorem 2. *At the end of the inputs addition, a coalition of up to t participants (not including* P_i*) cannot gain any information about the input of* P_i*, except the information they can obtain from the outcome after the computation.*

Proof. Without loss of generality, let x_1 be the input of the player P_1. The view of at most t participants corrupted by a passive adversary (not including P_1), denoted

by $VIEW_{\mathcal{A}}$, would include the following shares:

$$f_2 = \frac{\sum_{i=1}^{n}([z_1]_t + [y_1]_t)}{2}$$

$$f_3 = \frac{\sum_{i=1}^{n}([z_1]_t + [y_1]_t)}{2}$$

$$\vdots$$

$$f_{t+1} = \frac{\sum_{i=1}^{n}([z_1]_t + [y_1]_t}{2}$$

Where, based on the threshold feature of the Shamir's secret sharing discussed in section 1.1, $VIEW_{\mathcal{A}}$ cannot gain any information about the free term $(z_1 + y_1) \leftarrow [(x_1 + r_1) + (x_1 - r_1)]$. Thus, the value x_1 is t-private.

Multiplication

In this online sub-phase each participant computes a correct share of a two-inputs multiplication gate. Note that the protocol can be employed for computing a share of all the multiplication gates with the same multiplicative depth at the same time in parallel. In other words, in a circuit with m multiplication gates with the same multiplicative level (m gates with the same degree), it can be executed m times in parallel.

Output: Each participant P_k computes a share of $f(x_i, x_j) = x_i \times x_j$ in the given function $f(x_1, \ldots, x_n)$.

- A set of at least $t+1$ participants pools their shares $[z_j]_t$ and $[y_i]_t$, and reconstructs the values z_j and y_i, respectively.
- Each P_k computes a share of the inputs multiplication gate with no interaction using the corresponding share $[R]_t$, sent by T in the pre-processing phase, as follows:

$$[x_i \cdot x_j]_t = \frac{z_j \cdot [z_i]_t + y_i \cdot [y_j]_t}{2} - [R]_t$$

Figure 2.4 : online computation sub-phase of the inputs multiplication gate

Moreover, the protocol can be extended to compute a share of the multiplication gates with different multiplicative levels such that it must be implemented for each intermediate gate separately. As an example, suppose each participant P_k wishes to compute a share of the function $f(x_1, x_2, x_3) = x_1 \cdot x_2 \cdot x_3$. First, the participants obtain two sets of the pre-distributed information (r_1, r_2, R_1) and (r_{12}, r_3, R_2) where $R_1 \leftarrow r_1 \cdot r_2$ and $R_2 \leftarrow r_{12} \cdot r_3$. The participants compute a share of the multiplication gate $x_1 \cdot x_2$ denoted by $[x_1 \cdot x_2]_t$ using the multiplication sub-protocol in one communication round . Then, similarly, they execute the protocol in another communication round re-assigning the shares $[x_i]_t \leftarrow [x_1 \cdot x_2]_t$, $[x_j]_t \leftarrow [x_3]_t$, $[r_i]_t \leftarrow [r_{12}]_t$, $[r_j]_t \leftarrow [r_3]_t$ and $[R]_t \leftarrow [R_2]_t$. Thus, in a circuit with n multiplication gates with different multiplicative depths, the protocol must be executed in n separate rounds of communication. This is analogous to the rounds complexity of the Beaver's scheme [6]. The communication complexity of a multiplication gate is bounded to be linear $O(n)$.

The security condition of the each multiplication gate is evaluated in following section.

Correctness

Theorem 3. *Each participant P_k computes the correct share of the multiplication gate $f = x_i \times x_j$ at the end of the inputs multiplication sub-phase.*

Proof. After reconstructing the values z_j and y_i, the share $[x_i \cdot x_j]_t$ can be written as:

$$[x_i \cdot x_j]_t = \frac{(x_j + r_j) \cdot [z_i]_t + (x_i - r_i) \cdot [y_j]_t}{2} - [R]_t$$

If a set of at least $t + 1$ participants pools their shares $[x_i \cdot x_j]_t$, and based on the linear feature of the secret sharing, the outcome is computed as:

$$\begin{aligned} f &= \frac{(x_j + r_j)(x_i + r_i) + (x_i - r_i)(x_j - r_j)}{2} - r_i \cdot r_j \\ &= \frac{2x_i \cdot x_j + 2r_i \cdot r_j}{2} - r_i \cdot r_j \\ &= x_i \cdot x_j \end{aligned}$$

Privacy

Theorem 4. *At the end of the inputs multiplication gate a coalition of t participant corrupted by a passive adversary cannot gain any information about the input of P_i and P_j, except the information they can obtain from the outcome after the computation.*

Proof. Without loss of generality, let assume $P_1 \leftarrow P_i$ and $x_1 \leftarrow x_i$. The view of the adversary, $VIEW_{\mathcal{A}}$, would include the following shares:

$$f_2 = \frac{(x_j + r_j) \cdot [z_1]_t + (x_1 - r_1) \cdot [y_j]_t}{2} - [R]_t$$

$$f_3 = \frac{(x_j + r_j) \cdot [z_1]_t + (x_1 - r_1) \cdot [y_j]_t}{2} - [R]_t$$

$$\vdots$$

$$f_{t+1} = \frac{(x_j + r_j) \cdot [z_1]_t + (x_1 - r_1) \cdot [y_j]_t}{2} - [R]_t$$

Where, based on the threshold feature of the Shamir's secret sharing, $VIEW_{\mathcal{A}}$ cannot gain any information about the shared secret $[z_1]_t \leftarrow [x_1 + r_1]_t$ and R which is t-shared in the pre-processing phase. Furthermore, the input x_1 is masked by the value $y_1 \leftarrow (x_1 - r_1)$ where the t-sharings of r_1 is distributed in the pre-processing phase. Thus, $VIEW_{\mathcal{A}}$ can obtain no information about the input x_1 and and it is t-private.

2.4 Conclusion

we propose an unconditionally secure MPC protocol using an initializer in this Chapter. The initializer is involved in an offline pre-processing phase which can be conducted at any time before running the actual online protocol. It can be considered as a server which generates and distributes some random pre-computed information among the participants in a probabilistic functionality such that the information can be utilized in the actual online computation phase. Our protocol presents a solution for the problem of inputs multiplication with linear communication complexity $O(n)$ in the presence of only $n \geq t+1$ participants in both offilne and online phases which can be considered as an alternative technique to the Beaver's scheme. The protocol is unconditionally secure against a coalition of t participants corrupted by a passive adversary. The online sharing phase can be employed to execute both the inputs addition and multiplication gates, with the same multiplicative depth, in parallel. Also, it can be extended to the case of multiplication gates with different multiplicative levels at the cost of separate communication rounds.

Chapter 3

Unconditionally Fast Secure Multi-party Computation with Multi-depths Gates Using Pre-computed Information

3.1 Introduction

One of the main problems in a MPC system is the computation of multiplication gate with different multiplicative depths as it raises the communication rounds overhead. For instance, in order to compute a share of a given function $f(x_1, x_2, x_3) = x_1 \cdot x_2 \cdot x_3$, the players must perform the first round of communication with the shares of x_1 and x_2 to obtain the shares of $x_1 \cdot x_2$ and then conduct another communication round with the shares of $x_1 \cdot x_2$ and x_3 to compute the shares of the gate $x_1 \cdot x_2 \cdot x_3$. This can be done, for example, using the Beaver's triples scheme [6] or the method we introduced in Chapter 2 [58]. In other words, for the circuit $f(x_1, \ldots, x_n) = \prod_{i=1}^{n} x_i$ the participants need to implement $n-1$ separate rounds of communication, i.e., only the shares of the multiplications with the same multiplicative level is calculated in each communication round. We aim to deal with this problem and give a solution for it in this Chapter where a share of the n-inputs multiplication gate can be obtained in just one round of communication in the actual computation phase. Therefore, our protocol enables computing a share of any given function in just one round of online computation resulting a fast MPC system.

In the previous Chapter, we employed an initializer trusted server in our proposed MPC protocol. However, it may raise the security concern as the initializer can later

collude with the participants which breaches the privacy of the whole protocol. Thus, we employ a technique that requires at least $2t+1$ parties to generate and compute some pre-processed information in the offline pre-processing phase. As a result, it reduces the risk of central point of failure attack on a single server.

3.1.1 Our Contribution

In this Chapter, we present an information-theoretic secure MPC protocol to compute a share of any given function $f(x_1, \ldots, x_n)$, including up to the simultaneous n-inputs multiplication gate with different multiplicative depths, in just one round of online computation [60]. Note that all the multiplication gates can be implemented in parallel in the computation phase which makes this protocol fast and it gives an improvement on the current MPC systems in terms of the communication rounds complexity. Our protocol is operated in the arithmetic circuit and has two phases, an offline pre-processing phase and an online computation phase. We employ the technique of hyper-invertible matrices of [8], which has been used by many other MPC studies, in a loop to generate pseudo-random pre-computed values in the pre-processing phase. This technique ensures the privacy of the pre-processed values and, thus, the probabilistic functionality of the pre-processing phase. Our protocol is secure against a passive adversary corrupting up to t participants. The communication complexity to perform a n-inputs multiplication gate with different depths and an addition gate is just $O(n^2)$.

3.1.2 Outline

The rest of this Chapter is organized as follows: section 3.2 presents the preliminaries of our protocol. Section 3.3 describes our proposed protocol including the both phases, and the discussion of the security analysis. Finally, section 3.4 gives the conclusion.

3.2 Preliminaries

3.2.1 Secret Sharing with Double Sharings

As we already discussed in the section of unconditionally secure secret sharing (Section 1.1), the t-sharings $[s]_t$ denotes a set of $t+1$ shares of a random polynomial with the degree/threshold t and the secret s. As a feature of the Shamir's secret sharing, the secret s can be distributed by any number of polynomials with different degrees over the field. Hence, we also denote the double sharings $[.]_{d,d'}$ of a same secret with different degrees d and d'.

3.2.2 Model

We propose a secure MPC model that enables $t+1$ (where $n \geq t+1$) out of n participants to compute a share of any given circuit $f(x_1, \ldots, x_n)$ in only one round of computation. Specifically, it is capable of computing a share of a simultaneous n-inputs multiplication with intermediate multiplicative depths, which belong to the function $f(x_1, \ldots, x_n) = \prod_{i=1}^{i=n} x_i$, in just one online communication round. We employ the hyper-invertible matrices technique introduced in [8] for generating secure double sharings (two Shamir's secret sharing of the same random value with two different thresholds) of random values. In fact, an input holder generates the sharings $[\alpha]_d$ and $[\alpha]_{d'}$ of a random value α, unknown to adversary, where d and d' are two different degrees. This technique enables each pair of input holders P_i and P_j, where $1 \leq i, j \leq n$ $(i \neq j)$, to generate a triple securely in the offline pre-processing phase [8]. To achieve this goal in our model, a number of $n \geq 2t+1$ participants is required in the pre-processing phase. Then, using a for-loop for each multiplication with the same multiplicative depth would enable each participant to compute a share of a triple in threshold t which can be used for the n-inputs multiplication. Finally, the random generated t-sharings can be employed to compute both n-inputs addition gate and multiplication gates in parallel.

3.2.3 Generating Random Triples Based on Hyper-Invertible Matrices

We employ the technique of hyper-invertible matrices introduced in [8] to compute t-sharings of random values in the pre-processing phase of our model. Their method can be used to generate double sharings of a random value. This technique also has been used in other MPC studies to generate the shares of random triples in the secure fashion, see e.g., [38, 33, 22]. For the sake of readability and simplicity, we describe the important concepts and protocols here; however, it is recommended to refer to [8] for more detail.

In order to generate double sharings $[r]_{d,d'}$ of a random value r, we need to define the notion of hyper-invertible matrices.

Definition 5. *An $m \times n$ matrix is hyper-invertible if for any sets $R \subseteq \{1, \ldots, m\}$ of rows and $C \subseteq \{1, \ldots, n\}$ of columns with $|R| = |C| > 0$, the matrix M_R^C is invertible, such that it consists of the intersections between rows in R and columns in C.*

It has an important property where any mapping of n points, for example $(x_1, \ldots, x_n)$ to $(y_1, \ldots, y_n)$, can be calculated using two sets of fixed elements. In other words, any combination of n inputs/outputs induced by the matrix can be written as a linear function of the other n inputs/outputs. This feature enables a mapping of any n input values to be expressed as a linear function of the remaining n input values.

Based on the proposed technique on [8] for generating triples, each player P_i distributes a random value s_i, of degrees d and d', to the other participants using the Shamir's secret sharing. Each player now holds two vectors of the shares $[s_1]_{d,d'}, \ldots, [s_n]_{d,d'}$ and he can compute two new sets of shares with the degrees d and d', using the hyper-invertible matrix M, as follows:

$$([r_1]_{d,d'}, \ldots, [r_n]_{d,d'}) = M([s_1]_{d,d'}, \ldots, [s_n]_{d,d'})$$

Such that each set of the sharings $[r_i]_d$ lies on the polynomial $g(.)$ of degree d, and the value r_i is private against d passive adversaries. Thus, the double sharings $[r_1]_{d,d'}, \ldots, [r_n]_{d,d'}$ are the output for each player.

In order to generate random triples of degree t in a pre-processing phase, given sharings $[a_i]$, $[b_i]$ and $[r_i]$ of the random values a_i, b_i and r_i, each player computes a share of a polynomial of degree $2t$ as follows:

$$[d_i]_{2t} = [a_i]_t[b_i]_t - [r_i]_{2t} = [a_i b_i - r_i]_{2t}$$

and at least $2t+1$ players pool their shares $[d_i]_{2t}$ and publicly reconstruct the value d_i. Note that a number of at least $2t+1$ parties is required in this phase. Finally, each player locally computes his new share $[c_i]_t$ as follows:

$$[c_i]_t = d_i + [r_i]_t = a_i b_i - r_i + [r_i]_t$$

Each party now holds the t-sharings $[a_i]_t$, $[b_i]_t$ and $[c_i]_t$ of the triple $c_i = a_i \cdot b_i$. the scheme is clearly t-private and the communication complexity for generating one set of the double sharing $[r_i]_{t,2t}$ and the triple $[a_i]_t$, $[b_i]_t$ and $[c_i]_t$ is $O(n)$, and, thus, for n sets is $O(n^2)$ [8].

3.3 The Protocol

Our proposed protocol has one pre-processing phase for both addition and multiplication gates, and an online computation phase for calculating a share of any given function $f(x_1, \ldots, x_n)$. Note that the protocol can be implemented in parallel for all multiplication gates with different multiplicative levels as well. It is assumed that private communication lines between each two players are secure and synchronous.

1. **Pre-Processing Phase**: In this offline phase, each input holder generates a random value and shares it among the players. Depending on the existing multiplicative depths, the players use the hyper-invertible matrices in a loop to compute $t-$sharings of multiplying random values.

2. **Online Computation phase**: In this phase, each participant computes a share of the given function including n-inputs addition and n-inputs multiplication with different depths. The shares of the both gates can be computed in parallel. Thus, a share of any given function can be computed in only one round of online computation phase.

3.3.1 Pre-Processing Phase

The probabilistic functionality of this phase ensures the privacy of inputs in the computation phase. Note that we employ the result of generating random triples based on the hyper-invertible matrices technique discussed in section 3.2.3. Let assume that a given function $f(x_1, \ldots, x_n)$ has the gate of n-inputs multiplication with $n-1$ different multiplicative depths. It should be stated that this phase can be implemented in parallel for all multiplication gates $\prod_{i=1}^{m} x_i$ $(m \leq n)$ in the function $f(x_1, \ldots, x_n)$. Figure 3.1 shows the detail of the pre-processing phase.

- Each input holder P_i (for $i = 1, \ldots, n$) generates a random number γ_i (where $\gamma_i \neq 0$) and distributes the t-sharings $[\gamma_i]_t$ among the participants.
- Also, each input holder generates a random value s_i and shares it in double sharings $[s_i]_{t,2t}$ among the participants.
- Each player P_k locally computes two vectors of double sharings $[r_i]_{t,2t}$ using hyper-invertible matrices M described in section 3.2.3 as:

$$([r_1]_{t,2t'}, \ldots, [r_n]_{t,2t}) = M([s_1]_{t,2t'}, \ldots, [s_n]_{t,2t})$$

- In the gate of multiplying n inputs $f(x_1, \ldots, x_n) = \prod_{i=1}^{n} x_i$, for each pair of the inputs multiplication $x_i \cdot x_j$ with the same depth, where $j \leftarrow i+1$, while $j \leq n$ (executing a loop):
  - Each player P_k computes a $2t$-sharing as follows:

$$[\gamma_i]_t[\gamma_j]_t - [r_i]_{2t} = [\gamma_i \cdot \gamma_j - r_i]_{2t}$$

    and the players open the value $\gamma_i \cdot \gamma_j - r_i$.
  - Each player P_k computes the new t-sharing as:

$$[\gamma_i \cdot \gamma_j]_t = \gamma_i \cdot \gamma_j - r_i + [r_i]_t$$

  - The players execute $i \leftarrow i+1$ and $[\gamma_i]_t \leftarrow [\gamma_i \cdot \gamma_j]_t$. Note that the new local sharings $[\gamma_i]_t$ inside the loop is different from the initial sharings each input holder distributes as the local sharings change depending on the multiplicative depths of the gates.
  - if $i = n$, P_k returns the t-sharing $[\gamma_1 \times \ldots \times \gamma_n]_t \leftarrow [\gamma_i]_t$.

Output: Each player P_k holds the set of t-sharings $([\gamma_1]_t, \ldots, [\gamma_n]_t)$ and $[\gamma_1 \times \ldots \times \gamma_n]_t$.

Figure 3.1 : The pre-processing phase of the protocol

The communication complexity of this phase for computing a share of a two-inputs multiplication gate is $O(n)$, thus, for a n-inputs multiplication it equals $O(n^2)$.

3.3.2 Computation Phase

In this phase, each participant P_k computes a share of any n-inputs addition and n-inputs multiplication in parallel, i.e., this phase can be implemented once for any

given function. All the participants commence this phase while they are holding the t-sharings $[\gamma_i]_t$ (for $i = 1, \ldots, n$) and $[\prod_{i=1}^{n} \gamma_i]_t$. Figure 3.2 depicts the computation phase of the proposed protocol.

Input: The players $P_1, \ldots, P_n$ hold the inputs $x_1, \ldots, x_n$.

Output: A set of $t+1$ or more participants $(1 < t+1 \leq n)$ computes a share of the given function $f(x_1, \ldots x_n)$.

- Each input holder P_i calculates a random value α_i as:

$$\alpha_i = \frac{\gamma_i}{x_i}$$

  and shares it by $[\alpha_i]_t$ among the participants.

- A set of $t+1$ participants reconstructs the values α_i (for $i = 1, \ldots, n$).

Addition

- Each participant P_k calculates the new share of up to n-inputs addition as follows:

$$[x_1 + \ldots + x_n]_t = \sum_{i=1}^{n} \frac{[\gamma_i]_t}{\alpha_i}$$

Multiplication

- Each P_k computes the new share of the n-inputs multiplication gate as follows:

$$[x_1 \times \ldots \times x_n]_t = \frac{[\prod_{i=1}^{n} \gamma_i]_t}{\prod_{i=1}^{n} \alpha_i}$$

Figure 3.2 : The computation phase of our proposed protocol

Note that the amounts of inputs cannot be zero in the proposed protocol. However, it

doesn't affect the correctness since, in any MPC circuit, if a mathematical term in a function $f(x_1, \ldots, x_n)$ has a zero input, we can remove that term and it doesn't affect the outcome as well as the privacy of the other inputs. The fact that this protocol can compute a share of any given function including addition and multiplication gates with different multiplicative levels in only one communication round, would result a fast computation phase. This gives an improvement on the round complexity of the current MPC systems. The computation phase has the communication upper bound of $O(n^2)$ field elements.

3.3.3 Security Evaluation

We evaluate the protocol based on the security requirements discussed in section 1.0.1. Also, please refer to the section 3.2.3 and [8] for security assessment of the random sharings $[\prod_{i=1}^{n} \gamma_i]_t$ generated in the pre-processing phase.

Correctness

Theorem 5. *each participant P_k $(1 \leq k \leq t+1)$ obtains the correct share of a given function $f(x_1, \ldots, x_n)$ at the end of executing the protocol.*

Proof. After implementation of the pre-processing phase, each participant holds the sharings $[\gamma_i]_t$ (for $i = 1, \ldots, n$) and $[\prod_{i=1}^{n} \gamma_i]_t$. In the online computation phase, a set of $t + 1$ parties open the random values α_i ($\alpha_i \neq 0$). For the n-inputs addition gate, each participant P_k computes the following share:

$$f_k = \frac{[\gamma_1]_t}{\alpha_1} + \ldots + \frac{[\gamma_n]_t}{\alpha_n}$$

where, based on the linear feature of the secret sharing scheme, the shares f_k belong

to a polynomial with the the free term:

$$\begin{aligned} f(x_1, \ldots, x_n) &= \frac{\alpha_1 \cdot x_1}{\alpha_1} + \ldots + \frac{\alpha_n \cdot x_n}{\alpha_n} \\ &= \sum_{i=1}^{n} x_i \end{aligned}$$

Similarly, for the n-inputs multiplication with different multiplicative depths, each P_k holds the following share:

$$f_k = \frac{[\gamma_1 \times \ldots \gamma_n]_t}{\alpha_1 \times \ldots \times \alpha_n}$$

Where the shares f_k belong to a polynomial with the constant term:

$$\begin{aligned} f(x_1, \ldots, x_n) &= \frac{(\alpha_1 \cdot x_1) \times \ldots \times (\alpha_n \cdot x_n)}{\alpha_1 \times \ldots \times \alpha_n} \\ &= \prod_{i=1}^{n} x_i \end{aligned}$$

Privacy

Theorem 6. *A coalition of up to t parties corrupted by a passive adversary cannot gain any information about the private inputs, except what they can obtain from the output.*

Proof. The probabilistic functionality of hyper-invertible matrices technique ensures the privacy of the double sharings $[r_i]_{t,2t}$ and, thus, the random values $\gamma_i \leftarrow x_i \cdot \alpha_i$ in the pre-processing phase. Moreover, the given for-loop in this phase which is utilized to generate the t-sharings $[\gamma_i \cdot \gamma_j]_t$ preserves the privacy against a coalition of t semi-honest parties. Hence, the pre-processing phase outputs the sharings $[\gamma_1 \times \ldots \times \gamma_n]_t$ which are t-private.

Without loss of generality, let assume that a coalition of t participants (not including P_1) corrupted by a passive adversary has the view denoted by $VIEW_{\mathcal{A}}$. For the gate of n-inputs addition in the computation phase, $VIEW_{\mathcal{A}}$ is:

$$f_2 = \frac{[\gamma_1]_t}{\alpha_1} + \ldots + \frac{[\gamma_n]_t}{\alpha_n}$$
$$f_3 = \frac{[\gamma_1]_t}{\alpha_1} + \ldots + \frac{[\gamma_n]_t}{\alpha_n}$$
$$\vdots$$
$$f_{t+1} = \frac{[\gamma_1]_t}{\alpha_1} + \ldots + \frac{[\gamma_n]_t}{\alpha_n}$$

Where, based on the threshold feature of Shamir's secret sharing, the term $\frac{[\gamma_1]_t}{\alpha_1}$ is private against $VIEW_{\mathcal{A}}$.

Furthermore, $VIEW_{\mathcal{A}}$ for the n-inputs multiplication gate is:

$$f_2 = \frac{[\gamma_1 \times \ldots \times \gamma_n]_t}{\alpha_1 \times \ldots \times \alpha_n}$$
$$f_3 = \frac{[\gamma_1 \times \ldots \times \gamma_n]_t}{\alpha_1 \times \ldots \times \alpha_n}$$
$$\vdots$$
$$f_{t+1} = \frac{[\gamma_1 \times \ldots \times \gamma_n]_t}{\alpha_1 \times \ldots \times \alpha_n}$$

Where, again based on the threshold feature of Shamir's secret sharing, $VIEW_{\mathcal{A}}$ can gain no information about the free term of the sharings $[\gamma_1 \times \ldots \times \gamma_n]_t$ generated in the pre-processing phase. Hence, the proposed protocol is secure against a passive adversary corrupting a coalition of t parties.

3.4 Conclusion

Multiplication gates with different multiplicative depths are the bottleneck in MPC systems. we present a protocol with information-theoretic security to compute a share of a simultaneous n-inputs multiplication gate with different multiplicative

depths in a MPC circuit [60]. Indeed, our protocol can be used to compute a share of the n-inputs addition and the n-inputs multiplication in parallel in only one online round of communication. This would result a fast computation technique since a share of any given function can be computed in one online communication round which gives an improvement on the round complexity of current MPC systems, e.g., the renowned Beaver's scheme [6] or the protocol given in the Chapter 2 [58]. Our protocol has two phases: offline pre-processing phase and online computation phase. We employ the technique of hyper-invertible matrices introduced in [8] in the pre-processing phase to generate private pre-computed t-sharings which can be utilized in the computation phase. Our scheme is secure against a coalition of t participants corrupted by a passive adversary and the communication complexity for computing a share of the n-inputs multiplication gate is $O(n^2)$ field elements.

Chapter 4

Efficient Distributed Keys Generation of Threshold Paillier Cryptosystem

4.1 Introduction

As it was mentioned before in Chapter 1, MPC has various areas of applications in real-world cryptography. Such a notable example is the secure keys generation of a cryptosystem without a trusted dealer in the distributed system which is called threshold cryptography.

In threshold cryptography, all the parties are required to participate and cooperate in the system to perform a secure cryptographic computation. One may think of using a trusted dealer to generate public and private keys of the encryption system. It is a trivial task as the dealer publishes the public key and distributes the private key among the participants such that all the qualified parties collaboratively can decrypt the ciphertext. However, this system cannot be reliable and secure in practice, since all the secret information can be leaked, changed or even deleted by an adversary carrying out a single point of attack on the dealer. Thus, the important notion of distributed key generation is the solution to generate the keys in a form of the secret shared among the participants such that it would not be available in a single location.

4.1.1 Background

Generating an RSA modulus N (which is the product of two prime numbers) in a distributed fashion has been an important research topic in threshold cryptography. It is the core of many cryptographic protocols in which the computations

are executed without giving any information about the factorization of N to less than a number of threshold participants in the system. Numerous number of studies have been undertaken in the field of distributed RSA key generation, see e.g. [19, 44, 45, 91, 36, 37, 66, 52]. Boneh and Franklin (1997) [19] proposed the first RSA key generation for a two-party setting. Their protocol was secure against a passive adversary and they used a trusted third party for the security purpose. [91] employed pro-active secret sharing and a computationally bounded verifiable secret sharing in their RSA key generation protocol. A robust protocol with honest majority was suggested by [45] using the technique of secret sharing over the integers. [36] proposed a threshold RSA scheme with an efficient security method of zero-knowledge proofs, based on the hardness of discrete logarithm, where the modulus N is a product of two safe primes. Nevertheless, their protocol requires an honest third party to generate and distribute the signature keys. [37] presented an efficient and robust protocol with honest majority that the cost of going from passive security to active security is a constant factor and any fault or malicious behaviour can be detected. However, the cost of this efficiency is a simplification assumption in which their protocol is just limited to the number of three parties.

Paillier cryptosystem [88], due to its additive homomorphic feature, is an important building block of many cryptographic frameworks, see e.g. [31, 12, 5, 39, 12]. It has the same public key and the ciphertexts' algebraic structure as the RSA encryption scheme, however, the private key and the decryption procedure of the Paillier encryption do not follow from the RSA system. Therefore, a different type of distributed keys generation technique is required for the threshold Paillier cryptosystem. Nishide and Sakurai (2010) [86] conducted the first study of the Paillier's distributed keys generation and proposed a protocol with the honest majority. They employed the multiparty computation method of [9] and the Pedersen's verifiable secret sharing, which is based on the hardness of discrete logarithm, such that the

protocol holds the security against an active adversary corrupting the minority of the parties. Their protocol has the private point-to-point communication complexity $O(n^2)$ field elements, determined by the factor $6n^2$, and broadcasts $O(tn)$ field elements for the process of the shares verification where t is the threshold number of the parties, and at least $n \geq 2t+1$ participants are required to generate the keys. An important question remaining here is that whether can the keys generation of the Paillier's system be improved? Recently, [66] presented a system for threshold keys generation of the RSA and Paillier's cryptosystems in the two-party setting. Their protocol maintains the security against an active adversary using the commitment scheme (zero-knowledge proof) of ElGamal encryption.

4.1.2 Our Contribution

In this Chapter, we present a scheme for distributed keys generation of the threshold Paillier encryption system without a trusted dealer [59]. Our protocol maintains the full security against a static active adversary corrupting at most t participants where only one honest party is required to detect any malicious behaviour using message authentication codes. Furthermore, the keys are t-private, i.e. any set of equal or less than t parties cannot obtain any information about the factorization of N. Our model consists of two phases, an offline pre-processing phase and an online computation phase, where the offline phase can be implemented at any time before running the actual online phase. The offline phase is executed just for one time in the whole protocol and random shares of a triple, computed in this phase, can be used for the generation process of both the public and private keys. This idea allows us to give a faster computation phase than the protocol of [86] as the computation of the random preprocessed information can be carried out without the need of the inputs required for the keys generation. The communication complexity of our protocol is bounded to $O(n^2)$ field elements without any broadcast communication, and

the required number of participants in the online phase is just $n \geq t+1$ giving the improvements on the scheme of [86].

To achieve these goals, we employ the technique of hyper-invertible matrices, presented in section 3.2.3 , to generate the t-sharings of a random triple in the offline phase [8]. One may think of using the Beaver's multiparty computation scheme [6] to generate the threshold sharings of the keys, however, we propose another multiparty computation approach with the same communication overhead/rounds as the Beaver's scheme which was presented in Chapter 2 of this thesis [58]. Moreover, we utilize the distributed biprimarily test for an RSA modulus, given in [19], and we propose a non-interactive zero-knowledge proof technique to make the players commit to their shares in this test.

The remaining structure of this Chapter is designed as follows: Section 4.2 describes the required materials for our protocol. Section 4.3 presents the actual scheme with the security proofs. Finally, section 4.4 gives the conclusion.

4.2 Preliminaries

4.2.1 Secret Sharing Over the Integers

As we previously mentioned in section 1.1, the normal Shamir's secret sharing is threshold scheme with unconditional security. We also use the secret sharing over the integers, as a variant of [96], which was first introduced by [44] and modified by [45]. In this scheme, the secret s must be $s \in [0, I]$ where I is the interval for s. The free constant term $a_0 = \Delta \cdot s$ and the integer coefficients a_j, $1 \leq j \leq t$, are randomly chosen from the interval $a_j \in [0, K\Delta^2 I]$ where $\Delta = n!$ and $K = 2^\sigma$. Note that σ is the statistical security parameter and K is chosen such that $1/K$ is negligible. Each party P_i (for $i = 1, 2, \ldots, n$) is given the share $f_i \leftarrow f(i)$, and to reconstruct the secret a set of at least $t+1$ participants pools their shares and compute the free

constant term as:

$$\Delta \cdot f(0) = \Delta \cdot \sum_{i=1}^{t+1} f_i \cdot l_{0,i}$$

where $l_{0,i}$ is the Lagrange coefficient of P_i. Finally, since $\Delta \cdot f(0) = \Delta^2 s$, the parties calculate $s = \frac{\Delta \cdot f(0)}{\Delta^2}$.

Clearly, the secret s cannot be leaked to any subset of less than $t+1$ parties with σ bits statistical security for the this variant of secret sharing. Furthermore, this method is linear meaning that a player can compute a share of any linear function with no interaction.

We denote the $[s]_t^Z$ as a set of the shares generated in the setting of the secret sharing scheme over the integers with the degree t and the secret s. Without loss of generality, the secret addition of two values s and r distributed by the normal secret sharing $[s]_t$ and the secret sharing over the integers $[r]_t^Z$ can be reconstructed by having the parties pool their shares and compute it over the field as:

$$s + r = \frac{1}{\Delta^2} \sum_{i=1}^{n} ([\Delta^2 \cdot s]_t + [\Delta \cdot r]_t^Z) \times l_{0,i}$$

4.2.2 Threshold Paillier Cryptosystem with a Trusted Dealer

The Paillier cryptosystem [88] is a public key encryption system which holds semantic security according to the decisional composite residuosity (DCR) assumption. This assumption implies that given a ciphertext encrypted under the problem of DCR, a probabilistic polynomial time adversary has a negligible advantage to guess the corresponding plaintext [34]. More formally:

Definition 6. *Let x_0 and x_1 be encrypted under a k-bits public key encryption system based on the problem of DCR assumption. Suppose a probabilistic polynomial time adversary $\mathcal{A}$ obtains an encryption of x_β for a random $\beta \in \{0,1\}$. Let $\mathcal{A}$ can guess the values x_0 and x_1 with the probabilities $p_0(\mathcal{A}, k)$ and $p_1(\mathcal{A}, k)$, respectively. The encryption system is semantically secure, if $|p_0(\mathcal{A}, k) - p_1(\mathcal{A}, k)| \leq \varepsilon$ where ε is negligible in k.*

We now describe the algorithm of threshold Pailler encryption system where a trusted party generates and distributes the keys.

Keys Generation: A trusted dealer invokes a probabilistic algorithm $\mathsf{Gen}(1^k)$ to generate a pair of the keys $(pk, sk) \leftarrow \mathsf{Gen}(1^k)$. The public key is an RSA modulus $pk \leftarrow N$ where $N = p \cdot q$ and $\gcd(N, \phi(N)) = 1$ such that p and q are two safe primes, i.e., $p = 2p' + 1$ and $q = 2q' + 1$ where p' and q' are prime numbers as well. That is because to make sure that there is a sufficiently large number of generators in the cyclic group of N^2 [86].

The private key sk is the Euler's totient $sk = p' \cdot q' = \phi(N)/4$ where $\phi(N) = (p-1) \cdot (q-1)$. The dealer chooses a random $\beta \in Z_N^*$, and he masks the private key as $\theta = \beta \cdot \phi(N)/4 \ mod \ N$ and distributes the t-sharings over the integers $[\theta]_t^Z$. Also, θ is added to the public key, i.e. $pk = (N, \theta)$.

Encryption: A probabilistic algorithm $\mathsf{Enc}_{\mathsf{pk}}(m, r)$ is invoked to encrypt a plaintext $m \in Z_N$ and compute the ciphertext $c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m, r)$ as follows:

$$\mathsf{Enc}_{\mathsf{pk}}(m, r) = g^m \cdot r^N \ mod \ N^2$$

where the simplest value for g is $g = N + 1$ an element in $Z_{N^2}^*$ and r is a random number in Z_N^*.

Decryption: The parties execute the deterministic algorithm $\mathsf{Dec}_{\mathsf{sk}}(c)$ to decrypt and obtain the plaintext $m \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c)$. To achieve the threshold decryption, each party P_i computes the decryption share $c_i = c^{2\Delta \cdot [\theta]_t^Z} \ mod \ N^2$ and publishes it. He also makes a proof of correct commitment to his share using the zero-knowledge proof technique described in [43]. The parties compute the plaintext as follows:

$$m = L(\prod_{i=1}^{t+1} c_i^{2l_{0,i}} \ mod \ N^2) \cdot (4\Delta^2 \cdot \theta)^{-1} \ mod \ N$$

where the function $L(x)$ is defined as $L(x) = \frac{x-1}{N}$.

4.2.3 Message Authentication Code

The message authentication code (MAC) is an information-theoretic method to authenticate an output in a multiparty computation system. The output can be manipulated by an active adversary in the form of a shared secret in the system. Since this method offers a more efficient computationally unbounded security compared to other verifiable secret sharing schemes, it has been used in multiparty computation systems to detect any inconsistency or malicious behaviour, see e.g. [40, 35, 70, 5]. A prover party sending a message m calculates the MAC value, denoted by $\gamma_\alpha(m)$, as $\gamma_\alpha(m) = \alpha \cdot m$ in the field of the computation where α is the MAC key generated by the verifier. The verifier party accepts m if the equation of the MAC value is correct, otherwise outputs *fail* representing the detection of dishonest behaviour and the protocol is aborted. Clearly, this method is linear and parties can use a global MAC key α, as an additive secret of the each player's random key α_i, to verify the computation output. In this case, the probability of cheating ε without being detected is equivalent to guessing the global MAC key α over the field, i.e., $\varepsilon = 1/\mathbb{F}$. More formally:

Definition 7. *A MAC scheme with the key space $\mathcal{K}$ is ε-secure to validate an output m in the field $\mathbb{F}$, where there is a computationally unbounded adversary $\mathcal{A}$ deviating from the system to compute an output m' such that:*

$$\{\forall \alpha \in \mathcal{K}, m \in \mathbb{F},\ \exists\, m', m' \leftarrow \mathcal{A}|\ \ Pr[\gamma_\alpha(m) = \alpha \cdot m' \wedge m \neq m'] < \varepsilon\}$$

4.2.4 Security

We present the security of our scheme based on the ideal/real models. The ideal model achieves the highest level of security as there exists no type of adversary in this model. It is assumed a simulator $\mathcal{S}$ takes inputs from the participants and executes the functionality $\mathcal{F}$ such that the players do not interact directly with each

other. The model is denoted by $IDEAL_{\mathcal{F},\mathcal{S}}$. On the other hand, the participants implement the protocol Π in the presence of a computationally bounded adversary $\mathcal{A}$ in the real model denoted by $REAL_{\Pi,\mathcal{A}}$. The protocol Π is said to be secure, if the ideal model $IDEAL_{\mathcal{F},\mathcal{S}}$ and the real model $REAL_{\Pi,\mathcal{A}}$ are computationally indistinguishable [76]. This implies that $\mathcal{S}$ simulates the adversary in the ideal model by trying to change the actual output without being detected.

We assume there exists a static active adversary in our scheme which intends to deviate from the protocol and change the outcome in the fashion of malicious behaviour. A static (non-adaptive) adversary corrupts the players before running the scheme and the corrupt players do not change during the execution of our scheme. The correctness and the privacy of our scheme is maintained against at most t corrupt parties in the presence of at least $n \geq t + 1$ participants.

4.3 Our Distributed Keys Generation Scheme

In this section, we present our scheme to generate the public and private keys of the threshold Paillier cryptosystem in the distributed fashion. Our method includes two offline and online phases, where the public key and the shares of the private key are computed and verified in the actual online computation phase in the presence of only $n \geq t + 1$ participants.

4.3.1 Pre-processing Phase

This phase can be executed at any time before running the actual online phase and it is needed only once for the generation process of the both keys. We use the technique of hyper-invertible matrices, described in the section 3.2.3, to generate the shares of a random triple in the presence of at least $2t + 1$ parties where the majority is honest. Any inconsistency of the computation can be detected by the MAC scheme. The participants do not reveal the global MAC key as they locally

calculate the checking shares to authenticate the output.

Note that the computations of this phase and the public key generation must be in the order of a field greater than the public key N. Hence, according to [86], the participants pick a large prime number P such that at least $P > [n(3 \times 2^{k-1})]^2 > 2N$ where n is the number of participants and k is the security parameter which is also used to determine the range of two primes p and q for the generation of the public key. It is recommended to choose a large value for k such that the bit length of P is at least greater than 1024 bits [99]. Figure 4.1 shows the protocol Π_{Triple} to generate and verify t-sharings of a random triple.

Output: Each Party P_i obtains t-sharings $[a]_t$, $[b]_t$ and $[c]_t$ of the triple $a \cdot b = c$.

- Each player P_i (for $i = 1, \ldots, 2t+1$) generates a random MAC key α_i in $\mathbb{F}_P$ and sends it to the other participants P_j $(j \neq i)$.
- Given the sharings $[a']_t$, $[b']_t$ and $[s]_{t,2t}$ of three random values a', b' and s in the order of $\mathbb{F}_P$, each P_i locally computes the sets of the t-sharings $[a]_t$, $[b]_t$ and the double-sharings $[r]_{t,2t}$ using the hyper-invertible matrix M as:

$$([a]_t, [b]_t, [r]_{t,2t}) = M([a']_t, [b']_t, [s]_{t,2t})$$

- P_i computes a random $2t$-sharing as follows:

$$[a]_t \cdot [b]_t - [r]_{2t} = [c - r]_{2t}$$

and a set of at least $2t+1$ participants opens the value $c - r$.

- Each P_i computes the new t-sharing of c as:

$$[c]_t = c - r + [r]_t$$

Verification

- The participants repeat the steps above with another round of hyper-invertible matrix technique to compute new random double sharings $[r']_{t,2t}$ from random given sharings $[s']_{t,2t}$. Also, each party calculates the global MAC key α as the additive secret of the random MAC keys $\alpha = \sum_{i=1}^{n} \alpha_i$ and computes the random $2t$-sharing:

$$\alpha \cdot [a]_t \cdot [b]_t - \alpha[r']_{2t} = \alpha \cdot [c - r']_{2t}$$

  and they obtain the value $\alpha \cdot (c - r')$. Each P_i computes a MAC value share of c as follows:

$$[\gamma(c)]_t = \alpha \cdot (c - r') + \alpha \cdot [r']_t$$

- Finally, to validate the shares of the triple, P_i locally calculates:

$$[\sigma(c)]_t = [\gamma(c)]_t - \alpha \cdot [c]_t$$

  and a set of at least $t+1$ parties pools their shares $[\sigma(c)]_t$ and reconstructs $\sigma(c)$. The shares of the triple are consistent iff $\sigma(c) = 0$, otherwise the protocol *fails* and outputs $\perp$.

Figure 4.1 : The protocol Π_{Triple} for generating t-sharings of a random triple

Theorem 7. *The protocol Π_{Triple} is unconditionally secure against a static active adversary $\mathcal{A}$ corrupting up to t parties with small probability of error.*

Proof. Let H and C represent the honest and corrupted parties in the ideal model, respectively. Suppose $\{P_1, \ldots, P_t\} \in C$ and $\{P_{t+1}, \ldots, P_n\} \in H$. The simulator $\mathcal{S}$ sends the list of the corrupted parties to the functionality. Also, $\mathcal{S}$ picks random values $[c']_t$, α' and $[\gamma'(c)]_t$ for the inputs of the corrupted parties. This is analogous to the condition where $\mathcal{A}$ introduces the errors δ_c, δ_α and $\delta_{\gamma(c)}$ to the real model which can be denoted as $[c']_t = [c]_t + \delta_c$, $\alpha' = \alpha + \delta_\alpha$ and $[\gamma'(c)]_t = [\gamma(c)]_t + \delta_{\gamma(c)}$, respectively. $\mathcal{S}$ executes the functionality and the honest parties detect any inconsistency in the system with the probability $1 - 1/\mathbb{F}_P$ since $\sigma'(c) \neq 0$. Therefore, the ideal and the real models are computationally indistinguishable.

□

The communication complexity of the protocol Π_{Triple} is linear $O(n)$. Now, the participants need to check the multiplication correctness of the triple generated in this protocol. This can be achieved by sacrificing another random triple [35]. Figure 4.2 illustrates the protocol $\Pi_{\text{CheckTriple}}$ to check the multiplication correctness of the triple. If the check is passed successfully, each party holds the t-sharings $[a]_t$, $[b]_t$ and $[c]_t$ as the outputs of the offline phase.

- Another set of triple t-sharings $[f]_t$, $[g]_t$ and $[h]_t$ (where $f \cdot g = h$) are generated by having the participants implement the protocol Π_{Triple} once again. Also the players pick a random value t_r in the field.
- Each party computes the new t-sharings $[\rho]_t = t_r \cdot [a]_t - [f]_t$ and $[\tau]_t = [b]_t - [g]_t$. At least $t + 1$ participants obtain the random values ρ and τ.

- P_i locally calculates a checking share as follows:

$$[\sigma_{Check}]_t = t_r \cdot [c]_t - [h]_t - \tau \cdot [f]_t - \rho \cdot [g]_t - \tau.\rho$$

and the players open σ_{Check}. The check is *OK* iff $\sigma_{Check} = 0$, otherwise the protocol *fails* and outputs $\perp$.

Figure 4.2 : The protocol $\Pi_{\text{CheckTriple}}$ for checking the multiplication correctness of the triple

4.3.2 Online Phase

The public and private keys of the threshold Paillier cryptosystem are computed and verified in this phase. A number of $n \geq t + 1$ participants are able to perform this phase which is an improvement on the scheme of [86] where at least $2t + 1$ parties are required to generate the keys. Moreover, we use the MAC scheme to authenticate the keys output which is less expensive than the protocol of [86] using the expensive Pedersen's VSS based on the hardness of discrete logarithm.
Note that one can employ the Beaver's scheme [6] to compute the shares of the keys, however, we use our MPC approach described in Chapter 2 [58] which has the same efficient communication overhead and reconstruction rounds as the Beaver's scheme.

Distributed Public Key Generation

Inspired by the study of [19], the participants collaboratively generate two primes p and q using the notion of Blum integers, since about 1/4 of all RSA modulus are Blum integers. The pro-active secret sharing method allows the parties to redistribute the shares of the two primes such that no less than t players can gain their actual values, i.e., the factorization of N remains t-private. Figure 4.3 shows the protocol Π_{pk} to generate the public key of the Paillier's encryption system. The com-

putations are performed in the order of P as it described in the pre-processing phase.

Output: A set S of $n \geq t+1$ participants obtains a public key N modulo P.

- The participants choose random values $p_i,\ q_i \in [2^{k-1}, 2^k - 1]$ where k, as described in the section 4.3.1, is the security parameter such that the party P_1 calculates $p_1 = q_1 = 3 \mod 4$ and the every other party, except P_1, picks $p_i = q_i = 0 \mod 4$. Each party P_i distributes the t-sharings $[p_i]_t$ and $[q_i]_t$ in the order of P.

- Player P_j $(j \neq i)$ generates a random MAC key $\alpha_j \in Z_P$, and sends it to the party P_i . P_i replies with the MAC values $\gamma_{\alpha_j}([p_i]_t) = \alpha_j \cdot [p_i]_t$ and $\gamma_{\alpha_j}([q_i]_t) = \alpha_j \cdot [q_i]_t$.

- P_j accepts the shares $[p_i]_t$ and $[q_i]_t$ iff $\gamma_{\alpha_j}([p_i]_t) = \alpha_j \cdot [p_i]_t$ and $\gamma_{\alpha_j}([q_i]_t) = \alpha_j \cdot [q_i]_t$, otherwise he broadcasts *fail* and the protocol is aborted.

- After verifying the initial shares, each party $P_i \in S$ computes the new t-sharings of the primes using one round of pro-activization as $[p]_t = \sum_{i=1}^{n} [p_i]_t$ and $[q]_t = \sum_{i=1}^{n} [q_i]_t$.

- P_i calculates the random sharings $[y]_t = [p]_t + [a]_t$ and $[z]_t = [q]_t - [b]_t$. The parties obtain the values y and z.

- Each party calculates the global MAC key α as the additive secret of the random MAC keys $\alpha = \sum_{i=1}^{n} \alpha_i$. Then, each party $P_i \in S$ computes the MAC value shares of y and z as:

$$\gamma([y]_t) = \alpha([p]_t + [a]_t)$$

$$\gamma([z]_t) = \alpha([q]_t - [b]_t)$$

- P_i locally calculates the t-sharings $[\sigma(y)]_t = \gamma([y]_t) - \alpha[y]_t$ and $[\sigma(z)]_t = \gamma([z]_t) - \alpha[z]_t$. The parties open the checking values $\sigma(y)$ and $\sigma(z)$. The protocol *continues* iff $\sigma(y) = 0$ and $\sigma(z) = 0$, otherwise it *fails* and is aborted.

- P_i computes a share of the public key N as follows:

$$[N]_t = \frac{y \cdot ([q]_t + [b]_t) + z \cdot ([p]_t - [a]_t)}{2} - [c]_t$$

  and a set of at least $t + 1$ participants obtains the public key N. Each $P_i \in S$ computes a MAC value share of the public key as follows:

$$\gamma([N]_t) = \frac{\alpha \cdot y \cdot ([q]_t + [b]_t) + \alpha \cdot z \cdot ([p]_t - [a]_t)}{2} - \alpha \cdot [c]_t$$

- P_i locally computes:

$$[\sigma(N)]_t = \gamma([N]_t) - \alpha[N]_t$$

  and the parties open $\sigma(N)$. The public key output N is *OK* iff $\sigma(N) = 0$, otherwise the protocol *fails* and outputs $\perp$.

Figure 4.3 : The protocol Π_{pk} for distributed public key generation of the Paillier cryptosystem

Theorem 8. *The protocol* Π_{pk} *maintains statistical security against a static active adversary* $\mathcal{A}$ *which corrupts at most* t *participants with low probability of error.*

Proof. Without loss of generality, suppose $\{P_1, \ldots, P_t\} \in C$ and $P_{t+1} \in H$ where C and H represent the sets of corrupt and honest parties in the ideal model, respectively. The simulator $\mathcal{S}$ sends a list of corrupt parties to the functionality. Also, $\mathcal{S}$

sends the random values $[p'_i]_t$, $[q'_i]_t$ and α'_{t+1} to the functionality which simulate the errors $[p'_i]_t = [p_i]_t + \delta_p$, $[q'_i]_t = [q_i]_t + \delta_q$ and $\alpha'_{t+1} = \alpha_{t+1} + \delta_{\alpha_{t+1}}$ by $\mathcal{A}$ in the real model. Any inconsistency in the initial shares can be detected with small probability of error. $\mathcal{S}$ chooses random sharings $[y']_t$, $[z']_t$ and the random MAC key α' and sends them to the functionality. This is analogous to the condition where $\mathcal{A}$ introduces the errors $[y']_t = [y]_t + \delta_y$, $[z']_t = [z]_t + \delta_z$ and $\alpha' = \alpha + \delta_\alpha$ to the real model. The functionality is implemented and P_{t+1} can detect any malicious behaviour for obtaining y and z with the probability $1 - 1/\mathbb{F}_P$. Finally, $\mathcal{S}$ picks random $[N']_t$ and $\gamma'([N]_t)$ which can be considered as simulating the errors $[N']_t = [N]_t + \delta_N$ and $\gamma'([N]_t) = \gamma([N]_t) + \delta_{\gamma_N}$ in the real model. $\mathcal{S}$ executes the functionality and P_{t+1} detects any malicious behaviour in the computation with the error probability $1/\mathbb{F}_P$. Hence, the ideal and the real models are statistically indistinguishable with the security parameter k.

If $\{P_1, \ldots, P_{t+1}\} \in H$, the participants open the random values $y = p + a$ and $z = q - b$. Each party computes a share of the public key $[N]_t$ and the parties pool their shares and obtain the public key which can be written as:

$$\begin{aligned} N &= \frac{(p+a)\cdot(q+b) + (q-b)\cdot(p-a)}{2} - c \\ &= \frac{2(N+c)}{2} - c \end{aligned}$$

□

The total communication complexity for generating the Paillier's public key in the protocol Π_{pk} is $O(n^2)$ field elements with no broadcast communication which improves on the total communication overhead of [86] which is bounded to the private communication complexity $O(n^2)$ plus the broadcast coomunication overhead $O(tn)$. The public key N needs to be checked for small prime divisors up to some upper bound B. According to [99], it is more efficient in practice to check for the small prime divisors after computing N instead of checking the individual primes p

and q for it. This implies that N must be checked for dividing to any prime divisor smaller than B.

Distributed Biprimarily Test

The participants need to check the multiplication correctness of N that whether it is a product of two primes p and q without revealing the primes. [19] gave a scheme for this test using the Euler's theorem. We propose a technique of non-interactive zero-knowledge proof to make the participants commit to the values they reveal for this test. Figure 4.4 presents the protocol $\Pi_{\text{Biprimarily}}$ for distributed biprimarily test of N. Note that each participant already holds the random values p_i and q_i and their shares from the protocol Π_{pk}.

- The participants pick a random generator $g \in Z_N^*$ such that Jacobi $(\frac{g}{N}) = 1$.
- Party P_1 computes $\phi_1(N) = N + 1 - p_1 - q_1$ and every other participant, except P_1, calculates $\phi_i(N) = -(p_i + q_i)$. Each party P_i computes $\nu_i = g^{\phi_i(N)/4} \mod N$ and reveals it.
- Each $P_i \in S$ proves that he has committed to a correct value of ν_i. Namely, all parties conduct a non-interactive zero-knowledge proof to verify ν_i as follows:
  - Every P_j $(j = 1, \ldots, n)$ computes $d_j = g^{1/4([p_i]_t \cdot l_{0,j} + [q_i]_t \cdot l_{0,j})} \mod N$, where $l_{0,j}$ is the Lagrange coefficient of the party P_j and $[p_i]_t$ and $[p_i]_t$ are the shares he received from P_i in the protocol Π_{pk}. He publishes d_j.

- The parties compute:

$$e_i = \nu_i \cdot \prod_{j=1}^{n} d_j \bmod N$$

- For the inputs commitment of P_1, all the participants examine that whether $\log_g(e_1)$ is equal to $1/4(N+1)$. If *Ok* the commitment is successful, otherwise the protocol *fails*.
- For the inputs commitment of other parties except P_1, the participants check that whether $e_i = 1$, i.e., g^0. The commitment is successful if it is *Ok*, otherwise the protocol *fails*.

The participants check that whether $\prod_{i=1}^{n} \nu_i = \pm 1 \bmod N$. If it *fails*, N is not biprime and is discarded.

Figure 4.4 : The protocol $\Pi_{\text{Biprimarily}}$ for the distributed biprimarily test of N

Correctness

For the zero-knowledge proof of each party P_i, the term $\prod_{j=1}^{n} d_j \bmod N = g^{1/4(p_i+q_i)} \bmod N$ due to the homomorphism of discrete logarithm in the base g. Since $\phi_1(N) = N + 1 - p_1 - q_1$, the party P_1 proves that he has committed to the correct value of ν_1 by having all the parties compute $e_1 = g^{1/4(N+1)} \bmod N$. For every other party except P_1, the commitment value is $e_i = g^0 \bmod N$ because $\phi_i(N) = -(p_i + q_i)$.

For the biprimarily test, note that $\phi(N) = N + 1 - p - q$ and $\prod_{i=1}^{n} \nu_i = g^{\phi(N)/4}$. Since the Jacobi $(\frac{g}{N}) = 1$ and due to the Euler's theorem, the parties check that $g^{\phi(N)/4} = \pm 1 \bmod N$.

In the case that N is not biprime, the test may fail with the probability $\frac{1}{2}$. Therefore, the test must be repeated for few m iterations to reduce the error probability which gives the probability 2^{-m} for accepting N, if N is not biprime. However, in

practice the probability that a non-biprime N passes even one iteration of this test is actually much less than $\frac{1}{2}$ [19].

Distributed Private Key Generation

The participants start this stage while they are holding the public key N and the additive shares of the private key $\phi(N)$ from the protocol Π_{pk}. Recall that the idea is to mask the private key by a random number $\beta \in Z_N^*$ i.e., $\theta = \beta \cdot \phi(N)/4$ (see section 2.2). Note that the private key can be written as $\phi(N) = N - p - q + 1$. We use the similar multiparty computation approach to the protocol Π_{pk} to compute the t-sharings of θ. Figure 4.5 shows the protocol Π_{sk} for the distributed generation of the threshold Paillier's private key.

- **Output**: Each party of a set S of $n \geq t+1$ participants computes the t-sharing of the masked private key $[\theta]_t^Z$ over the integers.
  - Each party $P_i \in S$ generates a random number $\beta_i \in Z_N^*$. He distributes the t-sharings over the integers $[\phi_i(N)]_t^Z$ and $[\beta_i]_t^Z$ among the participants.
  - Every other player P_j $(j \neq i)$ picks a random MAC key $\alpha_j \in Z_N^*$ and sends it to P_i who replies with the corresponding MAC values of the shares as $\gamma_{\alpha_j}([\phi_i(N)]_t^Z) = \alpha_j \cdot [\phi_i(N)]_t^Z$ and $\gamma_{\alpha_j}([\beta_i]_t^Z) = \alpha_j \cdot [\beta_i]_t^Z$.
  - P_j accepts the shares $[\phi_i(N)]_t^Z$ and $[\beta_i]_t^Z$ iff $\gamma_{\alpha_j}([\phi_i(N)]_t^Z) = \alpha_j \cdot [\phi_i(N)]_t^Z$ and $\gamma_{\alpha_j}([\beta_i]_t^Z) = \alpha_j \cdot [\beta_i]_t^Z$, otherwise the protocol *fails*.
  - Each P_i calculates the new t-sharings of the secrets by employing one round of pro-activization as $[\phi(N)]_t^Z = \sum_{i=1}^n [\phi_i(N)]_t^Z$ and $[\beta]_t^Z = \sum_{i=1}^n [\beta_i]_t^Z$.

- P_i calculates the random t-sharings $[y']_t = [\phi(N)]_t^Z + [a]_t$ and $[z']_t = [\beta]_t^Z - [b]_t$, and the parties open the random values y' and z'.

- Every party calculates the global MAC key $\alpha = \sum_{i=1}^{n} \alpha_i$. Each $P_i \in S$ computes the MAC value shares of y' and z' as:

$$\gamma([y']_t) = \alpha([\phi(N)]_t^Z + [a]_t)$$

$$\gamma([z']_t) = \alpha([\beta]_t^Z - [b]_t)$$

- Every party locally calculates the t-sharings $[\sigma(y')]_t = \gamma([y']_t) - \alpha[y']_t$ and $[\sigma(z')]_t = \gamma([z']_t) - \alpha[z']_t$, and the participants obtain the checking values $\sigma(y')$ and $\sigma(z')$. The protocol *fails* iff $\sigma(y) \neq 0$ or $\sigma(z) \neq 0$, otherwise it *continues*.

- P_i computes a share of θ over the integers as follows:

$$[\theta]_t^Z = \frac{y' \cdot ([\beta]_t^Z + [b]_t) + z' \cdot ([\phi(N)]_t^Z - [a]_t)}{2} - [c]_t$$

- Also, P_i computes a MAC value share of θ over the integers as:

$$\gamma([\theta]_t^Z) = \frac{\alpha \cdot y' \cdot ([\beta]_t^Z + [b]_t) + \alpha \cdot z' \cdot ([\phi(N)]_t^Z - [a]_t)}{2} - \alpha \cdot [c]_t$$

- P_i locally computes:

$$[\sigma(\theta)]_t^Z = \gamma([\theta]_t^Z) - \alpha[\theta]_t^Z$$

and a set of at least $t+1$ participants reconstructs $\sigma(\theta)$. The generation of the private key shares is *OK* iff $\sigma(\theta) = 0$, otherwise the protocol is aborted and outputs $\perp$.

Figure 4.5 : The protocol Π_{sk} for the private key generation of the threshold Paillier system

Theorem 9. *The protocol* Π_{sk} *is statistically secure against a static active adversary* $\mathcal{A}$ *corrupting up to* t *parties with the negligible probability of error.*

Proof. The security proof follows the proof of the protocol Π_{pk}. □

Recall that the final step of the threshold decryption is to reveal the public key $\theta = \beta \cdot \phi(N) \bmod N$ (see section 4.2.2), however, the participants hold the t-sharings $[\theta]_t^Z$ over the integers. Thus, in order to deal with this issue, the parties transform the t-sharings $[\theta]_t^Z$ to the normal t-sharings $[\Delta^2 \cdot \theta]_t$ in the order of Z_N^*, by locally reducing the shares modulo N, and then they pool their new t-sharings to reconstruct θ in Z_N^* [99].

4.4 Conclusion

Distributed keys generation of encryption systems without a trusted dealer has been an important topic in the field of threshold cryptography. In this Chapter, we give an efficient scheme for distributed keys generation of the threshold Paillier cryptosystem using multiparty computation [59]. Our protocol has two offline and online phases. We employ the technique of hyper-invertible matrices to generate random t-sharings of a triple in the offline phase which can happen at any time before the actual online computation. Also, these shares are authenticated and the multiplication correctness of the triple is checked. The public and the private keys are computed and verified in the presence of at least $n \geq t + 1$ participants which gives an improvement on the scheme of [86] where at least $2t + 1$ parties are required for that purpose. Moreover, a distributed biprimarily test with a technique of non-interactive zero-knowledge proof, to check the commitment of the players' inputs, is presented to examine the correctness of the public key factorization.

Our scheme preserves the statistical security against a non-adaptive active adversary corrupting at most t participants with the low probability of error using message

authentication codes. Furthermore, the computed keys are t-private. The private communication complexity to generate the keys is $O(n^2)$ field elements with no broadcast communication overhead which also improves on the protocol of [86] where the private communication complexity is the same as our scheme but the broadcast overhead is (nt).

Chapter 5

Verifiable DOPE from Somewhat homomorphic Encryption, and the Extension to DOT

5.1 Introduction

As described in section 1.3.1, distributed oblivious polynomial evaluation (DOPE) is a variant of two-party computation where a sender party P_1 holds a polynomial $f(x)$ of degree k and a receiver party P_2 has a value α. They wish to perform a secure computation with the help of t distributed cloud servers over a field $\mathbb{F}_q$ such that P_2 gains the value $f(\alpha)$ while the privacy of their inputs is preserved. The system is the building structure of many cryptographic primitives and it can be denoted by the functionality $(f(x), \alpha) \rightarrow (\perp, f(\alpha))$. Using more than one cloud server $(t \geq 2)$ offers higher decentralization and security against the central point of failure attack as an adversary must corrupt more than one server to break the privacy or forge the output. Also, this system is more flexible as the main two parties do not communicate directly and they can remain anonymous to each other. Nevertheless, the security is achieved at the cost of higher communication overhead between the parties and the servers.

5.1.1 Background

In the literature, the idea of oblivious polynomial evaluation (OPE) has attracted more researches than DOPE. Some studies have utilized homomorphic feature of encryption systems to present their semantically secure OPE protocols. [49] employed additive (Paillier encryption) and multiplicative (ElGamal cryptosystem) homomorphic encryption methods (i.e., fully homomorphic encryption) to propose their OPE

protocols for the multivariate case. A simulation-based secure protocol (in ideal/real paradigm) with this technique was studied by [65] and [51] using homomorphic encryption. [47] proposed two OPE protocols for keyword search problem using homomorphic encryption and pseudorandom functions. Some studies have investigated the idea of employing one third party in their OPE protocols [64, 57, 20, 50]. [72] conducted a study to minimize the communication of a computationally-private information retrieval protocol using a single database for the datasets with large length. Recently, [49] presented a verifiable and private OPE protocol, by employing homomorphic feature of Paillier encryption and a trusted server, to record medical datasets.

There are only very few studies in the literature of DOPE. The first DOPE study was conducted by [75]. The main issue of their method is that the privacy of the parties' inputs is not held against the maximum possible number $t-1$ servers. They introduced some publicly known information to address this problem raising the communication complexity. Another private DOPE protocol was proposed by [27] with the communication complexity $O(kt)$ where their protocol requires that the main two parties P_1 and P_2 directly communicate with each other which does not meet the condition that the two parties are allowed to interact only with the cloud servers. All these studies have been conducted in the setting of passive adversary. Therefore, one may think that the need of considering an active adversary in the DOPE system (which is more practical and close to the reality) is observed.

5.1.2 Our Contribution

We aim to present the first verifiable and private DOPE protocol using additive homomorphic feature of Paillier encryption and secret sharing in the presence of n distributed cloud servers where t number out of them can be chosen to perform the computation [61]. An advantage of our protocol is that, unlike the protocol of

[26], the sender and the receiver parties do not communicate directly, i.e. they can be anonymous to each other. This interesting feature would enable the scheme to have a setup phase well in advance of the actual computation while the privacy of the inputs is preserved. Thus, the receiver essentially must not be available in the setup phase giving a more flexible system. Our protocol holds the security against a static active adversary corrupting P_1 and at most $t-1$ dishonest servers with the IND-CPA security of Paillier cryptosystem and the unconditional security of message authentication codes (MAC). The adversary tries to deviate from the protocol and forge the output of P_2. However, we assume P_2 is not controlled by the active adversary since he aims to obtain the correct output while he can still remain semi-honest. To the best of the authors' knowledge, this is the first verifiable secure DOPE protocol against a malicious sender and at most $t-1$ servers. Furthermore, it preserves information-theoretic security for the sender's inputs against a passive adversary controlling a coalition of the receiver P_2 and $t-1$ servers . This gives a security improvement on the DOPE protocol of [75] as their protocol is not privately secure against the maximum $t-1$ servers. The secure outsourced computation of the homomorphic encryption scheme is performed by the cloud servers in parallel. Our scheme has the communication complexity with the factor kt which improves on the communication overhead of [26] where it is determined by the term $2kt$.

In addition, the proposed scheme can be extended to a system of secure $\binom{1}{2}$ distributed oblivious transfer (DOT) with the same security setting and the linear communication complexity $O(k)$ which is discussed after presenting the main DOPE protocol.

5.2 Preliminaries

5.2.1 Model

We propose a secure DOPE protocol where the sender party P_1, with the polynomial $f(x)$ of degree k, and the receiver party P_2, holding a value α, aim to perform a secure computation in the presence of a threshold number of t cloud servers (security parameter) such that P_2 computes $f(\alpha)$ while the correctness and the strong privacy of the inputs are preserved. The idea is that the sender's polynomial along with its corresponding MACs are distributed among the servers using the secret sharing. The MAC keys α_i are generated by the servers and used to verify the distributed shares in the setup phase, and a global MAC key (which is the additive secret of the keys α_i) authenticates the output to P_2. Any malicious behaviour and inconsistency can be detected by having P_2 verify the output using the corresponding MAC value received from the servers. Moreover, the outsourced computation technique enables fast computation of the large values over the field. Note that in order to preserve the privacy of the sender's polynomial, the protocol with a fixed polynomial can only be implemented up to $k-1$ times by the same receiver.

This system can also be extended to secure DOT_1^2 as well, where P_1 holds two inputs, m_0 and m_1, and P_2 has $\sigma \in \{0,1\}$. By employing the extended protocol, P_2 learns m_σ while the same security condition is met. We utilize Paillier public key system to encrypt the P_2's input as this cryptosystem is the most suitable additive homomorphic encryption scheme for large datasets [65].

5.2.2 The Paillier Cryptosystem

We previously discussed the threshold version of Paillier cryptosystem in section 4.2.2. We now revisit the general version of this encryption system, the security setting and the homomorphic feature. The Paillier cryptosystem [88] works under the assumption of decisional composite residuosity (DCR) and the security of this

cryptosystem is said indistinguishability against chosen function attack (IND-CFA) under the DCR assumption. More formally:

Definition 8. *Let x_0 and x_1 be encrypted using a k-bits cryptosystem under the DCR assumption. Let a probabilistic polynomial-time adversary $\mathcal{A}$ gain the encryption of x_β for a randomly chosen $\beta \in \{0,1\}$. Suppose $\mathcal{A}$ can guess the plaintexts x_0 and x_1 with the probabilities denoted by $p_0(\mathcal{A},k)$ and $p_1(\mathcal{A},k)$, respectively. The encryption system is said to be IND-CFA secure if $|p_0(\mathcal{A},k) - p_1(\mathcal{A},k)| \leq \varepsilon$ for any ε negligible in k.*

This encryption system includes main three algorithms: keys generation, encryption and decryption.

Keys Generation: The dealer invokes a probabilistic algorithm $\mathsf{Gen}(1^k)$, with the security parameter k, to generate the keys pair $(pk, sk) \leftarrow \mathsf{Gen}(1^k)$. The public key pk is an RSA modulus $pk \leftarrow N$ where $N = p_c \cdot q_c$ such that p_c and q_c are two large prime numbers with k/2 bits, e.g., each at least 1024 bits. The private key sk is the Euler's totient $sk \leftarrow \phi(N)$ where $\phi(N) = (p_c - 1) \cdot (q_c - 1)$ such that $\gcd(N, \phi(N) = 1$.

Encryption: The dealer invokes a probabilistic algorithm $\mathsf{Enc}_{\mathsf{pk}}(m, r)$ to encrypt the message m and computes the ciphertext $c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m, r)$ as:

$$\mathsf{Enc}_{\mathsf{pk}}(m, r) = g^m \cdot r^N \mod N^2$$

where the simplest value for $g \in Z^*_{N^2}$ can be $g = N + 1$ and r is a random number chosen in Z^*_N.

Homomorphism. A very useful feature of this cryptosystem is homomorphism which can be applied to the ciphertexts. Namely, let m_1 and m_2 be two plaintexts in Z_N which are encrypted with the same public key denoted by $\mathsf{Enc}_{\mathsf{pk}}(m_1)$ and $\mathsf{Enc}_{\mathsf{pk}}(m_2)$,

respectively. It is trivial to show that $\mathsf{Enc}_{\mathsf{pk}}(m_1) \times \mathsf{Enc}_{\mathsf{pk}}(m_2) = \mathsf{Enc}_{\mathsf{pk}}(m_1 + m_2)$ and also $\mathsf{Enc}_{\mathsf{pk}}(m_1)^d = \mathsf{Enc}_{\mathsf{pk}}(d \cdot m_1)$ for any random $d \in Z_N$.

Decryption: To decrypt the ciphertext c, a deterministic algorithm $\mathsf{Dec}_{\mathsf{sk}}(c)$ is invoked to obtain the plaintext $m \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c)$. In detail, one raises the ciphertext to the private key $\phi(N)$ which, based on the Euler's totient function, can be simplified as:

$$[(N+1)^m \cdot r^N]^{\phi(N)} \mod N^2 = N \cdot \phi(N) \cdot m + 1$$

proven by the means of binomial coefficients in modulo N^2. Let the function $L(x)$ be $L(x) = \frac{x-1}{N}$, the plaintext m can be obtained as follows:

$$\mathsf{Dec}_{\mathsf{sk}}(c) = L[c^{\phi(N)} \mod N^2] \cdot \phi(N)^{-1} \mod N$$

5.2.3 Security

In this section, we discuss ideal/real security model and the strong security requirements for DOPE protocols. It is assumed that there exists a simulator $\mathcal{S}$ in the ideal model who takes inputs from the participants and implements the functionality $\mathcal{F}$, denoted by $IDEAL_{\mathcal{F},\mathcal{S}}$ such that the parties do not communicate with each other. Hence, this model doesn't have any type of adversary and it achieves the highest level of security. The view of this model is denoted by $VIEW_{\mathcal{S}}$. On the other hand, a protocol Π is executed in the presence of a probabilistic polynomial-time adversary $\mathcal{A}$ in the real model $REAL_{\Pi,\mathcal{A}}$. The view of the coalition controlled by the adversary is denoted by $VIEW_{\mathcal{A}}$. Based on the simulation-based security model, the protocol Π is implemented securely iff $IDEAL_{\mathcal{F},\mathcal{S}}$ and $REAL_{\Pi,\mathcal{A}}$ are computationally indistinguishable [76].

Suppose that P_1, with the polynomial $f(x)$, and P_2, holding a value α, aim to communicate with at least t distributed cloud servers to run a secure DOPE protocol. According to [15] and [75], the protocol must satisfy the following conditions to hold

the strong security:

- **Correctness**: After implementation of the protocol, P_2 receives $f(\alpha)$, while P_1 and a maximum threshold number of the distributed servers obtain nothing. Here, an active adversary might attack the system to manipulate the outcome computed by P_2. Thus, the system must be secure against a malicious adversary corrupting a coalition of at most $t-1$ dishonest servers in the computation phase. Note that P_2 normally does not deviate from the protocol since he intends to obtain the correct output; however, he still remains semi-honest.

- **Receiver's Privacy**: Here, the adversary $\mathcal{A}$ corrupts a coalition of up to $t-1$ servers and P_1. The protocol must not leak any information about the P_2's input α to $VIEW_{\mathcal{A}}$.

- **Sender's Privacy**: $\mathcal{A}$ controls a coalition of the maximum $t-1$ servers and P_2. **Before** and **after** running the computation phase, $VIEW_{\mathcal{A}}$ must learn nothing about the polynomial $f(x)$ except what it can get from the outcome $f(\alpha)$. Note that the protocol with the same sender must only be executed up to $k-1$ times, otherwise $\mathcal{A}$ can interpolate the sender's polynomial.

Recall the functionality of an OPE protocol, it can be extended to a DOPE protocol as follows:

$$(f(x), \underbrace{\perp, \perp, \ldots, \perp}_{t-1 \ servers}, \alpha) \rightarrow (\perp, \underbrace{\perp, \perp, \ldots, \perp}_{t-1 \ servers}, f(\alpha))$$

5.3 Our Protocol

The proposed DOPE scheme is discussed in this section. The protocol consists of two different phases: *setup* and *computation*. The sender party P_1 interacts with

the servers in the setup phase, whereas the receiver party P_2 communicates with the servers in the actual computation phase. Each phase has the verifiable stage that authenticates the inputs and the output using the MACs (described previously in section 4.2.3), and the protocol *fails* in the case of detecting any malicious behaviour. It is assumed that the private communication channels are secure and synchronous.

5.3.1 Setup Phase

This phase can be executed at anytime before running the actual computation phase. This property enhances the security and the flexibility (since P_2 is not available) of the system. In addition, each server checks the authentication of the given shares using the MAC scheme.

Assume P_1 wishes to distribute the polynomial $f(x)$ of degree k in the field Z_q among a set S of t distributed servers. Figure 5.1 shows the setup phase of the protocol Π_{DOPE}.

Inputs: P_1 has the polynomial $f(x) = a_0 + a_1x + \ldots + a_kx^k$, where $a_m \in Z_q$ for $m = 0, 1, \ldots, k$.

- Each server $S_i \in S$ (for $i = 1, \ldots, t$) generates a MAC key $\alpha_i \in Z_q$ uniformly random and sends it to P_1.

- P_1 generates $k+1$ random polynomials of the degree $t-1$ with the constant terms $a_0, a_1, \ldots, a_k$ and computes the shares $[a_m]_{t-1}$ over the integers. He performs the same process with the MAC values $\gamma_m \leftarrow \alpha_{mac} \cdot a_m$, where $\alpha_{mac} = \sum_{i=1}^{t} \alpha_i$, and computes the shares $[\gamma_m]_{t-1}$ over the integers. He gives the tuples $\langle a_{mi} \rangle$ to each server $S_i \in S$, which is defined as:

$$\{\forall S_i \in S,\ 0 \leq m \leq k : \quad \langle a_{mi} \rangle \leftarrow ([a_m]_{t-1}, \gamma_i([a_m]_{t-1}), [\gamma_m]_{t-1})\}$$

- P_1 leaves the protocol and takes no step further.

Verification

- Each S_i checks for the share $[a_m]_{t-1}$ that $\gamma_i([a_m]_{t-1}) = \alpha_i \cdot [a_m]_{t-1}$. It accepts the share iff *OK*, otherwise broadcasts *fail* and aborts the protocol.

Figure 5.1 : The setup phase of the protocol Π_{DOPE}

The communication complexity of this phase is $O(kt)$ where the number of servers t can be selected as the security parameter. Note that the number of servers t does not depend on the degree k of the sender's polynomial.

5.3.2 Computation Phase

P_2 firstly encrypts his inputs and broadcasts the ciphertexts. Each cloud server computes an encrypted share of the output using homomorphic encryption. It repeats the same process with the shares of the MACs to obtain the corresponding encrypted share of the MAC value. P_2 calculates the output and checks whether the final outcome satisfies the verification stage using the MACs he receives from the servers. Here, a static active adversary, corrupting at most $t-1$ servers, can forge

the output without being detected with the small probability of error $\varepsilon = 1/q$. Based on [15], the strong security conditions of DOT and DOPE protocols can be acquired with at least two communication rounds between the servers and P_2. Figure 5.2 illustrates the computation phase of the protocol Π_{DOPE}. In order to maintain the sender's privacy, a receiver is allowed to implement this phase only up to $k-1$ times.

Input: P_2 has the value $\alpha \in Z_q$.

Output: P_2 learns $f(\alpha)$.

- P_2 invokes the key generation algorithm $\mathsf{Gen}(1^k)$ of the Paillier cryptosystem to generate a pair of encryption keys (pk, sk) where $pk \leftarrow N$, $sk \leftarrow \phi(N)$ and $N = p_c \cdot q_c$. He encrypts the values α^m, for $m = 0, 1, \ldots, k$, to obtain the ciphertexts:

$$c_m \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\alpha^m)$$

  and broadcasts the respective ciphertexts c_m.

- Each $S_i \in S$ computes a new encrypted share:

$$c_i' = (\prod_{m=0}^{k} c_m^{[a_m]_{t-1}})^{l_{0,i}} \ mod \ N^2$$

  where $l_{0,i}$ is the Lagrange coefficient of S_i as:

$$l_{0,i} = \prod_{\substack{j \in S \\ j \neq i}} \frac{-j}{i-j} \ mod \ q$$

- Similarly, each S_i computes the encrypted share of the MAC value using another round of homomorphic encryption as follows:

$$c'_i(\gamma_{f_i}) = (\prod_{m=0}^{k} c_m^{[\gamma_m]_{t-1}})^{l_{0,i}} \ mod \ N^2$$

and it assigns them to the tuple $\langle C'_i \rangle \leftarrow (c'_i, c'_i(\gamma_{f_i}), \alpha_i)$. It privately sends $\langle C'_i \rangle$ to P_2.

- P_2 obtains the output using the encrypted shares c'_i of the servers as:

$$c' = \prod_{i=1}^{t} c'_i \ mod \ N^2$$

$$f(\alpha) \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c')$$

Verification

- P_2 calculates the global MAC key $\alpha_{mac} = \sum_{i=1}^{k} \alpha_i$.

- P_2 computes the MAC value of the output as follows:

$$c'(\gamma_f) = \prod_{i=1}^{t} c'_i(\gamma_{f_i}) \ mod \ N^2$$

$$\gamma(f(\alpha)) \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c'(\gamma_f))$$

- P_2 checks that whether:

$$\gamma(f(\alpha)) - \alpha_{mac} \cdot f(\alpha) = 0$$

and he accepts $f(\alpha)$ iff *OK*. Otherwise, he broadcasts *fail* and outputs $\perp$.

Figure 5.2 : The computation phase of the protocol Π_{DOPE}

Note that since the inputs, $f(x)$ and α, and the output $f(\alpha)$ are over the field of a prime number $\mathbb{Z}_q$ while the public key encryption and the homomorphic feature are computed in the ring $\mathbb{Z}_N$, we must determine the relation between these two fields. Therefore, in order to have the protocol work properly, it is required that $N = p_c \cdot q_c > q$; otherwise, P_2 would calculate $f(\alpha)$ mod N not $f(\alpha)$ mod q. More precisely, because each term $\alpha^m \cdot a_m$ is in in modulo q^2, N must hold the condition $N > (k+1)q^2$. It is recommended to choose two large values for the size of the prime numbers p_c and q_c, e.g., each at least 1024 bits.

It should be stated that the computation of the new encrypted shares and their corresponding MACs can be implemented by the servers in parallel which reduces the computation time of the protocol. The communication complexity is determined by the factor kt which gives an improvement of the DOPE protocols of [75] and [26].

5.4 Security Evaluation

In this section, we evaluate the protocol Π_{DOPE}, based on the security conditions described in section 5.2.3.

Theorem 10. *The protocol Π_{DOPE} maintains the security against a static active adversary $\mathcal{A}$ corrupting P_1 in the setup phase and a coalition of at most $t-1$ dishonest servers in the computation phase.*

Proof. Let C and H represent corrupt and honest parties/servers, respectively. In the ideal model, the simulator sends a list of $P_1, S_i \in C$ to the functionality. The simulator picks random values for the shares $[a'_m]_{t-1}$ in the setup phase simulating the errors $[a'_m]_{t-1} = [a_m]_{t-1} + \delta_{mi}$ by $\mathcal{A}$ in the real model. The errors can be detected using the corresponding MACs with the negligible probability of error $1/q$. In the computation phase, the simulator sends random values c''_i and $c''(\gamma_{f_i})$ which

is equivalent to introducing the errors $c_i'' = c_i' + \delta_i$ and $c''(\gamma_{f_i}) = c'(\gamma_{f_i}) + \delta_{\gamma_{f_i}}$ by $\mathcal{A}$ to the real model. The simulator runs the functionality and sends the output $f'(\alpha)$ to P_2 where $f'(\alpha) = f(\alpha) + \delta$. Similarly, the simulator calculates the global MAC key $\alpha'_{mac} = \alpha_{mac} + \delta_\alpha$ which can be considered as the MAC key cheated by $\mathcal{A}$ in the real model. Any inconsistency between the output and the MAC value $\gamma'(f(\alpha))$ is detected with the probability $1 - 1/q$ in the verification stage, i.e., there is just one value for α'_{mac} which results $\gamma'(f(\alpha)) - \alpha'_{mac} \cdot f'(\alpha) = 0$.

If $P_1, S_i \in H$, the simulator receives their inputs and runs the protocol. It computes the encrypted shares c_i' which can be written as:

$$c_i' = \mathsf{Enc}_{\mathsf{pk}}((\sum_{m=0}^{k} \alpha^m \cdot [a_m]_{t-1}) \times l_{0,i} \ mod \ N)$$

The simulator employs another round of homomorphic encryption to compute the new encrypted value c' as:

$$\begin{aligned} c' &= \mathsf{Enc}_{\mathsf{pk}}(\sum_{i=1}^{t}([a_0]_{t-1} + \alpha[a_1]_{t-1} + \alpha^2[a_2]_{t-1} + \ldots + \alpha^k[a_k]_{t-1}) \times l_{0,i} \ mod \ N) \\ &= \mathsf{Enc}_{\mathsf{pk}}(\sum_{i=1}^{t}(\sum_{m=0}^{k} \alpha^m \cdot [a_m]_{t-1}) \times l_{0,i} \ mod \ N) \end{aligned}$$

Finally, the simulator invokes the algorithm $\mathsf{Dec}_{\mathsf{sk}}(c')$ to decrypt c' and obtains the output $\sum_{m=0}^{k} \alpha^m \cdot a_m = f(\alpha)$. It then sends the output to P_2 and the verification stage is performed.

We now aim to prove that the privacy of the receiver's input is held against the adversary $\mathcal{A}$. P_2 encrypts the values α^m and broadcasts the respective ciphertexts c_m. Furthermore, the security of the homomorphic encryption is determined by the IND-CPA security of the Paillier cryptosystem. Thus, $VIEW_{\mathcal{A}}$ cannot distinguish between the input α and random values in the field, and the receiver's privacy is held

against the adversary in the real model. Therefore, the ideal and the real models are computationally indistinguishable. □

Theorem 11. *A passive adversary $\mathcal{A}$, corrupting a coalition of $t-1$ servers and the receiver P_2, cannot gain any information about the polynomial $f(x)$ before and after executing the computation phase of the protocol Π_{DOPE}.*

Proof. P_1 distributes the coefficients $a_0, a_1, \ldots, a_k$ among the t servers in the setup phase with random polynomials of degree $t-1$ such that each S_i receives $k+1$ shares. Without loss of generality, assume that the adversary $\mathcal{A}$ corrupts a coalition of $t-1$ servers $S_1, \ldots, S_{t-1}$ and the receiver P_2. Each S_i computes the encrypted share c_i' in the computation phase as:

$$c_i' \leftarrow \mathsf{Enc}_{\mathsf{pk}}((\sum_{m=0}^{k} \alpha^m \cdot [a_m]_{t-1}) \times l_{0,i} \bmod N)$$

Since $VIEW_{\mathcal{A}}$ holds the private encryption key, it can decrypt c_i' to obtain the plaintext. Without loss of generality, let f_i be the respective plaintext which can be shown as:

$$f_i \leftarrow (\sum_{m=0}^{k} \alpha^m \cdot [a_m]_{t-1}) \times l_{0,i} \tag{5.1}$$

Where, based on information-theoretic security of the secret sharing method and since f_k (the decrypted share of S_k) has more than one unknown share, the polynomial $f(x)$ is uniformly random over the field and, thus, $VIEW_{\mathcal{A}}$ can learn no information relating to it .

After implementing the computation phase, $VIEW_{\mathcal{A}}$ calculates the output $f(\alpha)$ which can be written as:

$$\begin{aligned} f(\alpha) &= f_1 + \ldots + f_t \\ &= \sum_{i=1}^{t} f_i \end{aligned}$$

$VIEW_{\mathcal{A}}$ has the plaintexts $f_1, f_2, \ldots, f_{t-1}$ and the output $f(\alpha)$. It also learns the number of $t-1$ shares of each coefficient a_m. Hence, it can calculate f_k by the formula $f_k = f(\alpha) - \sum_{i=1}^{t-1} f_i$. However, because the equation of f_k (equation 5.1) has more than one unknown share (i.e., at least for $m = 0, 1$) and due to the linear feature of the secret sharing, $VIEW_{\mathcal{A}}$ can gain no information relating to the shares $[a_m]_{t-1}$ of S_k. Therefore, $VIEW_{\mathcal{A}}$ can learn nothing about the coefficients $a_0, a_1, \ldots, a_k$ of the polynomial, and the sender's privacy is preserved.

□

5.5 Extension to DOT

Distributed oblivious transfer has been a subject of numerous studies, see e.g., [82, 16, 15, 29]. Our protocol can also be extended to a system of secure $\binom{1}{2}$ distributed oblivious transfer, denoted by DOT_1^2. Basically, this system has a similar functionality to DOPE where P_1 holds two inputs, m_0 and m_1, and P_2 has the choice bit $\sigma \in \{0, 1\}$ and they communicate with a set S of t distributed servers. A DOT_1^2 protocol is securely implemented such that P_2 receives m_σ and P_1 gains nothing while the security conditions (i.e., privacy and correctness) are preserved against the adversary [82]. Similar to the DOPE system, our extended DOT_1^2 protocol has two different phases and it holds the security setting. This improves the security of the protocol of [82] where the privacy of the parties' inputs is not preserved against maximum coalition of $t - 1$ servers. The homomorphic encryption is employed to perform the private outsourced computation by the cloud servers. Figure 5.3 shows the extension to the protocol $\Pi_{\text{DOT}_1^2}$.

Input: P_1 has the values m_0 and m_1 in $\mathbb{Z}_q$. P_2 holds σ where $\sigma \in \{0,1\}$.

Output: P_2 learns m_σ.

Verifiable setup phase:

- Each server $S_i \in S$ (for $i = 1, \ldots, t$) generates a random MAC key $\alpha_i \in \mathbb{Z}_q$ and sends it to P_1.
- P_1 generates $(t-1)$-sharings $[m_0]_{t-1}$, $[m_1]_{t-1}$ over the integers integers. He also calculates $\gamma_0 = \alpha_{mac} \cdot m_0$ and $\gamma_1 = \alpha_{mac} \cdot m_1$ where $\alpha_{mac} = \sum_{i=1}^{t} \alpha_i$, and generates the $(t-1)$-sharings $[\gamma_0]_{t-1}$ and $[\gamma_1]_{t-1}$ over the integers. He sends the tuples $\langle m_0 \rangle$ and $\langle m_1 \rangle$ to each server S_i where:

$$\{\forall S_i \in S,\ \tau = 0,1 : \quad \langle m_{\tau i} \rangle \leftarrow ([m_\tau]_{t-1}, \gamma_i([m_\tau]_{t-1}), [\gamma_\tau]_{t-1})\}$$

- Each S_i accepts the share $[m_\tau]_{t-1}$ iff $\gamma_i([m_\tau]_{t-1}) = \alpha_i \cdot [m_\tau]_{t-1}$, otherwise broadcasts *fail* and aborts the protocol.
- P_1 leaves the protocol and takes no step further.

---

computation phase:

- P_2 generates two $k/2$-bit large prime numbers p_c and q_c to calculate a pair of the Paillier encryption keys (pk, sk) where $N = p_c \cdot q_c$. He computes the ciphertext $c_0 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(1-\sigma)$ and $c_1 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\sigma)$ and broadcasts them.

- Each server $S_i \in S$ computes in parallel using homomorphism:

$$c'_i = (c_0^{[m_0]_{t-1}} \cdot c_1^{[m_1]_{t-1}})^{l_{0,i}} \ mod \ N^2$$

where $l_{0,i}$ is the Larange coefficient of S_i.

- Each S_i employs another round of homomorphic encryption to compute the encrypted share of the corresponding MAC value as:

$$c'_i(\gamma_{m_\sigma}) = (c_0^{[\gamma_0]_{t-1}} \cdot c_1^{[\gamma_1]_{t-1}})^{l_{0,i}} \ mod \ N^2$$

and it privately sends the tuple $\langle C'_i \rangle$ to P_2 where $\langle C'_i \rangle \leftarrow (c'_i, c'_i(\gamma_{m_\sigma}), \alpha_i)$.

- P_2 obtains the output as follows:

$$c' = \prod_{i=1}^{t} c'_i \ mod \ N^2$$

$$m_\sigma \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c')$$

Verification

- P_2 calculates the global MAC key $\alpha_{mac} = \sum_{i=1}^{t} \alpha_i$.

- P_2 computes the MAC value of the output as:

$$c'(\gamma_{m_\sigma}) = \prod_{i=1}^{t} c'_i(\gamma_{m_\sigma}) \ mod \ N^2$$

$$\gamma(m_\sigma) \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c'(\gamma_{m_\sigma}))$$

- P_2 checks

$$\gamma(m_\sigma) - \alpha_{mac} \cdot m_\sigma = 0$$

and he accepts m_σ iff it is *Ok*. Otherwise, he broadcasts *fail* and outputs $\perp$.

Figure 5.3 : The extension of the protocol Π_{DOPE} to the protocol $\Pi_{\mathrm{DOT}_1^2}$

Theorem 12. *The protocol* $\Pi_{\mathrm{DOT}_1^2}$ *is secure against a static active adversary corrupting* P_1 *in the setup phase and a coalition of* $t-1$ *dishonest servers in the computation phase with the small probability of error. The protocol is also secure against a passive adversary controlling a coalition of* $P_2(P_1)$ *and* $t-1$ *servers before and after the implementation of the computation phase.*

Proof. It is straightforward to verify the correctness of the protocol $\Pi_{\mathrm{DOT}_1^2}$. Let C and H represent the corrupted and honest parties/servers in the ideal model, respectively. If $P_1 \in C$ in the setup phase, the simulator sends the random shares $[m'_\tau]_{t-1}$ to the functionality which simulates the the introduced errors δ_τ by the adversary $\mathcal{A}$ in the real model, i.e. $[m'_\tau]_{t-1} = [m_\tau]_{t-1} + \delta_\tau$. The simulator executes the functionality and any inconsistency is detected with the error probability of $1/q$. If $S_i \in C$ in the computation phase, the simulator sends a list of corrupt parties to the functionality. The simulator picks random values c''_i, $c''_i(\gamma_{m_\sigma})$ and α'_{mac} for their inputs. This is analogous to the condition where $\mathcal{A}$ deviates from the protocol by adding the errors δ_i, $\delta_{\gamma_{m_\sigma}}$ and δ_α in the real model which can be denoted as $c''_i = c'_i + \delta_i$, $c''_i(\gamma_{m_\sigma}) = c'_i(\gamma_{m_\sigma}) + \delta_{\gamma_{m_\sigma}}$ and $\alpha'_{mac} = \alpha_{mac} + \delta_\alpha$. The simulator runs the functionality and sends the output $m'_\sigma = m_\sigma + \delta_\sigma$ to P_2 who is able to detect the any inconsistency between the output and the corresponding MAC value with the probability $1 - 1/q$ and aborts the protocol. If $P_1, S_i \in H$, the simulator runs the protocol and P_2 receives the new encrypted value $c' \leftarrow \mathsf{Enc}_{\mathsf{pk}}((1-\sigma).m_0 + \sigma.m_1 \ mod \ N)$. Clearly, if $\sigma = 0$ or $\sigma = 1$, P_2 decrypts and receives either m_0 or m_1, respectively.

The privacy condition follows the security evaluation of the main protocol Π_{DOPE}. The protocol preserves the strong security in both the sender's and the receiver's sides and the communication overhead is linear $O(t)$. This gives an improvement

on the protocol of [82] where the communication overhead is $O(td_y)$ in which d_y is the variable degree of a bivariate polynomial with the free term m_σ. □

5.6 Conclusion

DOPE has been the building block of many cryptographic models. In this Chapter, we propose the first verifiable and private DOPE scheme using additive homomorphic encryption feature and secret sharing in the presence of t distributed cloud servers where t does not depend on the degree of sender's polynomial k [61]. Our protocol has two setup and computation phases, and the communication complexity is determined by the factor kt which gives an improvement on the previous DOPE protocols of [75] and [26]. The sender party is just involved in the setup phase which means that the receiver must not necessarily be available in the setup phase that is, also, an improvement on the protocol of [26]. Furthermore, the outsourced computations of the homomorphic encryption can be implemented by the servers in parallel which reduces the computation time.

The security is preserved against a corrupt sender in the setup phase and a static active adversary corrupting a coalition of at most $t-1$ dishonest servers in the computation phase using MAC technique. To the best of the authors' knowledge, this is the first verifiable secure DOPE protocol which authenticates the correctness of the receiver's output with the small probability of error. Moreover, the privacy is held in environment of the strong security where the privacy of the sender's/receiver's input(s) is maintained against a coalition of $t-1$ servers and the opposed party corrupted by a passive adversary. The security of the sender's inputs is information-theoretic before and after running the protocol, while the receiver's input holds IND-CPA security of the encryption system.

Our DOPE scheme can be extended to a secure DOT_1^2 protocol with the same security setting and linear communication complexity $O(t)$ which is an improvement

on the protocol of [82] where the communication overhead is not linear.

Chapter 6

Outsourcing Verifiable Distributed Oblivious Polynomial Evaluation from Threshold Cryptography

6.1 Introduction

In the last Chapter, we presented a verifiable and private DOPE scheme by employing the Paillier cryptosystem for the receiver party. Namely, the protocol requires that P_1 Shamir-shares the coefficients of the polynomial $f(x)$ among the servers and P_2 encrypts his input exponentiations α^j, for $j = 1, \ldots, k$. Each server uses homomorphic encryption to compute an encrypted share of the output and sends it to P_2 who employs another round of homomorphism, decrypts and verifies the final output. However, the system cannot be practical in reality for a normal receiver party P_2 with low-computation power devices. That is because the encryption and the decryption procedures of a public encryption system (in particular the Paillier encryption, since N is a number with at least 2048 bits and, also, the homomorphic encryption computation works with N^2) is very expensive and requires a lot of heavy computation power and time which makes that scheme less favourable for a normal receiver user P_2.

An important question arises here is that is it possible to delegate the expensive processes of encryption and decryption to P_1 and the cloud servers while the inputs security conditions are still maintained? Also is that possible to reduce the communication complexity while the heavy computations are outsourced? These improvements make it possible that any normal user, with a low computational

power device, plays the role of the receiver party P_2. Needless to say that, the proliferation of mobile devices, such as smart phones and laptops, has provided a new field in which a user with a computationally weak device would like to participate and be able to securely outsource a DOPE computation.

6.1.1 Applications

As we already mentioned, OPE (as the parent system of DOPE) has been a notable building block of various cryptographic models and security fields such as metering the number of visitors to a website [81], oblivious neural networking [23], symmetric cryptography [84], oblivious keyword search [47], data mining [1], RSA keys generation [55], set intersection [48] and electronic voting [87]. In secure information-comparison protocols, two parties with their private inputs, say x and y, wish to know whether $x > y$ without leaking any additional information on the inputs which can be used in password comparison, online auction and benchmarking [41]. Another important application is in privacy-preserving machine learning which can be used in healthcare [50], linear regression [28] and two-party inner product [42]. These algorithms usually have two phases: training and classification where OPE plays a secure tool to obtain the output in the classification phase in the secure fashion. As an example, a healthcare company provider trains a model in the training phase and a patient wishes to gain a prediction of his health status using their model without revealing any information about his personal health records [50].
In most of these applications, it is assumed that a normal user with low-computation power is capable of achieving the output. Thus, a DOPE scheme, as a more decentralized and secure alternative tool, must acclimatize to this requirement in order to be considered in these vast application fields.

6.1.2 Our Contribution

We present a lightweight DOPE scheme where a sender party P_1, holding a private polynomial $f(x)$ of degree k, and a receiver party P_2, with an input α, wish to conduct a secure computation with the help of t cloud servers such that P_2 obtains the output $f(\alpha)$ over a large field. The number of cloud servers (t) is independent of the polynomial degree (k). Unlike the protocol of [27], the main two parties in our system can be anonymous to each other where they interact with the cloud servers and the servers check that whether the parties commit to their inputs using non-interactive commitment techniques. We employ the idea of threshold decryption (described in section 4.2.2) such that the cloud servers perform the secure computation of modular exponentiation operations which is the main computational bottleneck of most public key cryptosystems (particularly in our case the Paillier encryption system) [69]. As a result, P_2 with a low-computational device is easily able to calculate and verify the final output using simple arithmetic operations which gives an asymptotic improvement on the study of the Chapter 5 [61]. Our scheme consists of two phases:

- **Setup Phase**: P_1 encrypts his inputs and reveals them, and the servers check the commitments. P_1 also distributes the masked private key among the cloud servers and leaves the protocol.

- **Computation Phase**: P_2 picks a set of random values over the field and adds his input to these elements, and reveals them. The servers check the P_2's commitment to these values. The homomorphic encryption and the heavy decryption computation of the set are outsourced by the cloud servers. P_2 utilizes one round of oblivious transfer to obtain the correct index and calculates the output. He repeats the same process to verify the output using message authentication codes.

Our scheme maintains the security (the inputs privacy and the output correctness) against a static active adversary corrupting a coalition of up to $t-1$ cloud servers and the opposed party, with IND-CPA security of Paillier cryptosystem for P_1 and statistical security for P_2. Unlike most of the works in this field which have considered just semi-honest adversaries, we present a fully secure DOPE protocol with low probability of error which can be employed for general distributed privacy preserving systems. The communication overhead is linear $O(t)$ improving on the previous DOPE protocols. This gives an important result that the communication complexity does not depend on the polynomial degree k.

6.2 Security

We previously discussed the security conditions of a DOPE system in the section 5.2.3. We now aim to give a more formal ideal/real security model of a DOPE system and we later evaluate our scheme based on this model. We assume there exists a simulator $\mathcal{S}$ playing the role of an adversary in the ideal model. $\mathcal{S}$ takes the inputs of the corrupt parties and executes the functionality $\mathcal{F}$ such that the participants do not interact directly with each other. This model achieves the highest level of security and is denoted by $IDEAL_{\mathcal{F},\mathcal{S}}$ with the view indicated by $VIEW_{\mathcal{S}}$. On the contrary, the participants implement the protocol Π in the presence of a probabilistic polynomial-time adversary $\mathcal{A}$ who corrupts the parties in the real model. The model is denoted by $REAL_{\Pi,\mathcal{A}}$ with the view of the adversary $VIEW_{\mathcal{A}}$. The protocol Π is said to be secure if these two models $IDEAL_{\mathcal{F},\mathcal{S}}$ and $REAL_{\Pi,\mathcal{A}}$ are computationally indistinguishable [21].

Most of the DOPE studies have attempted to only meet the strong privacy condition in their protocols. However, we also add the correctness condition to make our scheme more practical and secure. As a result, a fully secure DOPE scheme must satisfy the privacy and the correctness requirements as follows:

- **Receiver's Privacy**: The adversary $\mathcal{A}$ corrupts a coalition of P_1 and up to $t-1$ cloud servers while the receiver party P_2 gets involved in the protocol with the input $\alpha \in \mathbb{F}$. The protocol is private for the P_2's input, if for any $\alpha' \in \mathbb{F}$, the $VIEW_{\mathcal{A}}$ for α and that for α' are computationally indistinguishable.

- **Sender's Privacy**: Here, $\mathcal{A}$ controls a coalition of P_2 and at most $t-1$ servers, while the sender party P_1 has the polynomial $f(x)$ of degree k. The simulator $\mathcal{S}$ with any input α' in the field implements the same functionality in the ideal model and obtains the output $f(\alpha')$. The privacy of the P_1's polynomial is preserved, if $f(\alpha')$ is computationally indistinguishable from any random values over the field. This requirement implies that $VIEW_{\mathcal{A}}$ gains no information about the polynomial $f(x)$ except the value $f(\alpha)$. Note that P_2 is only allowed to evaluate at most $k-1$ values from the same sender P_1, otherwise he can compute the polynomial $f(x)$ and break the sender's privacy.

- **Correctness**: The adversary $\mathcal{A}$ holds the full control of a coalition of P_1 and up to $t-1$ cloud server. P_2 with the input α executes the protocol to obtain the output $f(\alpha)$ while $\mathcal{A}$ deviates from the protocol in an arbitrary fashion to change the output to $f(\alpha')$ for any $\alpha' \in \mathbb{F}$ without being detected. The correctness of the output is maintained if $f(\alpha)$ and $f(\alpha')$ are computationally indistinguishable with low probability of error.

6.3 Our DOPE Scheme

We discuss our protocol in this section which includes two phases: the setup and the actual computation. P_1 communicates with the cloud servers in the setup phase and then leaves the protocol, while P_2 interacts with the servers in the actual computation phase to evaluate the output.

6.3.1 Setup Phase

This phase can be executed at anytime well in advance of the computation phase. P_1 encrypts the coefficients of his polynomial $f(x)$ and commits to them showing that he knows the plaintexts. We propose a non-interactive zero-knowledge proof technique, which has some similarity to the scheme for the Paillier cryptosystem given in [4], such that the servers check the commitments and the protocol fails in case of detecting any inconsistency. Figure 6.1 shows the setup phase of the protocol Π_{DOPE}.

Input: P_1 holds the polynomial $f(x) = \sum_{j=0}^{k} a_j x^j$ where $a_j \in Z_q$.

- P_1 invokes the keys generation algorithm $\mathsf{Gen}(1^k)$ to produce a keys pair (pk, λ) of the Paillier cryptosystem. He encrypts the coefficients a_j to obtain the ciphertexts $c_j \leftarrow \mathsf{Enc}_{\mathsf{pk}}(a_j)$ as:

$$c_j = g^{a_j} \cdot r^N \bmod N^2$$

  and he reveals them.

- The servers get involved with P_1 to check that whether he has committed to the correct coefficients a_j using non-interactive zero-knowledge proofs. Namely, for each c_j:

  * P_1 picks random values $y \in Z_N$ and $s \in Z_N^*$ and computes $u_j = g^y \cdot s^N \bmod N^2$. He reveals the values u_j.

  * Each server $S_i \in S$ chooses a random value $e_i \in Z_N$ and sends it to P_1.

  * P_1 calculates $e = \sum_{i=1}^{t} e_i$. Then he computes $\tau = y - e \cdot a_j \bmod N$ and $\nu_j = s \cdot r^{-e} \bmod N^2$, and broadcasts them.

* S_i computes $c_j^{e_i} \mod N^2$ and reveals it. The value $c_j^e = \prod_{i=1}^{t} c_j^{e_i} \mod N^2$ is computed by the means of homomorphic encryption feature.

* The servers check that if $g^\tau \cdot c_j^e \cdot \nu_j^N = u_j \mod N^2$ and the protocol is aborted if this check fails.

Each server S_i picks a random MAC key α_i and sends it to P_1.

- P_1 calculates the global MAC key α_{mac} as the additive secret of the random keys $\alpha_{mac} = \sum_{i=1}^{t} \alpha_i$. He computes the MAC value of a_j as $\gamma(a_j) = \alpha_{mac} \cdot a_j$ and encrypts it to obtain the ciphertext $c(\gamma_{a_j}) \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\gamma(a_j))$.

- Similarly, the servers check that whether P_1 has committed to the correct encrypted values of $\gamma(a_j)$ using another round of the above non-interactive zero-knowledge proof technique.

- P_2 masks the private key by a random value $\beta \in Z_N^*$ as $\theta = \beta \cdot \phi(N) \bmod N$ and distributes the $(t-1)$-sharings $[\theta]_{t-1}$ in the field among the servers. He also distributes the $(t-1)$-sharings $[\beta \cdot \lambda]_{t-1}$ over the field $\mathbb{Z}_{\phi(N^2)}$.

Figure 6.1 : Setup phase of the protocol Π_{DOPE}

For the correctness of the proposed zero-knowledge proof technique, note that since $e = \sum_{i=1}^{t} e_i$, The homomorphic property of the encryption system determines that $c_j^e = \prod_{i=1}^{t} c_j^{e_i} \mod N^2$. Therefore, the final checking equation can be written as:

$$g^{y-e\cdot a_j} \cdot (g^{e\cdot a_j} \cdot r^{e\cdot N}) \cdot (s^N \cdot r^{-e\cdot N}) \mod N^2 =$$
$$g^y \cdot S^N \mod N^2 = u_j$$

6.3.2 Computation Phase

P_2 generates and publishes a set of m random values such that his input α is an element in it. The serves are involved with P_2 to make sure that he has committed to the correct inputs using a non-interactive zero-knowledge proof technique. The servers employ the idea of threshold decryption to perform the expensive part of the decryption procedure. Namely, the heavy computations of the homomorphic encryption and the modular exponentiations are outsourced to the cloud servers. Finally, P_2 conducts one round of 1-out-of-m oblivious transfer to obtain the correct outcome. Figure 6.2 illustrates the computation of the protocol Π_{DOPE}.
Note that, as we already mentioned in Chapter 5, it is required that N holds the condition $N > (k+1)q^2$.

Input: P_2 has the value $\alpha \in Z_q$.

Output: P_2 obtains $f(\alpha)$ modulo q.

- P_2 picks $m-1$ random elements $\{r_1, r_2, \ldots, r_{m-1}\}$ over the field $\mathbb{Z}_q$. He adds the input α to these elements with a random index n where $1 \leq n \leq m$. This makes a random tuple $\{r_1, \ldots, r_n, \ldots, r_m\}$ such that $r_n \leftarrow \alpha$. He publishes the tuple $\{r_e\}$ for $e = 1, \ldots, m$.

- The servers check that whether P_2 has committed to the correct values he sent (i.e., $r_1, \ldots, r_m$) using the MAC commitment scheme. Namely:

* Each server S_i sends the random MAC key α_i, generated in the setup phase, to P_2.

* P_2 calculates the MAC value of each r_e, as $\gamma_i(r_e)$, using the random MAC key α_i and sends it to S_i.

* S_i checks for each r_e that whether:

$$\gamma_i(r_e) = \alpha_i \cdot r_e$$

Each server $S_i \in S$ computes m new encrypted shares as:

$$\begin{aligned} c_{ie} &= c_0 \cdot c_1^{r_e} \cdot c_2^{r_e^2} \cdot \ldots \cdot c_k^{r_e^k} \bmod N^2 \\ &= c_0 \cdot \prod_{j=1}^{k} c_j^{r_e^j} \bmod N^2 \end{aligned}$$

where one of these m ciphertexts is the correct encrypted output c_{in}, i.e., $c_{in} \in \{c_{i1}, c_{i2}, \ldots, c_{im}\}$.

- S_i sends the $(t-1)$-sharings $[\theta]_{t-1}$ to P_2.

- The servers compute the modular exponentiations of the threshold decryption procedure for each c_{ie}. Namely, each S_i computes $c_{ie}^{[\beta\cdot\lambda]_{t-1}\cdot l_{0,i}} \bmod N^2$ where $l_{0,i}$ is its Lagrange coefficient. Each server pools its decrypted share to compute:

$$c_e^{\beta\cdot\lambda} = \prod_{i=1}^{t} c_{ie}^{[\beta\cdot\lambda]_{t-1}\cdot l_{0,i}} \bmod N^2$$

- P_2 conducts one round of 1-out-of-m oblivious transfer with a server to obtain $c_n^{\beta\cdot\lambda}$. He opens θ and can simply calculate $f(\alpha)$ as follows:

$$f(\alpha) = L(c_n^{\beta\cdot\lambda})/\theta \bmod N$$

Figure 6.2 : Computation of the protocol Π_{DOPE}

Verification

P_2 and the servers repeat the same computation steps with the encrypted MAC values to authenticate the output. Figure 6.3 shows the verification of the protocol Π_{DOPE}.

Verification

- Similar to the computation process on figure 6.2, S_i computes m encrypted MAC values using the tuple $\{r_e\}$ (for $e = 1, \ldots, m$) as:

$$c_i(\gamma_e) = c(\gamma_0) \cdot \prod_{j=1}^{k} [c(\gamma_{a_j})]^{r_e^j} \bmod N^2$$

  where one of these m ciphertexts is the encrypted MAC value of the output $c_i(\gamma_n)$.

- The servers perform the modular exponentiations computation of threshold decryption for each $c(\gamma_e)$. Namely, each S_i pools $[c_i(\gamma_e)]^{[\beta \cdot \lambda]_{t-1} \cdot l_{0,i}}$ to compute:

$$[c(\gamma_e)]^{\beta \cdot \lambda} = \prod_{i=1}^{t} [c_i(\gamma_e)]^{[\beta \cdot \lambda]_{t-1} \cdot l_{0,i}} \bmod N^2$$

- P_2 executes a 1-out-of-m oblivious transfer with a server to gain $[c(\gamma_n)]^{\beta \cdot \lambda}$. He simply calculates the MAC value of the output as:

$$\gamma(f(\alpha)) = L([c(\gamma_n)]^{\beta \cdot \lambda})/\theta \bmod N$$

- P_2 obtains the global MAC key $\alpha_{mac} = \sum_{i=1}^{t} \alpha_i$ and checks whether:

$$\alpha_{mac} \cdot f(\alpha) - \gamma(f(\alpha)) = 0$$

he accepts the output $f(\alpha)$ if it is *OK*, otherwise the protocol *fails* and outputs $\perp$.

Figure 6.3 : Verification of the protocol Π_{DOPE}

Note that the operations of computation and verification can be implemented in parallel. The communication complexity of our scheme is bounded to be linear $O(t)$ field elements which implies that it does not depend on the polynomial degree k. This improves on the communication overheads of the previous DOPE studies [61, 75, 27] which are $O(kt)$.

6.3.3 Security Evaluation

We evaluate our scheme based on the simulation security model described in section 6.2.

Theorem 13. *The protocol* Π_{DOPE} *is secure against a static active adversary corrupting a coalition of at most* $t-1$ *servers and* P_1/P_2 *with small probability of error. The security is semantic for the* P_1*'s polynomial and statistical for the* P_2*'s input.*

Proof. Let H and C represent the honest and corrupt parties/servers in the ideal model, respectively. Let $\{(S_1, \ldots, S_{t-1}), P_1\} \in \mathsf{C}$ and $S_t \in \mathsf{H}$ in the setup phase. The simulator first sends the wrong inputs $c_{j\delta}$ (for $j = 0, 1, \ldots, k$) to the functionality which simulate the errors $c_{j\delta} = c_j + \delta_j$ in the real model. The simulator runs the functionality and S_t detects any inconsistency in the commitments of the P_1's inputs using the proposed zero-knowledge proof technique. Then, the simulator sends the

random values $c_{ie\delta}$, $\alpha_{i\delta}$ and the shares $[\beta \cdot \lambda]^{\delta}_{t-1}$ and $[\theta]^{\delta}_{t-1}$ in the computation phase. This is analogous to the condition where $\mathcal{A}$ introduces the errors $c_{ie\delta} = c_{ie} + \delta_c$, $\alpha_{i\delta} = \alpha_i + \delta_\alpha$, $[\beta \cdot \lambda]^{\delta}_{t-1} = [\beta \cdot \lambda]_{t-1} + \delta_{\beta\lambda}$ and $[\theta]^{\delta}_{t-1} = [\theta]_{t-1} + \delta_\theta$ to the real model. The functionality is executed and P_2 can detect any inconsistency in the output using the global MAC key α_{mac} with the probability $1 - 1/q$.

Let $\{(S_1, \ldots, S_{t-1}), P_2\} \in \mathsf{C}$ and $S_t \in \mathsf{H}$ in the computation phase. The simulator runs the corresponding MAC commitment in this phase and S_t checks and detects any inconsistency in the commitment of the P_2's input.

Note that the privacy of the P_1's polynomial is preserved by the IND-CPA security of the Paillier cryptosystem, and P_2 holds the privacy of his input α using the 1-out-of-m oblivious transfer with the security parameter m. Some of the efficient oblivious transfer protocols can be found in [83].

If all the parties and the servers are in H, the functionality is executed with no fault detection. P_2 picks a random set R as:

$$R = \{r_1, r_2, \ldots, r_m\}$$

such that $r_n = \alpha$ where $n \in \{1, 2, \ldots, m\}$. Each server S_i computes a set of m ciphertexts $c_{ie} = \{c_{i1}, \ldots, c_{in}, \ldots, c_{im}\}$ $(e = 1, \ldots, m)$ where c_{in} can be written as:

$$c_{in} = \mathsf{Enc}_{\mathsf{pk}}[a_0 + \sum_{j=1}^{k} (a_j \cdot \alpha^j)]$$

Finally, the servers perform the heavy computations part of the Paillier's threshold decryption on the set of ciphertexts $\{c_e\}$. P_2 gains $c_n^{\beta \cdot \lambda}$ using one round of $\binom{1}{m}$ oblivious transfer and executes the final simple arithmetic decryption step to obtain the output $f(\alpha)$.

□

6.4 Conclusion

DOPE is a variant of OPE which has many applications in various areas from cryptographic models to privacy-preserving algorithms. We propose a lightweight DOPE scheme where the expensive computations of homomorphic encryption and modular exponentiations are outsourced to a number of t cloud servers which does not depend on the polynomial degree k [63]. This can be achieved by having the servers conduct the idea of threshold decryption such that the output still remains confidential to at most $t-1$ servers. Therefore, any normal user with low computational-power devices (e.g., mobile, laptop) would be able to gain and verify the output which makes this scheme more practical than the previous protocols and gives asymptotic improvement on the method of the Chapter 5 [61].

Our scheme includes two separate phases: the setup and the actual computation. The sender P_1 is involved with the servers in the setup phase while the receiver P_2 interacts with the servers in the computation phase. This implies that the main two parties can remain anonymous to each other. Our protocol holds the full security against a static active adversary corrupting a coalition of at most $t-1$ servers and the opposed party by the IND-CPA security of Paillier cryptosystem, and the statistical security of oblivious transfer with the security parameter m while P_2 verifies the output using the unconditional secure method of global message authentication code. Also, the servers check the commitments of the parties' inputs using two separate non-interactive zero-knowledge proof techniques. The communication complexity is bounded to $O(t)$ field elements giving an improvement on the previous studies [75, 27, 61] which had the communication overhead $O(kt)$. This implies that, unlike the previous studies, the communication complexity can be so efficient such that it does not depend on the polynomial degree k.

Chapter 7

Fair Distributed Oblivious Polynomial Evaluation via Bitcoin Deposits: Compute-as-a-Service

7.1 Introduction

In cloud computing service (the cloud computing providers e.g., Amazon Web Services, Microsoft Azure, Google Cloud Platform and ...), an honest cloud server is paid some amount of reward for the service it performs. Also, a corrupt server compensates some amount for conducting the wrong and malicious service. Hence, in order to make the DOPE system more practical in the cloud computing service, where the parties pay for the outsourced computation service to the servers, the notion of *fairness* is required such that any corrupt party/server must compensate as well. In other words, achieving the fairness property is a must in secure distributed systems involved with the cloud computing service, and it is where all the DOPE protocols are lack of. Thus, we aim to present a *fair* and *robust* DOPE scheme using Bitcoin transactions in this Chapter. In addition to the fairness, the robustness property ensures that the receiver party P_2 gains the correct output, despite the presence of a threshold number of corrupt servers controlled by an active adversary in the system. These features would enable the DOPE system to be adapted to the client/server world of cloud computing service.

7.1.1 Our Contribution

We present the first *fair* secure DOPE scheme where the sender party P_1, having a polynomial $f(x)$ of degree k, and the receiver party P_2, holding the input α,

conduct a secure outsourced computation service with a set of n cloud servers such that P_2 obtains the output $f(\alpha)$ [62]. The number of cloud servers is $n \geq 2t+1$ where t can be considered the security parameter and it is independent of the polynomial degree k. The fairness property ensures that an honest server/party never has to pay any penalty and also, if a server/party does not deliver the correct output to P_2 or aborts the protocol, it compensates to an honest party [62]. This can be achieved by the features of *scripts* and *time-lock* in Bitcoin transactions such that an honest server gains the reward for the computation service it performs, whereas each corrupt party/server gets penalized as well. Note that we choose Bitcoin network, as a decentralised ledger without the need of a third trusted party, since the market price is less volatile compared to the other cryptocurrencies with smart contracts. With regards to the outsourced computation service, each server computes an encrypted share of the output using homomorphic feature of Paillier cryptosystem. In detail, our scheme consists of two phases:

- **Setup Phase**: P_1 distributes the shares of his polynomial among the servers. He also commits to the shares using the Pedersen's non-interactive verifiable secret sharing scheme [89]. The servers check the commitments and if the honest majority complain, P_1 is dishonest and compensates to the servers. Otherwise, P_1 penalizes every corrupt server and eliminates it from the protocol.

- **Computation Phase**: P_2 encrypts his inputs and broadcasts them. Each server employs one round of homomorphic encryption to compute the encrypted share of the output and sends it back to P_2 who checks the commitment. If P_2 detects any faulty server, he eliminates it and gets compensated from it. Otherwise, he pays to the honest servers for the computation service they have executed.

We assume that the communication channels are asynchronous, and if a server/party does not perform the computation and communicate by the time-lock t_l, it is tagged as corrupt and makes the compensation for that. Moreover, the servers/parties send the Bitcoin deposits before commencing each phase. Note that an honest server gets back its deposit after completing the service in each phase. Our scheme holds the full security against a static active adversary corrupting a coalition of t cloud servers and P_1 in the presence of the major honest servers (i.e., $t+1$ servers). The privacy of the P_1's inputs is preserved by the unconditional security of secret sharing while the privacy of the P_2's inputs is maintained with IND-CPA security of Paillier cryptosystem. The communication complexity is bounded to $O(kt)$ while the fairness property is also achieved.

7.2 Preliminaries

7.2.1 Pedersen's Verifiable Secret Sharing

We employ the non-interactive secret sharing commitment approach of Pedersen [90] over the integers to detect any corrupt party in our protocol. Note that the Pedersen's verifiable secret sharing is *unconditionally hiding* and *computationally binding* under the assumption of discrete logarithm with the information rate $\frac{1}{2}$. Namely, a dealer chooses two large prime numbers p and q such that q divides $p-1$, i.e., the order q is a subgroup of the field $\mathbb{Z}_p$. The dealer picks two random generators g and h over the field $\mathbb{F}_q$ such that $\log_g h$ is unknown. The dealer, holding a secret s in $\mathbb{Z}_q$, shamir-shares the secret among the participants using a random polynomial $p(x) = s + b_1 x + b_2 x^2 + \ldots + b_t x^t \bmod q$ where $b_j \in \mathbb{Z}_q$ (for $j = 1, \ldots, t$). He also chooses a random companion polynomial $p'(x) = s' + b'_1 x + b'_2 x^2 + \ldots + b'_t x^t \bmod q$ where $s', b'_j \in \mathbb{Z}_q$ and distributes the shares among the participants. Thus, each party P_i is given two shares $[s]_t$ and $[s']_t$. The dealer computes:

$$A_0 = g^s \cdot h^{s'} \mod p$$

and publishes it. He also computes $A_j = g^{a_j} \cdot h^{a'_j} \mod p$ and broadcasts them. Each share-holder P_i checks that whether the dealer has committed to the correct share $[s]_t$ as follows:

$$g^{[s]_t} \cdot h^{[s']_t} = \prod_{j=0}^{t} (A_j)^{i^j} \mod p$$

P_i accepts the share $[s]_t$ if the check is OK, otherwise he broadcasts a complaint.

7.2.2 Bitcoin Transactions

Bitcoin is a decentralized peer-to-peer electronic cash system which was designed and developed by an anonymous person or group of people [80] as the first innovative idea of cryptocurrency. The transactions are stored in blockchain (as a public ledger) which helps to achieve agreement in decentralized scenarios without a trusted third party and also can avoid the single point of failure attack. Due to this property, Bitcoin has attracted some studies of multi-party computation to add fairness to their protocols, see e.g., [3, 2, 74, 73]. The data consistency in blockchain is maintained using a consensus algorithm called proof of work. Namely, the first node solving a difficult computation puzzle (which generally takes roughly 10 mins) is selected to record a block of transactions. Moreover, the security of the Bitcoin network is preserved by the honest majority of computing power. Note that one may choose any other cryptocurrency with smart contract which can be applicable to our scheme, nevertheless, we employ Bitcoin since the market price and the transaction fee is less volatile and, thus, it is more reliable.

In detail, the system consists of addresses and transactions between them. An address is the hash of a public key and a transaction works with asymmetric cryptography. Each block can have several transactions in the body section. A sender

signs a transaction with his private key and the recipient verifies the signature by the sender's public key. A transaction can have some inputs, i.e., it can accumulate money from several past transactions. Each transaction T_x includes the index of the previous transaction y, the scripts, the value d ₿ and the time-lock t_l. The scripts of a transaction have a very useful feature where the users have much more flexibility in defining the condition on how the transaction T_x can be redeemed. This is achieved by the *input-scripts* and *output-scripts*. The output-script of the transaction T_x is a description function π_x with a boolean output. The transaction T_x is redeemed successfully and is valid if π_x evaluates to true, and then it is taken to the input-script of the next transaction. In other words, the input-script σ_x is a witness that is used to make the output-script π_y of the last transaction T_y evaluates to true on the current transaction T_x. One may think of an input-script as a signature of the transaction and the output-script as a verification algorithm of the signature. Moreover, if the time-lock t_l of the transaction T_x is reached, the transaction is redeemed automatically. So, the time-lock tells at what time the transaction becomes valid. Figure 7.1 shows the structure of the current transaction T_x with the value d_2 ₿ and the last redeemed transaction T_y.

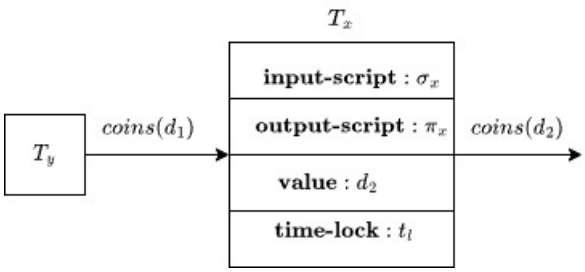

Figure 7.1 : The structure of the transaction T_x in Bitcoin

To summarize, a transaction is valid if 1) the output-script evaluates to true or 2) the time-lock is reached. In our case, we employ the script and time-lock properties such that an honest party has the authority to redeem a transaction deposit. This

can be either making a payment to an honest cloud server for the computation service it has performed or penalizing a corrupt server. Note that an honest server gets back its deposit after conducting the computation service.

7.2.3 Security Model

As we previously discussed the ideal/real security model of a DOPE scheme, the protocol Π is said to be securely implemented if the ideal and the real models, $IDEAL_{\mathcal{F},\mathcal{S}}$ and $REAL_{\Pi,\mathcal{A}}$, are computationally indistinguishable [21]. We now describe the security conditions of a fair secure DOPE model. Namely, in a secure DOPE protocol with n cloud servers, where P_1 holds a polynomial $f(x)$ of degree k and P_2 has a value α, the privacy and the correctness requirements must be satisfied as follows:

- **Receiver's Privacy:** The adversary $\mathcal{A}$ corrupts a coalition of P_1 and a number of maximum t cloud servers. The protocol maintains the privacy of the P_2's input, if for any α' in the field, the $VIEW_{\mathcal{A}}$ for α and that for α' are computationally indistinguishable in the real model.

- **Sender's Privacy:** The adversary $\mathcal{A}$ controls a coalition of P_2 and up to t cloud servers. The simulator $\mathcal{S}$ executes the functionality with a random value α' in the field to gain the output $f(\alpha')$. The privacy of the sender's polynomial is held if $f(\alpha')$ is computationally indistinguishable from any value randomly chosen over the field. In other words, $VIEW_{\mathcal{A}}$ must get no information about the polynomial $f(x)$ except the output $f(\alpha)$.

  It should be stated that P_2 is only allowed to obtain at most $k-1$ outputs from the same sender P_1, otherwise, he would be able to gain the polynomial $f(x)$ and break the sender's privacy.

- **Correctness:** Here, a static active adversary $\mathcal{A}$ takes the full control of a

coalition of P_1 and up to t cloud servers. P_2 implements the protocol to gain the value $f(\alpha)$ while $\mathcal{A}$ deviates from the protocol trying to change the output to $f(\alpha')$ without being detected for any α' over the field. The protocol preserves the correctness if $f(\alpha)$ and $f(\alpha')$ are computationally indistinguishable.

- **Fairness:** In addition to the conditions above, we add the *fairness* property to our scheme. Namely, in a fair DOPE scheme:
  - An honest cloud server gets paid for the computation service it performs.
  - An honest party/server never has to pay any penalty.
  - If a party/server does not deliver the correct output to P_2 or it aborts the protocol before the computation finishes, it compensates for conducting the malicious behaviour.

Note that we assume the receiver party P_2 is always honest and does not deviate from the protocol as he wishes to obtain the correct output. Furthermore, in order to detect a malicious sender P_1 and also achieve the robustness feature, the majority of the cloud servers are honest in our scheme, i.e., $n \geq 2t + 1$ servers with at most t corrupt servers.

7.3 Our Scheme

We discuss our DOPE protocol in this section. Our protocol includes two phases: setup and computation.

We assume that the parties have access to a perfect clock and the communication between the parties takes no time, unless the adversary delays. In particular, the parties and the servers have agreed on a time-lock t_l for the computation delay before the protocol begins.

7.3.1 Setup Phase

The sender party P_1 is involved with n cloud servers ($n \geq 2t+1$) in this phase. We assume that P_1 and the servers have already posted their deposits via the Bitcoin network and their transactions are already on the ledger before this phase starts. Figure 7.2 depicts the setup phase of our protocol Π_{DOPE}.

Input: P_1 holds the polynomial $f(x) = a_0 + a_1 x + \ldots + a_k x^k = \sum_{j=0}^{k} a_j x^j$ where $a_j \in Z_q$.

- P_1 has already made the deposit D_1฿ for each server. Also, Each server $S_i \in S$ has posted the deposit d ฿ to the ledger.

- P_1 distributes t-sharings $[a_0]_t, [a_1]_t, \ldots, [a_k]_t$, denoted by $[a_j]_t$ among the servers. Also, as described in the section 7.2.1, he distributes random companion t-sharings $[a'_j]_t$ over the field $\mathbb{Z}_q$. He picks a large prime p and two generators g and h in $\mathbb{F}_q$ such that $\log_g h$ is unknown, and he commits to the shares $[a_j]_t$ using the Pedersen's VSS scheme. Namely, P_1 computes the commitments $A_{j0}, A_{j1}, \ldots, A_{jt}$ in $\mathbb{F}_p$ (see section 7.2.1) denoted by A_{je} for $e = 0, \ldots, t$. P_1 broadcasts the commitments A_{je}, i.e., totally $(k+1) \times (t+1)$ commitments.

- Each server $S_i \in S$ checks for its share $[a_j]_t$ that:

$$g^{[a_j]_t} \cdot h^{[a'_j]_t} = \prod_{e=0}^{t} (A_{je})^{i^e} \mod p \tag{7.1}$$

  and accepts the share $[a_j]_t$ if the commitment is OK and receives the P_1's deposit D_1฿, otherwise it broadcasts a complaint.

  If the server does not respond about the check by the time-lock t_l, it is tagged as corrupt.

- If more than t servers complain, the sender P_1 is dishonest and the protocol *fails*. Each server gets compensated and redeems the deposit D_1₿ from P_1.
- If equal or less than t servers complain, they are tagged as corrupt. Let C denote the set of corrupt servers ($|\mathsf{C}| \leq t$). P_1 penalises each server in C redeeming its corresponding transaction d ₿ and eliminates it from the protocol.

Figure 7.2 : The setup phase of the protocol Π_{DOPE}

Note that in order to incentivise and prevent the servers from cheating, the reward value for each server's computation service must be greater than its initial deposit, i.e., $D_1 > d$. Of course, an honest server gets back its deposit at the end of this phase.

7.3.2 Computation Phase

The receiver P_2 starts this phase while the corrupt servers in C in the setup phase have been eliminated. So, P_2 communicates with a set of $S \subseteq n - \mathsf{C}$ servers where $|S| \geq t+1$. Similarly, the players post their deposit transactions to the ledger before commencing this phase. Each server computes an encrypted share of the output and P_2 verifies the shares using the Pedersen's commitments published in the setup phase. P_2 detects any corrupt server, gets the compensation from it and eliminates it. Figure 7.3 shows the computation phase of the protocol Π_{DOPE}.

Input: P_2 has the value $\alpha \in \mathbb{Z}_q$.

Output: P_2 obtains $f(\alpha)$ in $\mathbb{Z}_q$.

- P_2 has already posted the deposits D_2 ₿ for each server. Also, each server $S_i \in S$ has sent the deposit transaction d ₿ to the ledger.

- P_2 invokes the keys generation algorithm $\mathsf{Gen}(1^k)$ of the Paillier cryptosystem (described in section 5.2.2) to produce the keys (pk, sk) where the public key is $N = p_c \cdot q_c$. He encrypts the values α^j (for $j = 0, 1, \ldots, k$) to obtain the ciphertexts $c_j \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\alpha^j)$ and publishes them.

- Each S_i employs the homomorphic feature to compute an encrypted share of the output as:

$$c_i = \prod_{j=0}^{k} c_j^{[a_j]_t} \mod N^2$$

similarly, it computes an encrypted share of the companion polynomials, which it receives in the setup phase as:

$$c'_i = \prod_{j=0}^{k} c_j^{[a'_j]_t} \mod N^2$$

and sends them to P_2. If S_i delays and does not send the corresponding c_i and c'_i to P_2 by the time-lock t_l, P_2 tags it as a corrupt server.

- P_2 decrypts c_i and c'_i to open the share $[f(\alpha)]_t$ of the output and the companion share $[v']_t$ of the companion polynomials, respectively. Let $E_j = \prod_{e=0}^{t} A_{je}^{i^e} \mod p$ which can be computed by P_2 for each server S_i. He checks for S_i that whether:

$$g^{[f(\alpha)]_t} \cdot h^{[v']_t} = \prod_{j=0}^{k} E_j^{\alpha^j} \mod p \tag{7.2}$$

if it is *Ok*, he accepts the computation of the share $[f(\alpha]_t$ from the server S_i. Otherwise S_i is corrupt.

- P_2 penalizes every corrupt servers in this phase, redeems its deposit d ₿ and eliminates it. Finally, P_2 obtains $f(\alpha)$ using the t-sharings $[f(\alpha)]_t$.

Figure 7.3 : The computation phase of the protocol Π_{DOPE}

Note that an honest servers gets back its deposit after accepting its computation service by P_2. Similar to the setup phase, due to the incentive mechanism and to prevent the cloud servers from cheating, the reward amount D_2 ₿ has to be greater than each server's deposit d_2 ₿. The communication complexity of our protocol is $O(kt)$ which is the same as that in the previous DOPE protocols [61, 27]. However, our DOPE holds the *fairness* with the same communication overhead.

7.4 Security Evaluation

We assess the security of our scheme based on the security model described in section 7.2.3.

Theorem 14. *The protocol* Π_{DOPE} *is fair and robust against a static active adversary corrupting a coalition of* P_1 *and at most* t *cloud servers. The security is unconditional for the* P_1*'s polynomial and semantic for the* P_2*'s input.*

Proof. Let H and C denote the honest and the corrupt parties/servers in the ideal model, respectively. Let $\{P_1, (S_1, \ldots, S_t)\} \in \mathsf{C}$ and $\{S_{t+1}, \ldots, S_{2t+1}\} \in \mathsf{H}$ in the setup phase. The simulator $\mathcal{S}$ broadcasts wrong commitments $A_{je\delta}$ to the functionality which is analogous to the situation where the adversary $\mathcal{A}$ introduces the errors $A_{je\delta} = A_{je} + \delta_A$ to the real model. $\mathcal{S}$ runs the functionality and the servers in H do not accept the t-sharings $[a_j]_t$ using the non-interactive Pedersen's VSS (equation 7.1). Thus, they detect the corrupt P_1, penalize him by redeeming his deposit transaction D_1 ₿ and get back their own deposits d ₿. Let $\{S_1, \ldots, S_t\} \in \mathsf{C}$

and $\{P_1, (S_{t+1}, \ldots, S_{2t+1})\} \in \mathsf{H}$ in the setup phase. $\mathcal{S}$ delays by the time-lock t_l or broadcasts wrong complains regarding the P_1's commitments A_{je}. This is similar to the same condition in the real model. P_1 eliminates the servers in C and gets compensated by redeeming the deposit d ₿ from each server in that set. Moreover, each server in H gains the reward D_1 ₿ from P_1 and gets back its own deposit d ₿. Let $\{S_1, \ldots, S_t\} \in \mathsf{C}$ and $\{P_2, (S_{t+1}, \ldots, S_{2t+1})\} \in \mathsf{H}$ in the computation phase. The simulator $\mathcal{S}$ sends the wrong encrypted shares $c_{i\delta}$ and $c'_{i\delta}$ to P_2. This is analogous if $\mathcal{A}$ introduces the errors $c_{i\delta} = c_i + \delta_c$ and $c'_{i\delta} = c'_i + \delta_{c'}$ in the real model. $\mathcal{S}$ executes the functionality and P_2 detects a faulty server using the extension of the Pedersen's commitments in equation 7.2 which can be written as:

$$\begin{aligned} \prod_{j=0}^{k} E_j^{\alpha^j} \bmod p &= \prod_{j=0}^{k} g^{\alpha^j \cdot [a_j]_t} \cdot h^{\alpha^j \cdot [a'_j]_t} \bmod p \\ &= g^{[f(\alpha)]_t} \cdot h^{[v']_t} \bmod p \end{aligned}$$

P_2 redeems the deposit transaction d ₿ of each server in C and eliminates it. Also, each server in H achieves the reward transaction D_2 ₿ and gets back its own deposit d ₿.

P_2 accepts c_i from an honest server after the verification stage which can be shown as:

$$c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\sum_{j=0}^{k} \alpha^j \cdot [a_j]_t)$$

which clearly is an encrypted share of the output. P_2 invokes the decryption algorithm to gain the share $[f(\alpha)]_t \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c_i)$. He gathers at least $t+1$ shares from the honest servers and reconstructs the output $f(\alpha)$.

P_1 maintains the privacy of his polynomial $f(x)$ using the unconditional security of the secret sharing and the Pedersen's VSS scheme, and P_2 employs the IND-CFA security of the Paillier cryptosystem to preserve the privacy of his input α. Note that a P_2 is only allowed to evaluate at most $k-1$ values from the same sender P_1. □

7.5 Conclusion

DOPE is a variant of two-party computation which is the significant building block of many cryptographic models and privacy-preserving algorithms. We present the first fair DOPE protocol where an honest cloud server gains reward for performing a computation service while a corrupt server has to pay some penalty for conducting the malicious behaviour via Bitcoin deposit transactions [62]. This can be achieved by using the properties scripts and time-lock in a Bitcoin transaction as a decentralised means of electronic payment without the need of a trusted third party. Our scheme includes two separate phases: setup and computation. The sender party P_1 interacts with the cloud servers in the setup phase while the receiver party P_2 communicates with the servers in the computation phase. This implies that the computation phase can be implemented at any time well in advance of the setup phase. P_1 distributes his polynomial among the servers and commits to the shares using the non-interactive Pedersen's commitment scheme which are checked by the servers. Each server employs one round of homomorphic feature of the Paillier cryptosystem to compute an encrypted share of the output, and P_2 verifies the share and detects any corrupt party.

Our protocol maintains the security against an active adversary corrupting a coalition of P_1 and up to t cloud servers in the setup phase and a coalition of maximum t servers in the computation phase in the presence of honest majority of the servers. The communication complexity is bounded to $O(kt)$ field elements which is the same as that in the previous DOPE protocols [61, 27], while the fairness property is also achieved in our scheme.

Chapter 8

Conclusion

This Chapter presents a brief summary of the thesis results and some potential directions for future research.

In this thesis, we investigate on the notable research field multi-party computation (MPC) and the related field distributed oblivious polynomial evaluation (DOPE). More specifically, several new schemes were proposed to improve on the efficiency of the existing protocols in these fields. Namely, we utilize the idea of MPC from pre-distributed information to present the following contributions in MPC:

- An unconditionally secure MPC protocol using a third party initializer (Chapter 2) [58].
- A fast unconditionally secure MPC scheme with multi-depths multiplicative gates (Chapter 3) [60].
- An efficient scheme for distributed keys generation of threshold Paillier cryptosystem using MPC (Chapter 4) [59].

DOPE is a variant of two-party computation and it is relatively a new research area. We also give the following contributions to make DOPE more secure, efficient and practical in the world of cloud computing service:

- The first verifiable and private DOPE scheme secure against an active adversary (Chapter 5) [61].

- A lightweight DOPE protocol by outsourcing the heavy computation to the cloud servers using threshold cryptography (Chapter 6) [63].
- The first fair and robust DOPE scheme via Bitcoin deposits in cloud computing service (Chapter 7) [62].

While these results show significant contributions, there are still several interesting potential research directions to be explored in the fields of MPC and DOPE as follows:

* For the case of fast MPC with multi-levels multiplicative gates, an interesting topic could be that the participants only hold the shares of the inputs. In that case, the input holders may not exist in the protocol.

* Another potential idea for the future research is to conduct more studies in commodity-based MPC. With the recent development of cloud computing, companies and parties can delegate and outsource their MPC systems to the cloud servers while the security conditions are also satisfied.

* In DOPE, an interesting topic could be the investigation on unconditionally secure DOPE against an active adversary without using computational (encryption) assumptions and oblivious transfer in the system.

* Also, another potential future research in DOPE is to conduct an actual implementation of a fair DOPE scheme in the real world of cloud computing. It could be the scheme presented in this thesis or any other more efficient model.

References

[1] Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: Proceedings of the 2000 ACM SIGMOD international conference on Management of data. pp. 439–450 (2000)

[2] Andrychowicz, M., Dziembowski, S., Malinowski, D., Mazurek, Ł.: Fair two-party computations via bitcoin deposits. In: Financial Cryptography and Data Security: FC 2014 Workshops, BITCOIN and WAHC 2014, Christ Church, Barbados, March 7, 2014, Revised Selected Papers 18. pp. 105–121. Springer (2014)

[3] Andrychowicz, M., Dziembowski, S., Malinowski, D., Mazurek, Ł.: Secure multiparty computations on bitcoin. Communications of the ACM 59(4), 76–84 (2016)

[4] Baudron, O., Fouque, P.A., Pointcheval, D., Stern, J., Poupard, G.: Practical multi-candidate election system. In: Proceedings of the twentieth annual ACM symposium on Principles of distributed computing. pp. 274–283 (2001)

[5] Baum, C., Damgård, I., Toft, T., Zakarias, R.: Better preprocessing for secure multiparty computation. In: International Conference on Applied Cryptography and Network Security. pp. 327–345. Springer (2016)

[6] Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Annual International Cryptology Conference. pp. 420–432. Springer (1991)

[7] Beaver, D.: Commodity-based cryptography. In: Proceedings of the twenty-

ninth annual ACM symposium on Theory of computing. pp. 446–455 (1997)

[8] Beerliová-Trubíniová, Z., Hirt, M.: Perfectly-secure mpc with linear communication complexity. In: Theory of Cryptography Conference. pp. 213–230. Springer (2008)

[9] Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Proceedings of the twentieth annual ACM symposium on Theory of computing. pp. 1–10 (1988)

[10] Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali, pp. 351–371 (2019)

[11] Benaloh, J.C.: Secret sharing homomorphisms: Keeping shares of a secret secret. In: Conference on the theory and application of cryptographic techniques. pp. 251–260. Springer (1986)

[12] Bendlin, R., Damgård, I., Orlandi, C., Zakarias, S.: Semi-homomorphic encryption and multiparty computation. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 169–188. Springer (2011)

[13] Bilgin, B., Gierlichs, B., Nikova, S., Nikov, V., Rijmen, V.: Higher-order threshold implementations. In: Advances in Cryptology–ASIACRYPT 2014: 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, ROC, December 7-11, 2014, Proceedings, Part II 20. pp. 326–343. Springer (2014)

[14] Blier, H., Tapp, A.: A single initialization server for multi-party cryptography. In: Information Theoretic Security: Third International Conference, ICITS

2008, Calgary, Canada, August 10-13, 2008. Proceedings 3. pp. 71–85. Springer (2008)

[15] Blundo, C., D'Arco, P., De Santis, A., Stinson, D.: On unconditionally secure distributed oblivious transfer. Journal of Cryptology 20(3), 323–373 (2007)

[16] Blundo, C., D'Arco, P., De Santis, A., Stinson, D.R.: New results on unconditionally secure distributed oblivious transfer. In: International Workshop on Selected Areas in Cryptography. pp. 291–309. Springer (2002)

[17] Bogdanov, D., Kamm, L., Kubo, B., Rebane, R., Sokk, V., Talviste, R.: Students and taxes: a privacy-preserving social study using secure computation. Cryptology ePrint Archive (2015)

[18] Bogetoft, P., Christensen, D.L., Damgård, I., Geisler, M., Jakobsen, T., Krøigaard, M., Nielsen, J.D., Nielsen, J.B., Nielsen, K., Pagter, J., et al.: Secure multiparty computation goes live. In: International Conference on Financial Cryptography and Data Security. pp. 325–343. Springer (2009)

[19] Boneh, D., Franklin, M.: Efficient generation of shared rsa keys. In: Annual international cryptology conference. pp. 425–439. Springer (1997)

[20] Bultel, X., Das, M.L., Gajera, H., Gérault, D., Giraud, M., Lafourcade, P.: Verifiable private polynomial evaluation. In: Provable Security: 11th International Conference, ProvSec 2017, Xi'an, China, October 23-25, 2017, Proceedings. pp. 487–506. Springer (2017)

[21] Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: Proceedings 42nd IEEE Symposium on Foundations of Computer Science. pp. 136–145. IEEE (2001)

[22] Cascudo, I., Cramer, R., Xing, C., Yuan, C.: Amortized complexity of

information-theoretically secure mpc revisited. In: Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part III 38. pp. 395–426. Springer (2018)

[23] Chang, Y.C., Lu, C.J.: Oblivious polynomial evaluation and oblivious neural learning. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 369–384. Springer (2001)

[24] Chaum, D., Crépeau, C., Damgard, I.: Multiparty unconditionally secure protocols. In: Proceedings of the twentieth annual ACM symposium on Theory of computing. pp. 11–19 (1988)

[25] Cianciullo, L., Ghodosi, H.: Efficient information theoretic multi-party computation from oblivious linear evaluation. In: IFIP International Conference on Information Security Theory and Practice. pp. 78–90. Springer (2018)

[26] Cianciullo, L., Ghodosi, H.: Unconditionally secure distributed oblivious polynomial evaluation. In: International Conference on Information Security and Cryptology. pp. 132–142. Springer (2018)

[27] Cianciullo, L., Ghodosi, H.: Unconditionally secure distributed oblivious polynomial evaluation. In: Information Security and Cryptology–ICISC 2018: 21st International Conference, Seoul, South Korea, November 28–30, 2018, Revised Selected Papers. pp. 132–142. Springer (2019)

[28] Cock, M.d., Dowsley, R., Nascimento, A.C., Newman, S.C.: Fast, privacy preserving linear regression over distributed datasets based on pre-distributed data. In: Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security. pp. 3–14 (2015)

[29] Corniaux, C.L., Ghodosi, H.: A verifiable 1-out-of-n distributed oblivious transfer protocol. IACR Cryptol. ePrint Arch. 2013, 63 (2013)

[30] Couteau, G.: A note on the communication complexity of multiparty computation in the correlated randomness model. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 473–503. Springer (2019)

[31] Cramer, R., Damgård, I., Nielsen, J.B.: Multiparty computation from threshold homomorphic encryption. In: International conference on the theory and applications of cryptographic techniques. pp. 280–300. Springer (2001)

[32] Damgård, I., Damgård, K., Nielsen, K., Nordholt, P.S., Toft, T.: Confidential benchmarking based on multiparty computation. In: International Conference on Financial Cryptography and Data Security. pp. 169–187. Springer (2016)

[33] Damgård, I., Geisler, M., Krøigaard, M., Nielsen, J.B.: Asynchronous multiparty computation: Theory and implementation. In: International workshop on public key cryptography. pp. 160–179. Springer (2009)

[34] Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In: International workshop on public key cryptography. pp. 119–136. Springer (2001)

[35] Damgård, I., Keller, M., Larraia, E., Pastro, V., Scholl, P., Smart, N.P.: Practical covertly secure mpc for dishonest majority–or: breaking the spdz limits. In: European Symposium on Research in Computer Security. pp. 1–18. Springer (2013)

[36] Damgård, I., Koprowski, M.: Practical threshold rsa signatures without a trusted dealer. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 152–165. Springer (2001)

[37] Damgård, I., Mikkelsen, G.L.: Efficient, robust and constant-round distributed rsa key generation. In: Theory of Cryptography Conference. pp. 183–200. Springer (2010)

[38] Damgård, I., Nielsen, J.B.: Scalable and unconditionally secure multiparty computation. In: Annual International Cryptology Conference. pp. 572–590. Springer (2007)

[39] Damgård, I., Orlandi, C.: Multiparty computation for dishonest majority: From passive to active security at low cost. In: Annual cryptology conference. pp. 558–576. Springer (2010)

[40] Damgård, I., Pastro, V., Smart, N., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Annual Cryptology Conference. pp. 643–662. Springer (2012)

[41] David, B., Dowsley, R., Katti, R., Nascimento, A.C.: Efficient unconditionally secure comparison and privacy preserving machine learning classification protocols. In: Provable Security: 9th International Conference, ProvSec 2015, Kanazawa, Japan, November 24-26, 2015, Proceedings. pp. 354–367. Springer (2015)

[42] Dowsley, R., Van De Graaf, J., Marques, D., Nascimento, A.C.: A two-party protocol with trusted initializer for computing the inner product. In: Information Security Applications: 11th International Workshop, WISA 2010, Jeju Island, Korea, August 24-26, 2010, Revised Selected Papers 11. pp. 337–350. Springer (2011)

[43] Fouque, P.A., Poupard, G., Stern, J.: Sharing decryption in the context of voting or lotteries. In: International Conference on Financial Cryptography. pp. 90–104. Springer (2000)

[44] Frankel, Y., Gemmell, P., MacKenzie, P.D., Yung, M.: Optimal-resilience proactive public-key cryptosystems. In: Proceedings 38th Annual Symposium on Foundations of Computer Science. pp. 384–393. IEEE (1997)

[45] Frankel, Y., MacKenzie, P.D., Yung, M.: Robust efficient distributed rsa-key generation. In: Proceedings of the thirtieth annual ACM symposium on Theory of computing. pp. 663–672 (1998)

[46] Frederiksen, T.K., Keller, M., Orsini, E., Scholl, P.: A unified approach to mpc with preprocessing using ot. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 711–735. Springer (2015)

[47] Freedman, M.J., Ishai, Y., Pinkas, B., Reingold, O.: Keyword search and oblivious pseudorandom functions. In: Theory of Cryptography Conference. pp. 303–324. Springer (2005)

[48] Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: International conference on the theory and applications of cryptographic techniques. pp. 1–19. Springer (2004)

[49] Gajera, H., Giraud, M., Gérault, D., Das, M.L., Lafourcade, P.: Verifiable and private oblivious polynomial evaluation. In: IFIP International Conference on Information Security Theory and Practice. pp. 49–65. Springer (2019)

[50] Gajera, H., Giraud, M., Gérault, D., Das, M.L., Lafourcade, P.: Verifiable and private oblivious polynomial evaluation. In: Information Security Theory and Practice: 13th IFIP WG 11.2 International Conference, WISTP 2019, Paris, France, December 11–12, 2019, Proceedings. pp. 49–65. Springer (2020)

[51] Gavin, G., Minier, M.: Oblivious multi-variate polynomial evaluation. In: Progress in Cryptology-INDOCRYPT 2009: 10th International Conference on

Cryptology in India, New Delhi, India, December 13-16, 2009. Proceedings 10. pp. 430–442. Springer (2009)

[52] Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Robust and efficient sharing of rsa functions. In: Annual International Cryptology Conference. pp. 157–172. Springer (1996)

[53] Gennaro, R., Rabin, M.O., Rabin, T.: Simplified vss and fast-track multiparty computations with applications to threshold cryptography. In: Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing. pp. 101–111 (1998)

[54] Ghodosi, H., Pieprzyk, J.: Multi-party computation with omnipresent adversary. In: International Workshop on Public Key Cryptography. pp. 180–195. Springer (2009)

[55] Gilboa, N.: Two party rsa key generation. In: Annual International Cryptology Conference. pp. 116–129. Springer (1999)

[56] Goethals, B., Laur, S., Lipmaa, H., Mielikäinen, T.: On private scalar product computation for privacy-preserving data mining. In: Information Security and Cryptology–ICISC 2004: 7th International Conference, Seoul, Korea, December 2-3, 2004, Revised Selected Papers 7. pp. 104–120. Springer (2005)

[57] Guo, L., Fang, Y., Li, M., Li, P.: Verifiable privacy-preserving monitoring for cloud-assisted mhealth systems. In: 2015 IEEE Conference on Computer Communications (INFOCOM). pp. 1026–1034. IEEE (2015)

[58] Hamidi, A., Ghodosi, H.: Secure multi-party computation using predistributed information from an initializer. In: Security and Privacy: Second International Conference, ICSP 2021, Jamshedpur, India, November 16–17, 2021, Proceedings 2. pp. 111–122. Springer (2021)

[59] Hamidi, A., Ghodosi, H.: Efficient distributed keys generation of threshold paillier cryptosystem. In: International Conference on Information Technology and Communications Security. pp. 117–132. Springer (2022)

[60] Hamidi, A., Ghodosi, H.: Unconditionally fast secure multi-party computation with multi-depths gates using pre-computed information. In: Proceedings of Seventh International Congress on Information and Communication Technology: ICICT 2022, London, Volume 2. pp. 329–340. Springer (2022)

[61] Hamidi, A., Ghodosi, H.: Verifiable dope from somewhat homomorphic encryption, and the extension to dot. In: Science of Cyber Security: 4th International Conference, SciSec 2022, Matsue, Japan, August 10–12, 2022, Revised Selected Papers. pp. 105–120. Springer (2022)

[62] Hamidi, A., Ghodosi, H.: Fair distributed oblivious polynomial evaluation via bitcoin deposits: Compute-as-a-service. In: Nordic Conference on Secure IT Systems. pp. 73–86. Springer (2023)

[63] Hamidi, A., Ghodosi, H.: Outsourcing verifiable distributed oblivious polynomial evaluation from threshold cryptography. In: International Conference on Information and Communications Security. pp. 235–246. Springer (2023)

[64] Hanaoka, G., Imai, H., Mueller-Quade, J., Nascimento, A.C., Otsuka, A., Winter, A.: Information theoretically secure oblivious polynomial evaluation: Model, bounds, and constructions. In: Australasian Conference on Information Security and Privacy. pp. 62–73. Springer (2004)

[65] Hazay, C., Lindell, Y.: Efficient oblivious polynomial evaluation with simulation-based security. Cryptology ePrint Archive (2009)

[66] Hazay, C., Mikkelsen, G.L., Rabin, T., Toft, T., Nicolosi, A.A.: Efficient rsa key generation and threshold paillier in the two-party setting. Journal of

Cryptology 32(2), 265–323 (2019)

[67] Hemenway, B., Lu, S., Ostrovsky, R., Welser Iv, W.: High-precision secure computation of satellite collision probabilities. In: Security and Cryptography for Networks: 10th International Conference, SCN 2016, Amalfi, Italy, August 31–September 2, 2016, Proceedings 10. pp. 169–187. Springer (2016)

[68] Hirt, M., Maurer, U., Przydatek, B.: Efficient secure multi-party computation. In: International conference on the theory and application of cryptology and information security. pp. 143–161. Springer (2000)

[69] Hohenberger, S., Lysyanskaya, A.: How to securely outsource cryptographic computations. In: Theory of Cryptography: Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005. Proceedings 2. pp. 264–282. Springer (2005)

[70] Ishai, Y., Kushilevitz, E., Meldgaard, S., Orlandi, C., Paskin-Cherniavsky, A.: On the power of correlated randomness in secure computation. In: Theory of Cryptography Conference. pp. 600–620. Springer (2013)

[71] Keller, M., Orsini, E., Scholl, P.: Mascot: faster malicious arithmetic secure computation with oblivious transfer. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 830–842 (2016)

[72] Kiayias, A., Leonardos, N., Lipmaa, H., Pavlyk, K., Tang, Q.: Optimal rate private information retrieval from homomorphic encryption. Proc. Priv. Enhancing Technol. 2015(2), 222–243 (2015)

[73] Kiayias, A., Zhou, H.S., Zikas, V.: Fair and robust multi-party computation using a global transaction ledger. In: Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications

of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35. pp. 705–734. Springer (2016)

[74] Kumaresan, R., Vaikuntanathan, V., Vasudevan, P.N.: Improvements to secure computation with penalties. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 406–417 (2016)

[75] Li, H.D., Yang, X., Feng, D.G., Li, B.: Distributed oblivious function evaluation and its applications. Journal of Computer Science and Technology 19(6), 942–947 (2004)

[76] Lindell, Y.: How to simulate it–a tutorial on the simulation proof technique. Tutorials on the Foundations of Cryptography pp. 277–346 (2017)

[77] Lindell, Y., Pinkas, B., Smart, N.P., Yanai, A.: Efficient constant round multi-party computation combining bmr and spdz. In: Annual Cryptology Conference. pp. 319–338. Springer (2015)

[78] Micali, S., Goldreich, O., Wigderson, A.: How to play any mental game. In: Proceedings of the Nineteenth ACM Symp. on Theory of Computing, STOC. pp. 218–229. ACM (1987)

[79] Mohassel, P., Zhang, Y.: Secureml: A system for scalable privacy-preserving machine learning. In: 2017 IEEE symposium on security and privacy (SP). pp. 19–38. IEEE (2017)

[80] Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Decentralized business review p. 21260 (2008)

[81] Naor, M., Pinkas, B.: Oblivious transfer and polynomial evaluation. In: Proceedings of the thirty-first annual ACM symposium on Theory of computing. pp. 245–254 (1999)

[82] Naor, M., Pinkas, B.: Distributed oblivious transfer. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 205–219. Springer (2000)

[83] Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: SODA. vol. 1, pp. 448–457 (2001)

[84] Naor, M., Pinkas, B.: Oblivious polynomial evaluation. SIAM Journal on Computing 35(5), 1254–1281 (2006)

[85] Nikova, S., Rijmen, V., Schläffer, M.: Secure hardware implementation of nonlinear functions in the presence of glitches. Journal of Cryptology 24, 292–321 (2011)

[86] Nishide, T., Sakurai, K.: Distributed paillier cryptosystem without trusted dealer. In: International Workshop on Information Security Applications. pp. 44–60. Springer (2010)

[87] Otsuka, A., Imai, H.: Unconditionally secure electronic voting. Towards Trustworthy Elections: New Directions in Electronic Voting pp. 107–123 (2010)

[88] Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: International conference on the theory and applications of cryptographic techniques. pp. 223–238. Springer (1999)

[89] Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Annual international cryptology conference. pp. 129–140. Springer (1991)

[90] Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Advances in Cryptology—CRYPTO'91: Proceedings. pp. 129–140. Springer (2001)

[91] Rabin, T.: A simplified approach to threshold and proactive rsa. In: Annual International Cryptology Conference. pp. 89–104. Springer (1998)

[92] Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority. In: Proceedings of the twenty-first annual ACM symposium on Theory of computing. pp. 73–85. ACM (1989)

[93] Reparaz, O., Bilgin, B., Nikova, S., Gierlichs, B., Verbauwhede, I.: Consolidating masking schemes. In: Advances in Cryptology–CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I 35. pp. 764–783. Springer (2015)

[94] Rivest, R.: Unconditionally secure commitment and oblivious transfer schemes using private channels and a trusted initializer. Unpublished manuscript (1999)

[95] Scholl, P., Smart, N.P., Wood, T.: When it's all just too much: outsourcing mpc-preprocessing. In: Cryptography and Coding: 16th IMA International Conference, IMACC 2017, Oxford, UK, December 12-14, 2017, Proceedings 16. pp. 77–99. Springer (2017)

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

[97] Smart, N.P., Tanguy, T.: Taas: Commodity mpc via triples-as-a-service. In: Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop. pp. 105–116 (2019)

[98] Tompa, M., Woll, H.: How to share a secret with cheaters. journal of Cryptology 1(3), 133–138 (1989)

[99] Veugen, T., Attema, T., Spini, G.: An implementation of the paillier crypto system with threshold decryption without a trusted dealer. Cryptology ePrint Archive (2019)

[100] Yao, A.C.: Protocols for secure computations. In: 23rd annual symposium on foundations of computer science (sfcs 1982). pp. 160–164. IEEE (1982)