SELECTED APPLICATIONS OF MPC

by

Mohammad Ghaseminejad Raeini

A Dissertation Submitted to the Faculty of

The College of Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

Florida Atlantic University

Boca Raton, FL

August 2022

SELECTED APPLICATIONS OF MPC

by

Mohammad Ghaseminejad Raeini

This dissertation was prepared under the direction of the candidate's dissertation co-advisors, Dr. Feng-Hao Liu and Dr. Mehrdad Nojoumian, Department of Electrical Engineering and Computer Science, and has been approved by the members of his supervisory committee. It was submitted to the faculty of the College of Engineering and Computer Science and was accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

SUPERVISORY COMMITTEE:

Feng-Hao Liu
Feng-Hao Liu (Jul 11, 2022 13:57 EDT)

Feng-Hao Liu, Ph.D.
Dissertation Co-Advisor

[illegible]
Mehrdad Nojoumian (Jul 11, 2022 13:58 EDT)

Mehrdad Nojoumian, Ph.D.
Dissertation Co-Advisor

[illegible]

Hanqi Zhuang, Ph.D.

[illegible]
Reza Azarderakhsh (Jul 11, 2022 12:29 PDT)

Reza Azarderakhsh, Ph.D.

Edoardo Persichetti
Edoardo Persichetti (Jul 12, 2022 03:01 EDT)

Edoardo Persichetti, Ph.D.

[illegible]

Hanqi Zhuang, Ph.D.
Chair, Department of Electrical Engineering and Computer Science

[illegible]

Stella Batalama, Ph.D.
Dean, The College of Engineering and Computer Science

Robert W. Stackman Jr.
Robert W. Stackman Jr. (Jul 14, 2022 06:36 EDT)

Robert W. Stackman Jr., Ph.D.
Dean, Graduate College

Date

ACKNOWLEDGEMENTS

I would like to acknowledge and sincerely express my gratitude to all of those who helped and supported me during my PhD program. Particularly, I am extremely thankful to my PhD program's co-advisers, Dr. Feng-Hao Liu and Dr. Mehrdad Nojoumian, for their support of my research as well as for the time they devoted to advise me with their outstanding knowledge.

I am also very grateful to other members of the committee for my PhD program (including Dr. Hanqi Zhuang, Dr. Reza Azarderakhsh, and Dr. Edoardo Persichetti) for taking time to evaluate my PhD research and for providing me with helpful comments to improve my work.

I would also like to thank the department of computer science and electrical engineering (CEECS) as well as Florida Atlantic University (FAU) for providing me with the opportunity to pursue my PhD program.

Last but not least, I am truly thankful to my family for their continuous help and support during all my life, and particularly for supporting me in all of my academic careers.

ABSTRACT

Author:	Mohammad Ghaseminejad Raeini
Title:	Selected Applications of MPC
Institution:	Florida Atlantic University
Dissertation Co-Advisors:	Dr. Feng-Hao Liu Dr. Mehrdad Nojoumian
Degree:	Doctor of Philosophy
Year:	2022

Secure multiparty computation (secure MPC) is a computational paradigm that enables a group of parties to evaluate a public function on their private data without revealing the data (i.e., by preserving the privacy of their data). This computational approach, sometimes also referred to as secure function evaluation (SFE) and privacy-preserving computation, has attracted significant attention in the last couple of decades. It has been studied in different application domains, including in privacy-preserving data mining and machine learning, secure signal processing, secure genome analysis, sealed-bid auctions, etc. There are different approaches for realizing secure MPC. Some commonly used approaches include secret sharing schemes, Yao's garbled circuits, and homomorphic encryption techniques.

The main focus of this dissertation is to further investigate secure multiparty computation as an appealing area of research and to study its applications in different domains. We specifically focus on secure multiparty computation based on secret sharing and fully homomorphic encryption (FHE) schemes. We review the important theoretical foundations of these approaches and provide some novel applications for each of them. For the fully homomorphic encryption (FHE) part, we mainly focus on

FHE schemes based on the LWE problem [142] or RLWE problem [109]. Particularly, we provide a C++ implementation for the ring variant of a third generation FHE scheme called the approximate eigenvector method (a.k.a., the GSW scheme) [67]. We then propose some novel approaches for homomorphic evaluation of common functionalities based on the implemented (R)LWE [142] and [109] and RGSW [38,58] schemes. We specifically present some constructions for homomorphic computation of pseudorandom functions (PRFs). For secure computation based on secret sharing [150], we provide some novel protocols for secure trust evaluation (STE). Our proposed STE techniques [137] enable the parties in trust and reputation systems (TRS) to securely assess their trust values in each other while they keep their input trust values private. We would like to remark that trust and reputation are social mechanisms which can be considered as soft security measures that complement hard security measures (e.g., cryptographic and secure multiparty computation techniques).

We hope our research can contribute to advancing the interesting area of secure computation. Besides improving existing secure multiparty computation protocols, our proposed FHE-based constructions and secure protocols can potentially be used for providing more secure and robust data and computation infrastructures.

To those who have supported me during my PhD program and provided me with the opportunity to pursue my PhD program, including the Department of Electrical Engineering and Computer Science as well as Florida Atlantic University

SELECTED APPLICATIONS OF MPC

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Secure multiparty computation (secure MPC) is a branch of modern cryptography that has attracted significant attention in the last couple of decades. This computational paradigm enables a group of (possibly untrusted) parties to evaluate a public function on their private data without revealing their input data. Secure MPC has been vastly studied from both theoretical and application point of views. It has been investigated in various application domains, e.g., in privacy-preserving data mining and machine learning, secure signal processing, and secure genome analysis (privacy-preserving genome analysis), seal-bid auctions etc.

There are different approaches that can be used for implementing secure multi-party computation protocols. Some commonly used approaches include secret sharing schemes [150], [20], Yao's garbled circuits [169], and homomorphic encryption techniques [146], [131], [60], [66]. Each of these approaches has their own advantageous and disadvantageous. In fact, none of the mentioned approaches can surpass other approaches in all aspects (mainly in computation and communication costs). While some approaches can be computationally efficient, they can have too high communication overhead (e.g., secret sharing-based schemes). Those approaches which are efficient in terms of communication costs can be computationally intensive (e.g., fully homomorphic-based schemes). What's more, some approaches can support multiple number of participants (i.e., more than three parties) in a multiparty computation scenario, whereas some approaches might not be applicable for more than a few parties (i.e., they can support two or at most three parties).

1.1 MOTIVATION AND PROBLEM STATEMENT

Secure computation provides us with promising solutions specifically for data privacy-related issues. However, this area of research is still in its infancy and more research and development are needed in order for secure MPC solutions to be deployed in real world applications. One main bottleneck in the way of secure computation is that none of the secure computation approaches can provide solutions that can be utilized in general-purpose applications. That's because there are quite a few approaches that allow secure computation and each approach has its own strengths and weaknesses. Just as an example, secure computation using homomorphic encryption was envisioned in 1978 [146]. However, providing a fully homomorphic encryption scheme was an open problem until 2009 [66]. Even though Gentry in [66] provided a blueprint for fully homomorphic encryption techniques, even after significant amount of research and development FHE schemes are not yet practical nor efficient enough. The main challenge with FHE schemes is that such schemes are computationally expensive, mainly because of large cryptographic keys and/or costly arithmetic operations such as the multiplication operation. Another challenge with homomorphic encryption schemes, which seems to be overlooked, is that not all homomorphic encryption techniques support multi-key encryption.

On the other hand, secure computation based on secret sharing provides solutions which are computationally efficient, but such solutions can suffer from high communication cost. Although the multiplication operation in secure MPC based on secret sharing is also a bottleneck, secure computation based on secret sharing has some appealing properties. Specifically, it provides information-theoretical security if the underlying secret sharing scheme is information-theoretically secure. Moreover, in secure MPC based on secret sharing there is no need for any cryptographic keys and any finite number of parties can participate in a multiparty computation protocol.

The above discussion, besides the fact that secure computation has attracted significant attention in the last couple of decades, motivated us to focus on secure computation in this dissertation. After briefly reviewing different approaches in secure computation and considering the pros and cons of each approach we decided to devote the majority of the dissertation to secure multiparty computation based on secret sharing and fully homomorphic encryption (FHE) schemes. These two approaches have attracted significant attention in the research community and several secure computation protocols have been implemented based on these approaches.

1.2 DISSERTATION'S OBJECTIVES

In this dissertation, we focus on secure multiparty computation (secure MPC). We first study different approaches that are used in secure MPC and highlight the strengths and weaknesses of each approach. We then provide the preliminaries and theoretical bases of secure MPC and detail the main building blocks which are required for secure computation. We devote the majority of this dissertation to secure multiparty computation based on Shamirs (t, n)-threshold secret sharing [150] and fully homomorphic encryption (FHE) schemes. Specifically, we provide a C++ implmentation of the ring variant of a well-known FHE scheme called the approximate eigenvector method (a.k.a., the GSW scheme [67]); and utilize our implementation for homomorphic computation of pseudorandom functions (PRFs). The ring variant of the GSW FHE scheme is usually referred to as Ring-GSW or RGSW [58] and [38]. Furthermore, we provide some novel protocols for secure trust evaluation (STE) [137] based on Shamir's secret sharing scheme [150].

The fully homomorphic encryption scheme that we implement is based on the ring variant of a third generation FHE scheme, called the approximate eigenvector method also known as GSW scheme [67]. The security of the most of the FHE scheme is based on the learning with errors (LWE) problem [142] or its ring variant called ring learning with errors (RLWE) [109]. Our implementation of the ring variant of the

GSW scheme [67] is in C++ and uses NFLlib C++ library [5]. The NFLlib library [5] is a fast library for lattice-based cryptography computations. Specifically, it provides the required data structure and functionalities for polynomial arithmetic, such as the number theoretic transformation (NTT) [135] representation of a polynomial over a finite field and fast polynomial multiplication. Our proposed secure trust evaluation (STE) mechanisms [137] allow the parties in trust and reputation systems (TRS) to securely evaluate their trust values in each other without revealing such values (which are considered as private data). We propose three different STE techniques, namely STE based on multipath trust propagation [156], STE based on referral chain approach [156], and STE based on the aggregation of trust evidence [166]. Our secure trust evaluation techniques are based on Shamir's secret sharing scheme [150] (although other secret sharing schemes, e.g., Blakley's scheme [20], can be utilized). We would like to emphasize that trust and reputation systems have applications in different domains including in Internet of Things (IoT), cloud computing, social networks, intelligent transportation systems (ITS), peer-2-peer (P2P) networks, vehicle-2-vehicle (V2V) networks, wireless sensor networks (WSN), ad-hoc networks, and web services.

Our proposed secure MPC solutions for the above applications enhance the current solutions by preserving the privacy of the entities' data in the trust and reputation systems. In particular, there are various studies stating the security and privacy in an important issue in TRS systems [76, 106, 132, 144, 147]. Thus, our proposed solutions can be utilized to address such privacy issues.

1.3 DISSERTATION'S OUTLINE AND STRUCTURE

The current chapter (Chapter 1) provides the problem statement, motivation for our research in this dissertation, dissertation's objectives, as well as the structure of the dissertation. In chapter 2, we present the preliminaries and theoretical bases that are used throughout this dissertation. These include Shamir's secret sharing scheme [150] and the foundations of fully homomorphic encryption, e.g., the LWE [142] and RLWE [109] problems, the approximate eigenvector (a.k.a., GSW) scheme [67]. In chapter 3, we provide an introduction to secure multiparty computation (secure MPC) and a brief history of this area of research. We further detail different approaches that are commonly used for secure computation and the building blocks for secure computation. In addition, we provide a brief literature review on the previous research and development related to secure MPC and the developed secure MPC tools, and libraries.

In chapter 4, we describe the RGSW fully homomorphic encryption scheme (i.e., the ring variant of the GSW scheme [67]) and its C++ implementation. Specifically, we discuss what the plaintext and ciphertetxt spaces of the RGSW scheme [38, 58] are, how the public and secret keys for this scheme are generated, and how encryption and decryption algorithms work. We also describe our C++ implemenation of the RGSW scheme and explain how it works.

In chapter 6, we propose two secure trust evaluation (STE) techniques [137] that are based on multipath and referral chain trust propagation methods [156]. In addition, we propose a secure network routing protocol that allows the nodes in a network to securely find some high-quality routes in a network, e.g., a wireless sensor network (WSN). In chapter 7, we present our third proposed secure trust evaluation (STE) technique which is based on the aggregation of trust evidence [166]. Our proposed STE techniques can be utilized in different trust and reputation systems (TRS) to address or alleviate data security and privacy-related issues. Finally, chapter 8 provides the concluding remarks and highlights potential paths for future research directions.

CHAPTER 2
PRELIMINARIES AND THEORETICAL BASES

In this section, we provide the preliminaries and theoretical bases that we need throughout the dissertation. As we mainly focus on secure computation based on fully homomorphic encryption (FHE) and secret sharing schemes, we provide the main theoretical foundations behind these two approaches. Homomorphic encryption schemes are mostly based on hard problems in lattices and the learning with error problems (particularly LWE and RLWE) [142] and [109]. Besides these, most FHE schemes are constructed on the ring of polynomials. A commonly used ring is the ring of polynomials modulo a cyclotomic polynomial (such as X^{N+1} where N is a power of two, e.g., 1024). Secret sharing schemes, also rely on polynomials. However, the way that data is encoded in each approach is different. In homomorphic encryption the data is encrypted using some encryption algorithm and the computations are performed on encrypted data; whereas in secret sharing the data is shared using a secret sharing scheme and computations are performed on the shares of the data.

Regardless of the secure computation approach, there are some commonly used building blocks which are required for secure computation, such as secure comparison (a.k.a., the greater-than operation ($>$ or) or Yao's millionaire's problem [169]) and secure dot-product of two vectors. In this chapter of the dissertation, we provide a brief description of the required theoretical foundations for the schemes and algorithms used in the dissertation.

2.1 SHAMIR'S SECRET SHARING SCHEME

One of the common approaches in secure computation is secret sharing. A well-known secret sharing scheme was proposed by Shamir, which is known as Shamir's (t, n)-threshold secret sharing scheme [150]. Briefly speaking, this scheme allows us to distribute a secret among n parties and the secret is reconstructable only if t parties collaborate and provide their correct shares (pieces of secret). This scheme works as follows [150]:

In order to share a secret, the party who want to share its secret first fixes a finite field such as Z_p where p is a prime number. The party then selects a polynomial of degree $t-1$ over $Z_p[x]$ where the constant term of the polynomial is his secret and its other coefficients are random numbers in Z_p. Note that t is called the threshold. To share the secret into n pieces, the party evaluates the selected polynomial over n different points in Z_p. To reconstruct a shared secret from its pieces, at least t pieces of the secret are required. To do so, by having t pieces of a secret the party can use the Lagrange interpolation to reconstruct the polynomial which was selected for generating the shares of the secret. Then in order to get the secret, the party needs to evaluate the polynomial on origin, i.e., to calculate $f(0)$.

It is interesting to see how Shamir's secret sharing scheme works from the mathematical point of view. This scheme in fact encodes a secret into a set of points on a polynomial (more specifically on a polynomial of degree $t-1$ over $Z_p[x]$). A good property of polynomials is that by having sufficient number of points on a polynomial one can reconstruct the polynomial. That is, a polynomial of degree $t-1$ can be uniquely determined using at least t points of that polynomial. This propoerty of polynomials has also been utilized in Reed-Solomon error correcting codes [141]. In fact, Shamir's secret sharing scheme [150] is closely related to Reed-Solomon error correcting codes [141]. This interesting observation was first made by McEliece and Sarwate [115].

2.2 FULLY HOMOMORPHIC ENCRYPTION (FHE)

Homomorphic encryption, particularly fully homomorphic encryption (FHE), is one the promising solutions for performing computation on encrypted data. The idea of secure computation using homomorphic encryption was first envisioned in [146] and was boomed by the introduction of the first plausible fully homomorphic encryption (FHE) by Craig Gentry [66]. After decades of research in this domain, there are now several implementations and open-source libraries implementing FHE schemes. Some examples include INTEL's HEXL library[1], Microsoft SEAL's library [36], HElib[2], Palisade[3], TFHE [38] etc. The theory behind most FHE schemes relies on hard problems in lattices as well as the learning with errors (LWE) problem [142] or its ring variant, i.e., the RLWE problem [109]. Simply put, a lattice is a discrete subgroup of R^n, e.g., Z^n (where Z is the set of integer numbers). The learning with errors (LWE) problem [142] is the problem of finding the solution of a system of linear equation over a finite field or ring in the presence of noise (error). While solving a system of linear equation is an easy problem to solve and there are different efficient solutions for that (such as the Gaussian elimination), solving a system of linear equation with some small noise added to each equation is conjectured to be a hard computational problem [142]. Most of the state-of-the-art and promising FHE schemes, that have been implemented in the libraries, are based on the ring variant of the LWE problem (commonly referred to as RLWE) [109].

[1]https://github.com/intel/hexl
[2]https://github.com/homenc/HElib
[3]https://palisade-crypto.org

CHAPTER 3
SECURE MULTIPARTY COMPUTATION (SECURE MPC)

3.1 THE HISTORY OF SECURE MULTIPARTY COMPUTATION

Secure multiparty computation (secure MPC) is a branch of modern cryptography that has attracted significant attention in the last couple of decades. Early works on secure computation dates back to seminal papers of Andrew Yao [169], Rivest et al. [146], and Shamir [150]. In [169] Yao introduced the idea of secure computation as well as the Yao's millionaires problem. On the other hand, in [146] the pioneers of cryptography envisioned the idea of secure computation using mathematical homomorphisms. Shamir and Blakley introduced the notion of secret sharing independently in [150] and [20].

Although the main theme of [169], [146], [150] is secure computation, they have taken different approaches for offering secure computation mechanisms. In [169], Yao introduced the millionaires problem in which two millionaires want to determine who is richer, but they are not willing to disclose the amount of wealth they each have. Yao also proposed some solutions based on public-key cryptography for his millionaires problem. The proposed solution for his problem was the first example of a secure computation protocol [169]. Later, Yao introduced the idea of garbled circuits [170] which is one of the common secure computation approaches.

After the introduction of the core ideas behind secure computation, there have been extensive amounts of research based on each approach. Particularly, secure computation based on fully homomorphic encryption became really popular by a breakthrough work by Gentry in 2009 [66].

3.2 COMMON APPROACHES FOR SECURE COMPUTATION

There are different approaches to implement a secure multiparty computation scheme. These techniques are based on three cryptographic primitives, i.e., secret sharing [20, 150], homomorphic encryption [60, 66, 131, 146] and Yao's garbled circuits [170].

3.2.1 Secure Computation based on Secret Sharing

Secret sharing is one of the dominant approaches used in secure multiparty computation. In this approach, the participating parties use a secret sharing scheme, e.g., Shamir's scheme [150], to share their secrets (private data). In order to emulate a secure MPC protocol, the parties then perform computations on the shares of their data, rather than directly on their data. Since the shares of the private data are random values, no information about that data is revealed.

This approach of secure computation typically involves a couple of steps. As shown Fig. 3.1, the first phase is share generation and distribution. In this step, the participating parties share their data and distribute them among each other. The second phase is performing computations on the shares of data. In this phase each party locally performs the required computations on the shares of its data. The third and last step is reconstruction of the computations' result. After performing computation on the data, the parties send their updated result to a party (or a dealer) and the party (or dealer) reconstructs the result by using the Lagrange interpolation.

An advantage of secure computation protocols based on secret sharing is that such protocols can provide information-theoretical security given that the underlying scheme is information-theoretically secure. Another advantage of MPC based on secret sharing is that there is no need for any encryption/decryption key. However, secret-sharing-based MPC protocols require significant amount of communication among the participating parties. In fact, privacy is achieved by distributing the computations among the parties.

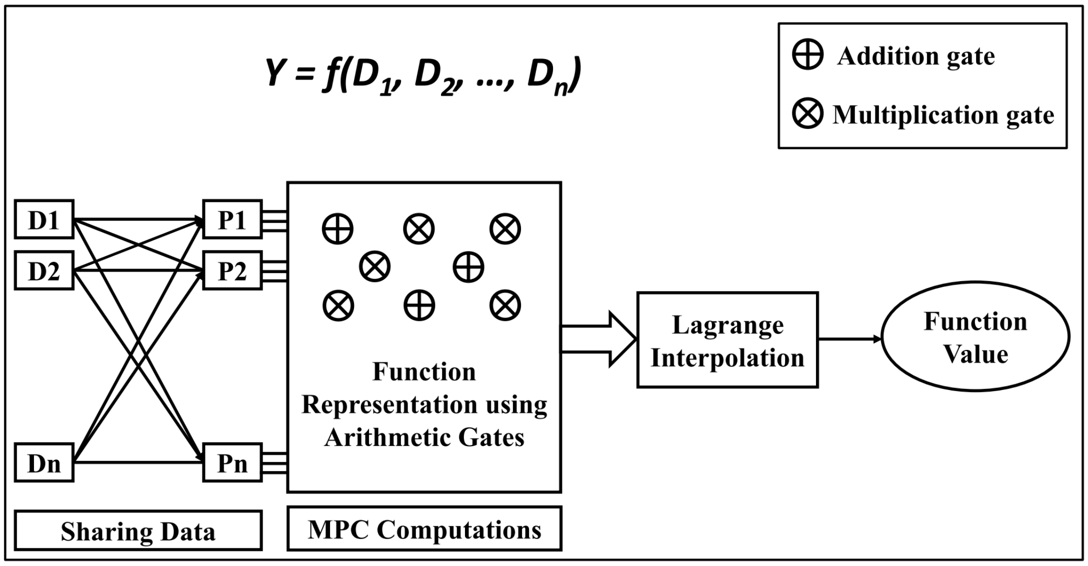

Figure 3.1: Secure Multiparty Computation

3.2.2 Secure Computation based on (Fully) Homomorphic Encryption

Another commonly-used approach in secure multiparty computation protocols is homomorphic encryption. In this case, the parties utilize a homomorphic encryption scheme, e.g., the Paillier scheme [131], to encrypt their data. The parties then perform computations on the encrypted form of data. Homomorphic encryption has attracted significant attention in the last decade. In particular, by the introduction of fully homomorphic encryption (FHE) schemes [66], this research area has shown to be more promising.

Most of the cryptographic schemes are based on the difficulty of some hard computational problems, e.g., integer factorization, discrete logarithm, the learning with errors problem (LWE) [142]. While lattice-based cryptography algorithms are considered to be resistant against quantum computers, not all cryptographic algorithms based on hard computational problems can resist against attacks by quantum computers. This is due to the fact that if the underlying difficult problem is solved (e.g., by utilizing quantum computers), the encryption scheme would not be secure anymore. In addition, homomorphic-encryption-based MPC protocols are computationally intensive and supporting multi-key encryption is a challenging task in such schemes [4].

3.2.3 Secure Computation based on Yao's Garbled Circuits

Yao's garbled circuits [170] is another dominant approach for secure two-party computation. A garbled circuit is an encrypted form of a function, which is supposed to be evaluated securely between two parties. More precisely, in this approach, one party encrypts the bits of their input and the intermediate state of the computation. This party then converts the computation into a circuit of binary gates, each represented as a garbled truth table. The other party, a.k.a., the evaluator, receives the circuit and the encrypted input bits. The evaluator then produces the encrypted output by evaluating each gate at the encrypted bits of the input and combining the results.

Yao's garbled circuit approach is the most efficient method for securely evaluating boolean circuits [99]. This approach does not require any communication between the parties during the evaluation. However, the intermediate state in the garbled circuits is far larger than the input data. This makes garbled circuits impractical for processing large data. Moreover, the garbled circuit approach provides computational security.

3.3 BUILDING BLOCKS FOR SECURE COMPUTATION

3.3.1 The Secure Comparison Operation (Yao's Millionaires Problem)

Comparison of numbers is a fundamental task in almost any kind of computation. Doing such a task in a secure or privacy-preserving fashion is also a well-known problem. The problem was originally formalized as the Yao's Millionaire problem [168]. Informally speaking, the millionaires problem states how two millionaires can figure out who is richer, while they do not reveal any additional information about their wealth. Note that in some literature the Yao's Millionaires problem is also called the greater-than (GT) problem and secure comparison problem.

Various solutions have been proposed for the Millionaires problem. The solutions largely rely on cryptographic tools and techniques. These techniques include secret sharing, homomorphic encryption, garbled circuits, and geometrical computation approaches. The proposed solutions are based on different assumptions. For instance, the security model can be semi-honest or malicious. The inputs and the output can be encrypted or shared using a secret sharing scheme. Moreover, the protocols may accept the inputs as bit strings or integer numbers. In particular, most of the protocols were designed in the semi-honest security model. Fortunately, any such protocol can be extended to a protocol that will be resistant to malicious adversaries. Such an extension can be obtained by adding a bit commitment scheme or by applying the GMW compiler [68] to the protocol which is resitant against semi-honest adversaries [107].

In what follows, we provide a comprehensive list of solutions and protocols for the Millionaires problem. This list is provided in Table 3.1, which contains different columns. The first column shows the name of each secure comparison protocol or the names of its authors. This column also includes the reference to the paper in which the protocol was proposed. The second column indicates whether each protocol needs a third party (TP) or the type of the third party that has been used in the protocol. The third column shows the operations ($>, \geq, <, \leq$ *or* $=$) that each protocol supports. The type of inputs (integer or rational/real numbers) is shown in the forth column. Finally the round/communication and computation complexities are presented in fifth and sixth columns respectively. The symbols that are used in the round and computation complexities are explained in Table 3.2. Due to space constraint, we used some abbreviations in Table 3.1. These abbreviations are explained in Table 3.3.

Protocols and Solutions for the Secure Comparison Problem					
Protocol or Reference	**TP**	**Op**	**Type**	**Round**	**Computation**
[168]	No	$>, \leq$	Integer	2	exponential
[62]	Yes	$>, =, <$	Integer	$O(1)^*$	$O(1)^*$
[32]	Yes	$>, =, <$	Integer	3	-
[46]	No	$>, \leq$	Integer	-	$8k$
[45]	No	$>, \leq$	Integer	$6n^2 + 4n^2\log(N)$	$4n^2\log(N)$
[26]	No	$=$	Integer	-	$O(k)$
[63]	Yes**	$>, \leq$	Integer	2	$\lambda d\log(N) + 6d\lambda + 3d$
[85]	No	$\geq, <$	Integer	d	-
[18]	No	$>, <$	Integer	1	$(4n+1)\log N + 6n$
[148]	No	$>, \leq$	Integer	$O(m)$	$12m$
[104]	No	$>, \leq$	Integer	2	$5d\log(p) + 4d - 6$
[47]	No	$>, =, <$	Integer	19	$l \times \log(l)$
[95]	No	$>, \leq$	Integer	$O(1)$	$O(\log^2(n))$
[48]	No	$>, \leq$	Integer	2	$l(t + \log(l))$
Pro 1 [65]	No	$>, =, <$	Integer	$\log(m)$	$150m$
Pro 2 [65]	No	$>, =, <$	Integer	≤ 9	$124m$
[123, 124]	No	$>, =, <$	Integer	≤ 133	$O(l\log(l))$
[154]	No	$>$	Integer	8	$29l + 36\log(l)$
[51]	DTP	$>, \leq$	Integer	2	$O(l(t + \log(l)))$
[98]	No	$=$	Integer	1	-
[152]	No	$<, \geq$	Rational	1	$2n$
[19]	No	$>, \leq$	Integer	1	$7n\log(N)$
[97]	No	$>, \leq$	Integer	-	negligible
[143]	No	$>$	Integer	8	$58.5l + 33$
[33]	No	$<$	FiPN	$\log(k) + 2$	$3k - 4$
[33]	No	$=$	FiPN	4	$k + 4\log(k)$
Pro 2 [127]	TI	$>, <$	Integer	$O(\theta)$	$O(\theta\ n^2\ \log^2 n)$
Pro 3 [127]	TI	$>, <$	Integer	$O(\theta)$	$O(n^2\ \log_2\theta\ \log^2 n)$
Pro 1 [157]	No	$>, \leq$	Integer	$O(\log(l))$	$O((\kappa + \log\log(l))\log(l))$
Pro 2 [157]	No	$>, \leq$	Integer	$O(c)$	$O(\sqrt{l}(\kappa + \log(l)))$
with Pre-Comp [160]	No	$\geq, <$	Integer	l	$4(l-1)$
without Pre-Comp [160]	No	$\geq, <$	Integer	l	$7(l-1) + 1.5$
[173]	DTP	$\geq, <$	Integer	$O(1)$	$O(l)^{****}$
[158]	No	$\geq, <$	Integer	-	-***
[7]	No	$=, >$	FlPN	6	$4l + 5k + 4\log(k) + 13$
[72]	No	$>, \leq$	Integer	2	$5d\log(p) + 4d - 6$
2nd Pro [129]	TI	$>, \leq$	Integer	1	2

[111]	No	$>, \leq$	Integer	2	$2d\log(p)$
[54]	TI	$>, \leq$	Integer	$O(l)$	$O(l)$
Pro 2, 3 [108]	No	$>, =, <$	Integer	2	$6L + 4$
Pro 4 [108]	No	$>, =, <$	Rational	1	negligible
Pro 1 [107]	No	$>, =, <$	Integer	1	$2(s + 2)\log(N)$
Pro 2 [107]	No	$>, =, <$	Integer	$2n(n - 1)$	Negligible
Pro 2 [103]	No	$>, \geq$	Rational	1	10
Pro 3 [103]	No	$>, =, <$	Rational	1	8
[44]	No	$>, =$	Integer	$O(l)$	$O(l)$
[87]	DTP	$>, =, <$	Integer	-	$O(n)$
[3]	DTP	$>, =, <$	Integer	$O(1)$	$O(l)^{****}$

*: Note that the protocol of [62] is a very simple protocol that can be utilized for comparing $\{1, 2, 3\}$ in Z_7.

**: The third party used in [63], is an oblivious third party, meaning that it remains oblivious about the outcome of the protocol.

***: [158] provides two improvements for the DGK protocol.

****: Note that the computational complexity of the protocols of [3] and [173] in the online phase is $O(\sqrt{l/\log(l)})$.

Table 3.1: Secure Comparison Protocols (significantly extended based on [108])

Most Efficient and/or Competitive Secure Comparison Protocols

In what follows, we discuss some of the (most) efficient or competitive solutions and protocols that have been proposed for the secure comparison problem (greater-than (GT) problem). There are a few important points that should be highlighted. First of all, since the proposed solutions work in different settings and based on different assumptions and approaches, a thorough comparison of the solutions is out of the scope of this dissertation. However, we intend to determine a list of the protocols that have been reported amongst the efficient or most efficient solutions as well as those that might provide competitive results. Another important point that has to be kept in mind is that there are different criteria for measuring the performance of a comparison protocol. Usually different complexities such as computation, communi-

Table of Notations (used in Table 3.1)		
Ref.	**Notation**	**Meaning**
[46]	k	the bit length of the input
[26]	k	a security parameter used in the protocol
[18]	n	n: the input's length in bits
	N	N: size of the Paillier scheme's plaintext domain
[47]	l	the number of bits of integer inputs
[65], [148]	m	the number of bits of integer inputs
[123]	l	the length of the prime number p in bits
[45], [19]	n	n: the count of bits of the inputs
	N	N: encryption scheme's modulus
[154], [143]	l	the length of the prime number p in bits, i.e. $l = log(p)$
[152]	n	the number of elements of the set used in their protocol
[51]	l	l: the length of integer input in bits
	k	k: the length of RSA modulus in bits
	t	t: the security parameter
[33]	k	a parameter used in fixed point representation of numbers
[127]	θ	θ: is the auction's price range
	n	n: the count of bidders in the auction
[157]	l	l: the bit-length of each private input
	κ	κ: a correctness parameter
[7]	l	l: the length of inputs in bits
	k	k: a parameter used in the representation of real numbers
[54]	l	the bit length of the input integers
[107]	s	s: the encoding vector's dimension
	n	n: the count of parties in the protocol
	N	N: the Paillier scheme's modulus
[44]	l	the bit length of the inputs
[63], [85]	d	d: the input's length in bits
	λ	λ: a parameter for handling error
	N	N: the modulus
[104], [72], [111]	d	d: the length of inputs
	p	p: modulus of Elgamal encryption scheme
[108]	L	the length of $0-1$ encoding vector
[3], [173]	l	the input's length in bits

Table 3.2: Meaning of the Notations used in Table 3.1

Table of Abbreviations (used in Table 3.1)	
Abbr.	**Meaning**
TP	Whether the protocol uses third party or not
DTP	Distributed Third Party
TI	Trusted Initializer
Op	Comparison operations ($>, =, <, \geq$ or $\leq$)
Type	Integer, rational numbers, fixed point or floating point numbers
Pro	Protocol
FiPN	Fixed point notation (representation)
FlPN	Floating point notation (representation)

Table 3.3: Meaning of Abbreviations used in Table 3.1

cation and round complexity are used. As such, a certain secure comparison protocol might be efficient when some measurement criteria is considered, while the same protocol might not be efficient when another criteria is considered. We would like to highly emphasize that the provided list is not meant to be a comprehensive list of the ultimate efficient solutions. Nevertheless, we hope it can be useful for future research particularly for selecting secure comparison protocols that might be appropriate for different applications and/or settings.

According to [54], the protocol proposed in [51] is among the most efficient solutions for the GT problem. This protocol is due to Damgard, Geisler and Kroigard and has been referenced in other papers, e.g., [44, 158]. It is sometimes referred to as the DGK protocol. Some other efficient protocols are specified in [44], which is a very recent paper at the time of writing this survey paper. According to [44], the protocols of [98], which were proposed for secure equality and comparison tests, are among the most efficient protocols (with regard to communication complexity). The proposed protocols in [98] were later improved in [97] and [174]. It is important to know that the solutions in [98] rely on Yao's garbled circuits. We should emphasize that the main focus of [98] and [174] is improving garbled circuits with the goal of using them in secure function evaluation. In [98], secure integer equality testing as well as integer addition were mentioned as examples of secure function evaluation.

The most efficient protocols in the category of homomorphic encryption are: [48], [49] which is a correction to [48], and [158] which is an improvement to [48]. In Two-Party computation (2PC), the protocol proposed in [65] is the most effective solution with regard to communication and computation overhead. For secure equality testing, an efficient protocol with constant number of modular exponentiations was proposed in [26], which is an improvement of the protocol proposed in [86]. The computational overhead of the solution in [86] is $\Theta(k)$ modular exponentiations, wherein k denotes a security parameter used in the protocol.

Authors in [24] claimed that there are at least two efficient methods for doing secure comparison. One approach is utilizing homomorphic encryption schemes specialized for this purpose [48,49,61,160]. The second way is using garbled circuits [16]. Techniques based on the first approach are more efficient for comparison of encrypted values. Whereas those based on the second approach are more efficient for comparing unencrypted values [24]. For comparing unencrypted values in [24], the authors used a scheme of [16], a short circuit developed in [96], and a known oblivious transfer (OT) scheme introduced in [122]. For comparison of encrypted inputs, they used a modified version of Veugen's protocol [160].

Another efficient protocol for secure comparison of integers was proposed in [160]. According to [160], most efficient solutions for secure comparison rely on homomorphic encryption (HE) or garbled circuits (GC). For instance, a very efficient solution based on garbled circuits was presented in [97]. Another well-known HE-based protocol was proposed in [18] and later was improved by Damgard et. al. in [49, 51]. For the improvement, they used a dedicated encryption scheme designed for small plain text values. The solution presented in [95] has also been reported as an efficient solution for the Millionaire problem.

The protocols proposed in [127] and [129] can be among the most efficient or

Most Efficient and/or Competitive Secure Comparison Protocols					
Protocol or Reference	**TP**	**Op**	**Type**	**Round**	**Computation**
[95]	No	$>, \leq$	Integer	$O(1)$	$O(\log^2(n))$
[48]	No	$>, \leq$	Integer	2	$l(t + \log(l))$
Pro 1 [65]	No	$>, =, <$	Integer	$\log(m)$	$150m$
Pro 2 [65]	No	$>, =, <$	Integer	≤ 9	$124m$
[51]	DTP	$>, \leq$	Integer	2	$O(l(t + \log(l)))$
[98]	No	$=$	Integer	1	-
[97]	No	$>, \leq$	Integer	-	negligible
Pro 3 [127]	TI	$>, <$	Integer	$O(\theta)$	$O(n^2 \log_2\theta \log^2 n)$
with Pre-Comp [160]	No	$\geq, <$	Integer	l	$4(l - 1)$
without Pre-Comp [160]	No	$\geq, <$	Integer	l	$7(l - 1) + 1.5$
[158]	No	$\geq, <$	Integer	-	-
Pro 2 [129]	TI	$>, \leq$	Integer	1	2

Table 3.4: Most Competitive Secure Comparison Protocols

competitive solutions for the settings and applications that have been considered in the corresponding references [127] and [129]. In particular, the second protocol of [129], which is based on verifiable secret sharing [42], has very small and constant complexities (please see the round, communication and computation complexities provided in Table 2 in [129]). Moreover, the third protocol of [127], which is verifiable and provides unconditional security, can be very efficient depending on the values of θ and n (where θ represents the range for the auction's price and n is the number of bidders). Overall, Table 3.4 illustrates a list of efficient secure comparison protocols that can provide competitive results for different settings that each protocol has considered.

To recap, numerous solutions (protocols) have been proposed for the secure comparison problem. The proposed solutions are based on different assumptions and approaches. As authors have mentioned in [159], no comparison protocol can perform the best with regard to all the three efficiency measurement criteria. Recall that the measures used for performance analysis of comparison protocols include round complexity, communication complexity and computation complexity. It is worth pointing out that the methods in the table might be under different security levels or model assumptions. This is because different protocols have used different parameters, e.g., the modulus of the underlying homomorphic encryption, the security and error parameters. They may also work in different settings or use different building blocks. Here we just provide a reference of protocols with competitive running times. The provided list is not meant to be a comprehensive list of the ultimate efficient solutions for the secure comparison problem. However, it can pave the way for future research, particularly for helping the researchers to consider some potentially efficient solutions.

3.3.2 Other Building Blocks for Secure Computation

For performing secure computation, the four main arithmetic operations need to be implemented in a secure fashion. These operations can be implemented securely using secret sharing schemes, homomorphic encryption techniques and Yao's garbled circuits. For instance, the Paillier homomorphic encryption scheme [131] allows us to calculate the addition of two encrypted values by multiplying their corresponding ciphertexts and without decrypting them. In the case of secret sharing schemes, two or more parties can calculate the addition of their secret values by adding the shares of the secret values locally and then conducting a Lagrange interpolation on their updated shares.

Depending on the application domain, the secure implementation of other operations may also be needed. For instance, in privacy-preserving data mining and machine learning, the secure version of three operations is needed. These operations include secure comparison, secure inner product of two vectors, and secure argmax [25]. In some cases, the secure version of natural logarithm, i.e. the *ln()* function, the sign function, the sigmoid function is also required [35].

3.4 A LITERATURE REVIEW ON SECURE MPC TOOLS AND LIBRARIES

In this section, we compare the state-of-the-art secure multiparty computation tools (including libraries, implementations and frameworks). These tools can be used for secure computation in the IoT and big data domains [10, 153]. We would like to emphasize that this part of the dissertation is based on the student's research during his PhD program which was already published in [64]. More importantly, there might be numerous other relevant tools and libraries for secure computation. Due to lack of space and page limitations and a short time interval that was given to this part of the research, we studied and included only some of the previous works. For more and some recent MPC-related frameworks and libraries, interested readers may refer to [30], [79] and [93], among other references.

Fairplay [112] is a secure function evaluation (SFE) tool that allows two parties to perform a joint computation without any trusted third party. This tool is based on Yao's garbled circuits and provides a high-level function description language called SFDL. The Fiarplay compiler compiles SFDL programs into a boolean circuit and evaluates the circuit using its runtime environment.

FairplayMP [17] is an extension of Fairplay [112] for multiple parties. This tool is based on Yao's garbled circuits and secret sharing schemes. FairplayMP uses an emulated trusted third party. The emulated trusted third party receives the inputs from the parties, does the desired computations and privately informs the parties of their outputs.

Sharemind [23] is a secure multiparty computation framework consisting of three parties. It is one of the most developed and efficient MPC tools and supports 32-bit integer arithmetic. However, it uses a non-standard secret sharing technique and does not extend to more than three parties [178].

VIFF [50] is a compiler for secure multiparty computation based on standard secret sharing schemes. It uses parallelization and multi-threading to provide faster computations. This framework supports computations consisting of basic primitives, e.g. addition and multiplication, on secret-shared values.

SEPIA [31] is a Java library based on linear secret sharing schemes. It separates the parties into computational parties and the parties who provide inputs and obtain outputs. SEPIA is used for secure distributed computation on network data, e.g., for privacy-preserving network intrusion detection.

TASTY [82] is a tool (with a compiler) for two-party secure computation (2PC) based on Yao's garbled circuits and homomorphic encryption. This tool can be used for describing, generating, executing, benchmarking and comparing secure 2PC protocols. It allows a user to provide a description of the computations to be performed and transforms the description into a 2PC protocol.

SPDZ [53] is a secure multiparty computation protocol based on secret sharing and homomorphic encryption. SPDZ consists of an offline (preprocessing) phase and an online phase. In the offline phase, the required shared random data is generated and in the online phase, the actual secure computation is carried out.

SCAPI [59] is an open-source library for developing MPC frameworks and secure computation implementations. It comes with two instantiations of the Yao's garbled circuits. One instantiation is secure against active adversaries and the other is secure against passive adversaries. SCAPI is implemented in Java and uses the JNI framework for calling native codes, to make the library efficient.

Wysteria [139] is a high-level programming language for writing MPC programs. It supports mixed-mode programs consisting of private computations with multiparty computations. Wysteria compiles the MPC programs to circuits and then executes the circuits by its underlying MPC engine.

Obliv-C [175] is a language for secure computation programming based on the garbled circuits. It is an extension of the C programming language that provides data-oblivious programming constructs. The Obliv-C compiler, implemented as a modified version of CIL, transforms Obliv-C codes to plain C codes.

Enigma [180] is a decentralized computation framework which combines MPC and Blockchain technology to provide guaranteed privacy. It allows different parties to jointly store and perform computations on their data without exposing the privacy of the data. Enigma also removes the need for trusted third parties.

Frigate [120] is a validated compiler and fast circuit interpreter for secure computation. It introduces a C-style language for secure function evaluation based on garbled circuits. Frigate has been developed with an emphasis on the principles of compiler design. It addresses the limitations of many previous MPC frameworks and produces correct and functioning circuits [120].

Chameleon [145] is a hybrid framework for privacy-preserving machine learning. This framework is based on the ABY framwork [55], which implements a combination of secret sharing, garbled circuits and the GMW protocol [69]. Chameleon has an offline and an online phase and most of the computation is performed in the offline phase. It uses a semi-honest third party (STP) in the offline phase, for generating the required correlated random values.

Wys* [140] is a domain-specific language (DSL) for writing mixed-mode secure MPC programs. It is based on the the idea of Wysteria [139] and embedded/hosted in F* programming language. For running a MPC program in Wys*, the program is first compiled using the F* compiler. Then each party runs the compiled codes using the Wys* interpreter. The result, which is a boolean circuit, is evaluated using the GMW protocol [69] on the parties' secret shares.

Conclave [162] is a query compiler that makes secure computation on big data efficient. Conclave generates codes for cleartext processing in Python and Spark and codes for secure computation using Sharemind [23] and Obliv-C [175]. The idea behind Conclave is to minimize the computations under MPC as much as possible.

Conclave can support only two or three parties and withstands a passive semi-honest adversary.

We summarized the reviewed secure MPC tools in Table 3.5. The table illustrates the main details and characteristics of the tools. Note that, due to the space constraints, we used some abbreviations in Table 3.5. The meaning of the abbreviations is provided in Table 3.6.

The first column of Table 3.5 shows the name of the MPC tools, the year in which each tool was developed, and the reference related to each tool. The second column determines the number of parties that each tool supports. The third column specifies the cryptographic primitives that have been used in the development of each tool. The fourth column defines the type of security, i.e. computational or information-theoretical, that each tool provides. The fifth column shows whether each tool uses some trusted third party (TTP) or such a party is simulated in the tool. The idea of doing secure multiparty computation without relying on any trusted third party is an interesting one. However, realizing such a computational model seems to be a challenging task; as the fifth column of Table 3.5 shows, the majority of the listed tools either need trusted third parties or simulate them. The last column of the table shows the programming languages that were used for the development of each tool.

We also provided a table that illustrates the adversarial model for each MPC tool; see Table 3.7. The table specifies the number of corrupted parties that each tool can tolerate. Note that in secure multiparty computation, the participating parties might be corrupted by some adversaries. The parties may also collude with each other. Therefore, it is important to consider such scenarios in the implementation. Finally, Table 3.8 shows some applications for each MPC tool.

Secure MPC Tools and Libraries					
Tool/Library	**Parties**	**Based on**	**Security**	**TTP**	**Prog. Lang.**
Fairplay 2004 [112]	2	GC	Computational	Yes	SFDL (Java)
FairplayMP 2008 [17]	≥ 3	GC and SS	Computational	Em. TTP	SFDL (Java)
Sharemind 2008 [23]	3	Additive SS	Info. Theoretic	Yes	SecreC (C++)
VIFF 2009 [50]	≥ 3	SS	Info. Theoretic	No	Python
VIFF 2009 [50]	2	Paillier HE scheme	Computational	No	Python
SEPIA 2009 [31]	≥ 3	Shamir's SS	Computational	Sim. TTP	Java
TASTY 2010 [82]	2	HE and GC	Computational	No	Python
SPDZ 2012 [53]	≥ 2	SS and HE	Computational	Yes	C++/Python
SCAPI 2012 [59]	≥ 2	GC	Computational	No	Java
Wysteria 2014 [139]	≥ 2	GMW protocol	Info. Theortic	Sim. TTP	OCaml
Obliv-C 2015 [175]	2	GC	Computational	No	C
Enigma 2015 [180]	≥ 2	VSS and Blockchain	Info. Theortic	No	WebAssembly
Frigate 2016 [120]	2	GC	Computational	No	C++
Chameleon 2018 [145]	2	SS, GMW, GC	Computational	STP	C++
WYS* 2019 [140]	≥ 2	[139]	Info. Theortic	Sim. TTP	F*
Conclave 2019 [162]	2 or 3	[23] and [175]	Computational	Yes	Python/Spark

Table 3.5: Secure MPC Tools and Libraries (based on [153] and [64])

Table of Abbreviations (used in Table 3.5)	
Notation	**Meaning**
HE	Homomorphic Encryption
GC	Yao's Garbled Circuits
GMW	the Goldreich, Micali, and Wigderson (GMW) protocol [69]
SFDL	Secure Function Definition Language
SS	Secret Sharing
VSS	Verifiable Secret Sharing
STP	Semi-honest Third Party
TTP	Trusted Third Party
Em. TTP	Emulated Trusted Third Party
Sim. TTP	Simulated Trusted Third Party

Table 3.6: Abbreviations used in Table 3.5 (based on [64])

Table of Adversarial Model for MPC Tools and Libraries	
Tool/Library	**Secure against**
Fairplay 2004 [112]	not mentioned
FairplayMP 2008 [17]	a collection of $\lfloor \frac{n}{2} \rfloor$ corrupt computation players, as long as they operate in a semi-honest way
Sharemind 2008 [23]	a passive adversary able to corrupt at most one party
VIFF 2009 [50]	not mentioned
SEPIA 2009 [31]	$t < \frac{m}{2}$ colluding privacy peers. Note that the systems has n input peers and m privacy peers
TASTY 2010 [82]	not mentioned
SPDZ 2012 [53]	an active adversary capable of corrupting up to $(n-1)$ parties
SCAPI 2012 [59]	both active and passive adversaries
Wysteria 2014 [139]	a semi-honest adversary capable of corrupting up to $(n-1)$ parties
Obliv-C 2015 [175]	semi-honest adversaries
Enigma 2015 [180]	not mentioned
Frigate 2016 [120]	semi-honest model
Chameleon 2018 [145]	semi-honest (honest-but-curious) model
WYS* 2019 [140]	semi-honest (honest-but-curious) model
Conclave 2019 [162]	a passive semi-honest adversary

Table 3.7: Table of Adversarial Model (based on [64])

Table of Applications for MPC Tools and Libraries	
Tool/Library	**Applications**
Fairplay 2004 [112]	secure two-party computation
FairplayMP 2008 [17]	secure multiparty computation
Sharemind 2008 [23]	tax fraud detection system
VIFF 2009 [50]	sugar beet auction, decision tree learning, privacy-preserving verifiable computation
SEPIA 2009 [31]	private information aggregation, network security and monitoring
TASTY 2010 [82]	set intersection, face recognition
SPDZ 2012 [53]	oblivious RAM schemes and oblivious data structures for MPC
SCAPI 2012 [59]	privacy-preserving impersenation detection systems and fair exchange protocols
Wysteria 2014 [139]	DStress (a framework for privacy-preserving and distributed graph analytics)
Obliv-C 2015 [175]	secure computation and data-oblivious computation
Enigma 2015 [180]	decentralized computation, IoT, crypto bank, blind e-voting, n-factor authentication
Chameleon 2018 [145]	privacy-preserving machine learning, e.g. SVM and deep learning
WYS* 2019 [140]	joint median, card dealing, private set intersection (PSI)
Conclave 2019 [162]	secure MPC on big data, e.g. credit card regulation and market concentration

Table 3.8: Table of Applications (based on [64])

CHAPTER 4
THE RGSW FHE SCHEME AND ITS C++ IMPLEMENTATION

In this section we describe the approximate eigenvector (a.k.a., GSW) fully homomorphic encryption (FHE) scheme [67] and its ring variant referred to as Ring-GSW or RGSW ([58] and [38]). Furthermore, we provide an efficient C++ implementation of the RGSW scheme ([58] and [38]).

4.1 INTRODUCTION

The GSW scheme [67] is a third generation fully homomorphic encryption scheme takes advantage of a nice property of matrices that there is a full homomorphism on the ring of matrices (i.e., $M_{(n\times n)}[Z_q]$). The security of the GSW scheme relies on the well-known learning with errors problem (LWE) [142], which has been conjectured to be a hard computational problem. The RGSW scheme ([58] and [38]), on the other hand, is constructed based on the ring of polynomials (i.e., $Z_Q[x]/(X^N+1)$, where N is a power of two such as 1024) and its security is based on the RLWE problem [109].

Our C++ implementation of the RGSW FHE scheme ([58] and [38]) works on the cyclotomic ring of polynomials and utilizes several algorithmic optimizations for faster computations of homomorphic operations. Particularly, we use the number theoretic transformation (NTT) [135] to speed up the multiplications of polynomials. Our implementation also allows to use different bases for reducing the dimensions of the RGSW ciphertexts. This is achieved by decomposing the coefficients of the polynomials in different bases (e.g., base 2 for bit decompisition and base 256 for byte decompisition) and constructing the gadget matrix (G) [117] accordingly. For polynomial arithmetic, we use the NFLlib [5], which is a fast C++ library specifically developed for lattice based cryptography and fast polynomial arithmetic.

4.1.1 Our Contribution

The contribution of this chapter of the dissertation is twofold. For one, we provide an efficient C++ implementation for the ring variant of the GSW scheme [67], commonly referred to as RGSW [58] and [38]. The RGSW scheme (see also [58] and [38]) is a fully homomorphic encryption scheme that supports both addition and multiplication of ciphertexts. Our implementation supports these functionalities and has the flexibility to work with different configurations. As the second contribution, we use our implementation of the RGSW scheme for homomorphic computation of pseudorandom functions (PRFs). Pseudorandom functions are of great interests in cryptography and secure computation. Particularly, we propose two constructions for homomorphic computation of PRFs based the learning with rounding (LWR) [15] and the ring variant of learning parity with noise (LPN) [21, 22] problems. Our constructions can be used for improving the existing secure multiparty computation protocols, e.g., the SPDZ secure MPC protocol and compiler [53].

We would like to emphasize that our implementation is based on the NFLlib C++ library [5], which is a very efficient library for lattice-based cryptography. We use this library only for polynomial arithmetic, particularly for fast polynomial multiplication using the number theoretic transformation (NTT) [135]. Furthermore, for our proposed constructions for homomorphic computation of pseudorandom functions (PRFs) we utilize a ciphertext embedding technique which is in spirit of the Nussbaumer transformation [130]. The embedding technique benefits from the properties of the ring of cyclotomic polynomials and incorporates the NTT representation [135] of the polynomials in a clever way. This enables us to pack polynomials of a smaller ring and lift them to a larger ring for faster multiplication in larger rings.

4.2 THE APPROXIMATE EIGENVECTOR (GSW) METHOD

The approximate eigenvector (a.k.a., GSW) scheme is a fully homomorphic encryption scheme which was introduced in [67]. This scheme is based on an observation that there exists some full homomorphism on the ring of matrices. However, since such a homomorphism cannot solely be useful for cryptographic purposes (because it is not secure), using this property of matrices along with some computational hard problem can potentially provide us a secure cryptographic scheme. Similar to most FHE schemes, the security of the GSW scheme relies on the learning with errors (LWE) problem [142]. The RGSW scheme [58] and [38] is the ring-variant of the GSW scheme [67]. More precisely, the GSW scheme [67] works on the ring Z_q where q is a positive integer (or on a finite field in case q is a prime number). On the other hand, the RGSW scheme ([58] and [38]) works on the ring of polynomials with integer coefficients in Z_q (i.e., $Z_Q[x]/(X^N+1)$, where N is a power of two such as 1024).

4.2.1 How the GSW Scheme Works

In the GSW FHE scheme [67], unlike other well-known FHE schemes, a message is encrypted by mapping it to a matrix over the ring of integers modulo q, i.e., Z_q. The GSW scheme has been studied in a few previous works, including in [9, 80, 119]. In what follows we briefly explain how this scheme works. Our explanation is based on [9, 67, 80, 119].

In the GSW scheme (as explained in [9]), the message space is the ring of integer numbers (i.e., $\mathbb{Z}$). The ciphertext space in this scheme is the set of square binary matrix of dimension $n \times l$ by $n \times l$ (i.e., $C \in \{0,1\}^{nl \times nl}$), where n is a parameter specifying the dimension of the scheme and $l = \lceil \log_2 q \rceil$ is the bit-length of the modulus q. The secret key generation, encryption and decryption algorithms of this scheme are explained in [9, 80, 119].

By having a secret key (which is a vector of dimension $1 \times n$ sampled from Z_q with the last element being 1), a message (which can be 0 or 1 or any integer number) is encrypted to a matrix of dimension $n \times l$ by $n \times l$ (as described in [9, 80, 119]).

The encryption process, in the GSW scheme [67], mostly involves straightforward linear algebra operations [9, 80, 119] and uses a block-diagonal matrix called the gadget matrix, introduced in [117] (see also [9, 80, 119]). Now by having the ciphertexts of different messages, the other important thing to understand is how the homomorphic oprations over ciphertexts (i.e., the addition of ciphertexts and multiplication of ciphretext) are defined. The definition of the addition operation in the GSW scheme [67] is straightforward, the ciphertexts are simply added (which in fact is the component-wise addition of the matrices). The multiplication operation of this scheme is a little more involved. As illustrated in [9, 80, 119], given two ciphertexts C_1 and C_2, their multiplication in the GSW scheme [67] is defined as:

$$C^{\times} := C_1.G^{-1}(C_2) \tag{4.1}$$

where, according to [119], G^{-1} is an operation (algorithm) that returns the binary decomposition of its input. Note that G^{-1} can operate on vector or matrix inputs over Z_q [9]. When the input is a matrix, G^{-1} is applied to every column of the matrix independently [119]. The references [9, 80, 119] have provided the details of the multiplication operation of the GSW scheme [67].

The decryption process of the GSW scheme [67] again requires straightforward linear algebra operations. While the decryption process explained in [119] involves operations on matrices, the approach provided for a simplified version of the GSW scheme [67], described in [9], needs the dot-product of two vectors.

4.3 THE RING-GSW (RGSW) FHE SCHEME

The Ring-GSW (or RGSW for short) is a variation of the approximate eigenvector fully homomorphic encrytpion scheme [67] that work on the ring of cyclomotic polynomials [58] and [38]. More accurately, the RGSW FHE scheme is a mapping from the ring of cyclotomic polynomials to the ring of matrices over the ring of cyclotomic polynomials. In this section we describe how the RGSW scheme works [58] and [38]. Furthremore, we provide the details of our C++ implementation for this scheme. We also discuss the applications of our implementation for homomorphic evaluation of functions, e.g., homomorphic evaluation of pseudorandom functions (PRFs).

4.3.1 The Plaintext and Ciphertext spaces for the RGSW Scheme

Similar to other RLWE-based FHE schemes, the plaintext space for the RGSW scheme ([58] and [38]) is the ring of cyclotomic polynomials, i.e., $R_q = Z_q[x]/(X^N+1)$, where N is usually a power of two such as 1024. Simply put, in the RGSW scheme the plaintext can be a polynomial with integer coefficients in Z_q and of degree degree at most $N-1$. A ciphertext is a matrix of polynomials with dimensions $m \times l$, where $l = 2 \times K$ and $K := \lceil \log_2(q)/\log_2(B) \rceil$. Moreover, B is base, which can be 2 or other powers of two, e.g., 2^7, 2^9 etc.

4.3.2 The Public and Private Keys for the RGSW Scheme

The secret key for the RGSW scheme is a vector of polynomials defined as follows:

$$sk = (s(x), 1) \in R_q^2 \tag{4.2}$$

where $s(x)$ is a polynomial in R_q with random coefficients and the second component of the secret key is 1. Given a secret key as $sk = (s(x), 1)$, the public key is generated as follows:

$$A := \begin{pmatrix} A' \\ e - s(x) \times A' \end{pmatrix} \in R_q^{2\times 2K} \tag{4.3}$$

where e is the error (noise) vector which contains $2K$ polynomials each with small coefficients in Z_q, i.e., $e = (e_1(x), e_2(x), \dots, e_{2K}(x)) \in R_q^{2K}$. Similarly, A' is a vector of random polynomials in R_q^{2K} with arbitrary coefficients in Z_q, i.e., $A' = (a_1(x), a_2(x), ..., a_{2K}(x)) \in R_q^{2K}$. Recall that R_q is the ring of cyclotomic polynomials $R_q = Z[x]/(x^N + 1)$.

4.3.3 The Encryption and Decryption Algorithms for the RGSW Scheme

The encryption and decryption algorithms for the RGSW scheme [58] and [38] are similar to the encryption and decryption algorithms of the GSW scheme [67], but instead of working on the numbers in Z_q, the encryption and decryption algorithms of the RGSW scheme [58] and [38] work on polynomials in R_q. Given a message $m \in R_q$, the message is encrypted as follows:

$$\begin{aligned} RGSW.Enc\colon R_q &\longrightarrow M_{2\times 2K}(R_q) \\ m &\longmapsto A \times R + m \times G \end{aligned} \tag{4.4}$$

where $A \in R_q^{2\times 2K}$ is the public key, $R \in R_q^{2\times 2K}$ is a matrix of random polynomials with small coefficients and G is the gadget matrix, which was introduced in [117]. The gadget matrix is defined as follows:

$$G := \begin{bmatrix} 1 & B & B^2 & \dots & B^{K-1} & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 & B & B^2 & \dots & B^{K-1} \end{bmatrix} \in Z_q^{2\times 2K} \tag{4.5}$$

where B is the base and can be two or a power of two such as 2^7, 2^9, or 2^{10}.

The decryption algorithm for the RGSW [58] and [38] works as follows. Given a ciphertext $CT \in R_q^{2\times 2K}$, we need to find the closest vector to $sk \times CT$. The result of $sk \times CT \in R_q^{2K}$ is a vector of polynomials, where its two last components can be used for decrypting the ciphertext. This is because how the message is encoded using the gadget matrix, which is used as part of the encryption process.

4.3.4 The Homomorphic Operartions for the RGSW scheme

The RGSW scheme [58] and [38] supports homomorphic operations on ciphertexts, i.e., addition and multiplication of ciphertexts. Given two ciphertexts C_1 and C_2 (which are assumed to be encryptions of m_1 and m_2 respectively), the addition of ciphertexts is defined as follows:

$$\begin{aligned} \oplus\colon & M_{2\times 2K}(R_q) \times M_{2\times 2K}(R_q) \longrightarrow M_{2\times 2K}(R_q) \\ & (C_1, C_2) \longmapsto C_1 + C_2 \end{aligned} \tag{4.6}$$

The multiplication of two ciphertexts is a little more involved. Given two ciphertexts C_1 and C_2, the multiplication of the two ciphertexts is defined as follows:

$$\begin{aligned} \otimes\colon & M_{2\times 2K}(R_q) \times M_{2\times 2K}(R_q) \longrightarrow M_{2\times 2K}(R_q) \\ & (C_1, C_2) \longmapsto C_1 \times G^{-1}(C_2) \end{aligned} \tag{4.7}$$

where G^{-1} is the inverse operation of the gadget matrix, introduced in [117] (for more details regarding the gadget matrix and its inverse operation, see also [9,80,119]). Simply put, the gadget matrix G is defined as $G = g^{\intercal} \otimes I_n \in R_q^{n\times nl}$ [119]. Here, g which is called the gadget vector is defined as $g = (1, B, B^2, ..., B^{l-1})$. Recall that B is the base which can be two or other powers of two, e.g., 2^7, 2^9 etc.

4.4 AN IMPLEMENTATION OF THE RGSW SCHEME IN C++

We provide a C++ implementation of the RGSW scheme that we described in the previous sections ([58] and [38]). Our C++ implementation uses the C++ standard template library (STL) and some other well-known C++ libraries which have been developed specifically for mathematical and number theoretic computations (including the GMP library[1] and the NFLlib library [5]). For testing purposes and verification of computations, we sometimes used the NTL C++ library[2]. To speed up our imple-

[1]https://gmplib.org
[2]https://libntl.org

mentation we applied two main optimizations. The first one deals with the dimension of the RGSW ciphertexts. A ciphertext in the RGSW scheme [58] and [38] is a matrix of polynomials. In general, RGSW ciphertexts are constructed as matrices of dimension $m \times n$ where m is a positive integer; $n = 2l$ and l denotes the bit-length of the polynomial's coefficients. For each ciphertext, we considered two rows, i.e., $m = 2$, and for the number of columns in the matrix, i.e., n, our implementation can work with different values. The second optimization deals with the polynomials multiplication. Polynomial multiplication is a common arithmetic operation particularly in fully homomorphic encryption schemes based on the RLWE problem [109]. There are several well-known techniques for fast polynomial multiplication. For polynomial multiplications in general the fast Fourier transform (FFT) enables us to perform polynomial multiplication in $O(N \log(N))$, where N represents the polynomial's degree. Whereas the complexity of the naive approach for polynomial multiplication is $O(N^2)$. Polynomials that are usually utilized in homomorphic encryption schemes, however, are on finite fields or on rings (i.e., on Z_q, where q is a very large prime number). For polynomial multiplication in these domains the number theoretic transform (NTT) [135] can be used, which in fact is considered as the equivalent of FFT over finite fields.

4.4.1 The Best Working Parameter Sets for the RGSW Scheme

Cryptographic schemes usually impose some requirements on the domain and parameters that they work on. The considered requirements are usually considered for the scheme to be secure or the requirements are considered for efficiency reasons. For example, for many cryptographic algorithms and schemes that work on the ring of integers mod q, i.e., $\mathbb{Z}_q$, the modulus q needs to be a prime number. Likewise, for cryptographic schemes that work on the ring of cyclotomic polynomials, i.e., $R_q = Z_q[x]/(x^N + 1)$, usually some conditions are considered. Particularly, the degree of the polynomial, i.e., N is usually considered a power of two, e.g., 2^{10}. In order to be able to use the fast polynomial multiplication based on the number theoretic transformation (NTT), the degree N and the modulus q must satisfy $q \equiv 1 \pmod{2N}$ [5].

Our C++ implementation of the RGSW scheme works on the ring of cyclotomic polynomials, i.e., $R_q = Z_q[x]/(x^N + 1)$, and supports different configurations and sets of parameters. The NFLlib C++ library [5] that our implementation is built on supports different module q and degrees for the cyclotomic polynomial, i.e., N. Particularly, it supports powers of two including 256, 512 and 1024. It supports integers of different bit-length, including 16 bits, 32 bits, and 64 bits. Similarly, our RGSW implementation supports different degrees, including $N = 256$, $N = 512$ and $N = 1024$ and different bit lengths. Moreover, it supports different dimensions for the ciphertext based on different bases, i.e., B, that can be set in the implementation. The recommended parameters for which our implementation can work very efficiently are as follows:

- 32-bit integers: $q = 1073479681$, base $B = 2^7$ or $B = 2^9$.
- 64-bit integers: $q = 4611686018326724609$, base $B = 2^{20}$ or $B = 2^{30}$.

The degree of the cyclotomic polynomial, i.e., N, can be any power of two, particularly, $N = 256$, $N = 512$ and $N = 1024$. We would like to emphasize that the NFLlib library [5] supports different sets of parameters, including polynomials of degree up to $2^{20} = 1048576$ as well as 16-bit integers.

CHAPTER 5
HOMOMORPHIC COMPUTATION OF PSEUDORANDOM FUNCTIONS

5.1 INTRODUCTION

Secure multiparty computation (MPC), and particularly fully homomorphic encryption, have attracted extreme amount of attention during the last couple of decades. There have been several significant achievements in this area of research. Envisioned by the pioneers of cryptography in [146] as an open problem and boomed by the introduction of the first plausible blueprint [66], fully homomorphic encryption (FHE) is now almost standardized [6] and has been realized in several attempts [161]. Among many other important works, the learning parity with noise LPN problem[1] [21, 102, 134], the learning with errors problem LWE [142] and its ring variants, i.e., ring-LWE (RLWE) [109], seem to have played a significant role in this developing branch of modern cryptography. Specifically, the security of the majority of the promising FHE schemes are based on the LWE and RLWE problems.

The theory and applications of fully homomorphic encryption schemes in various domians, e.g., privacy-preserving machine learning [136, 164], have been significantly investigated in the last decade. More recently, researchrs have focused on the application of FHE schemes for homomorphic evaluation of pseudorandom functions (PRFs) and pseudorandom correlation generators (PCGs), see e.g., [2, 27, 29]. Pseudorandom functions, e.g., pseudorandom number generators and pseudorandom correlation generators, are very important building blocks in cryptography and cryptographic ap-

[1]https://en.wikipedia.org/wiki/Parity_learning

plications, particularly in secure multiparty computation (MPC). For instance, the SPDZ secure multiparty computation protocol and compiler [52, 53] heavily relies on correlated random triples [2, 27, 29]. An interesting direction in secure multiparty computation is how to design pseudorandom number generators or psudorandom correlation generators that benefit from the underlying hard problems in lattice cryptography. Motivated by this important application of homomorphic encryption and its connection to secure multiparty computation, in this chapter of the dissertation we focus on homomorphic computation of pseudorandom functions (PRFs).

The rest of the article is organized as follows. In section 5.2 we provide a brief review of the theoretical preliminaries, i.e., fully homomorphic encryption (FHE) and pseudorandom functions (PRFs). We also go over other necessary cryptographic building blocks such as BlindRotate and SampleExtract operations as well as LWE-to-RLWE and RLWE-to-LWE conversions. In section 5.3, we provide an interesting ciphertext embedding technique for performing homomorphic computations more efficiently. Our proposed constructions for homomorphic computation of pseudorandom functions are presented in 5.4. In sections 5.5 and 5.6 we analyze the security of our constructions and provide our experimental results. Lastly, concluding remarks are presented in section 5.7.

5.1.1 Related Works

There are some recent works that have focused on different class of pseudorandom functions (PRFs) and pseudorandom correlation generators PCGs. Recently the researchers in [2] have proposed a PCG construction that can generate correlated random triples for the SPDZ protocol [52, 53]. The proposed PCG construction [2] is based on the ring variant of the LPN problem; and can be used for multiparty setting [2]. In [29] the authors have proposed some efficient PCG constructions based on the ring-LPN [83] problem. Prior to that, a fast approach for generating pseudorandom instances of vector oblivious linear function evaluation, with application to oblivious function evaluation, was proposed in [27]. Yet, in [28] the researchers have studied PCGs more concretely and provided several constructions with applications in secure multiparty computation.

5.1.2 Our Contributions

In this article we focus on another class of functions for generating pseudorandom values, i.e., weak pseudorandom functions (weak PRFs); and provide constructions for homomorphic computation of such functions. Particularly, we consider weak pseudorandom functions based on the learning with rounding problem (LWR) [15] and the ring variant of the learning parity with noise problem (ring-LPN) [83].

We then utilize three well-known instantiations of fully homomorphic encryption schemes, i.e., LWE, RLWE and RGSW schemes, along with a nice ciphertext embedding technique to implement our homomorphic computations of PRFs. The ciphertext embedding technique that we use benefits from the well-known polynomial transformations, i.e., the number theoretic transformation (NTT) [135] and the Nussbaumer transformation [130]; and allows us to pack and lift the ciphertexts from a small domain to ciphertexts in a larger domain. This enables perfomring more efficient homomprhic computations on the ciphertexts. Our experimental results show that pseudorandom functions can be evaluated efficiently using FHE schemes (in hundreds of milliseconds and with reasonable security, i.e., 128 or 256 bits based on the homomorphic encryption standard [6]).

5.2 PRELIMINARIES

In this section we provide the preliminaries that we need throughout this article. These include the definition of fully homomorphic encryption (FHE) schemes and pseudorandom functions (PRFs). We follow the notations of the definitions in [101].

In this article, we use the following notations. For a positive integer q, $\mathbb{Z}_q$ denotes the set of integers modulo q. The ring of cyclotomic polynomials with coefficients in $\mathbb{Z}_q$ and of degree less than N is represented by $R_q = \mathbb{Z}_q[x]/(x^N + 1)$. Furthermore, we use a bigger ring of cyclotomic polynomials which we define and represent as $R'_q = \mathbb{Z}_q[x]/(x^{N^2} + 1)$.

5.2.1 Fully Homomorphic Encryption

We first provide the definition of a fully homomorphic encryption and then describe the FHE instantiations that we need for our constructions.

Fully Homomorphic Encryption [101] Given a security parameter κ, a (fully) homomorphic encryption technique $\mathsf{FHE} = (\mathsf{FHE.KeyGen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval})$ is a tuple of polynomial-time algorithms such that:

- $\mathsf{FHE.KeyGen}(1^\kappa)$ generates a secret key (sk), a public key (pk), and an evaluation key (evk).
- $\mathsf{FHE.Enc}(m, \mathsf{sk})$ generates a ciphertext ct which is an encryption of the message m with the secret key sk.
- $\mathsf{FHE.Dec}(\mathsf{ct}, \mathsf{sk})$ decrypts the given ciphertext ct using the decryption key sk.
- $\mathsf{FHE.Eval}(\{\mathsf{ct}_i\}, f, \mathsf{evk})$ given a set of ciphertexts ct_i and a function f as inputs, $\mathsf{FHE.Eval}$ evaluates the function on the ciphertexts using the evaluation key (evk) and outputs the computed ciphertext corresponding to f, i.e., ct_f.

Throughout this paper we deal with three instantiations of homomorphic encryption schemes, namely LWE [142], RLWE [109], and RGSW [38, 58] schemes.

An LWE homomorphic encryption scheme [142] is a mapping from $\mathbb{Z}_q^n$ to $\mathbb{Z}_q$. Moreover, an LWE instance is represented by a vector in $\mathbb{Z}_q^{n+1}$, i.e., $(\vec{\boldsymbol{a}}, b)$, where $\vec{\boldsymbol{a}} \in Z_q^n$ and $b = \langle \vec{\boldsymbol{a}} . \vec{\boldsymbol{s}} \rangle + m\frac{q}{p} + e \in Z_q$. Here, $e \in \mathbb{Z}_q$ is the noise which is supposed to be a small value; and p is the modulus for plaintext space, where typically we need to have $p \ll q$. Moreover, $\vec{s} \in Z_q^n$ is the secret key. Similarly, an RLWE homomorphic encryption scheme [109] is a map from R_q to itself. An RLWE instance is a pair of polynomials in R_q, i.e., $(a(x), b(x)) \in R_q^2$ such that $b(x) = a(x).s(x) + \frac{q}{p}.m(x) + e(x)$. In this case, $a(x) \in R_q$ is a random polynomials, $s(x) \in R_2$ is the secret key, $m(x) \in R_p$ is the message, and $e(x) \in R_q$ is a polynomial with small coefficients for noise. Moreover, $R_q := \mathbb{Z}_q[x]/(x^N + 1)$ is the cyclotomic ring of polynomials, where N is a positive power of two such as 1024.

The last category of FHE instantiation that we need is RGSW instances [38, 58]. An RGSW instance is represented by a matrix over the ring of polynomials. More specifically, an RGSW instance can be denoted by $A.R + m.G \in M_{m\times n}(R_q)$, where $A \in M_{k\times n}(R_q)$ is the public key, $R \in M_{k\times n}(R_q)$ is a matrix of polynomials with random small coefficients, and G is the gadget matrix, which was introduced in [117]. Furthermore, an RGSW fully homomorphic encryption scheme [38, 58] is a map from R_q to $M_{k\times n}(R_q)$, where $M_{k\times n}(R_q)$ is the ring of m by n matrices over the ring of cyclotomic polynomials, i.e., $R_q := \mathbb{Z}_q[x]/(x^N + 1)$.

5.2.2 Pseudorandom Functions (PRFs)

Pseudorandom function [101] Given two finite sets A and B and a set of functions from A to B, i.e., $\mathcal{F} = \{F_i : A \longrightarrow B\}$, $\mathcal{F}$ is said to be a family of (t, Q, ϵ)-pseudorandom functions (PRF) if a sampled function from $\mathcal{F}$, i.e., $F \longleftarrow \mathcal{F}$, is (t, ϵ)-indistinguishable from a uniformly selected random function $R : A \longrightarrow B$ from $\mathcal{F}$ (given that up to Q adaptive queries are allowed).

In this paper we propose techniques for homomorphic computation of two different instantiations of weak pseudorandom functions (PRFs), namely PRFs based on the learning with rounding LWR problem [15] and PRFs based on the ring variant of the learning parity with noise LPN problem [83]. It should be noted that the LWR-based PRF is a weak pseudorandom function and the LPN-based PRF is considered a bounded-query-weak pseudorandom function.

Weak PRF*s based on the* LWR *Problem:*

Our first construction for pseudorandom functions (PRFs) is based on the learning with rounding (LWR) assumption [15]. Given two vector of integers $\vec{\boldsymbol{s}} \in \mathbb{Z}_2^n$ and $\vec{\boldsymbol{a}} \in \mathbb{Z}_q^n$, an LWR-based PRF is instantiated as follows:

$$F(\vec{\boldsymbol{a}}, \vec{\boldsymbol{s}}) = \lfloor \langle \vec{\boldsymbol{a}} . \vec{\boldsymbol{s}} \rangle \rfloor_{q \longrightarrow p} \tag{5.1}$$

where p and q are positive integers and usually p must be much smaller than q, i.e., $p \ll q$.

Bounded-Query-Weak PRF*s based on the Ring-*LPN *problem:*

The second type of PRF instatioation that we use is based on the ring variant of learning parity with noise problem LPN [21], i.e., ring-LPN [83]. This instantiation work as follows: given a sparse polynomial $a(x) \in R'_q$ and $s(x) \in R'_2$, we define:

$$F(a, s) = a(x).s(x) \in R'_q \tag{5.2}$$

where $R' = \mathbb{Z}[x]/(x^{N^2} + 1)$ is the ring of cyclotomic polynomials. Moreover, by a sparse polynomial we mean a polynomial that the majority of its coefficients are zero.

5.2.3 FHEW/TFHE Abstraction

There have been significant progresses in developing efficient fully homomorphic encryption schemes. Particularly, the third generation of FHE schemes, which began with the introduction of the GSW scheme [67], was followed by a couple of other significant works, i.e., the FHEW [58] and TFHE [38] FHE schemes. Among several other important building blocks, the works [58] and [38] introduced several useful techniques including blind rotation and sample extraction techniques. As our constructions rely on these techniques, here we review these techniques very briefly [38, 39].

Blind Rotation [38, 39]:

Blind rotation is a technique that allows to homomorphically rotate the coefficient of a polynomial which was encrypted by an RLWE scheme. Given an RLWE ciphertext ct, an LWE ciphertext that encrypts μ, and the encryption of the bits of the secret key as RGSW ciphertexts, this algorithm basically computes another ciphertext ct' for which we have $\mathsf{ct}' = x^{-\mu'}.\mathsf{ct}$; where μ' is the noisy version of the plaintext μ stored in the given LWE ciphertext. Assuming ct is an RLWE encryption of a message $m(x) = \sum_{i=0}^{n-1} m_i x^i$, intuitively, the obtained ciphertext ct' is an RLWE encryption of a message $m'(x)$ whose constant term is $m_{\mu'}$. In other words, the polynomial $m(x)$ has been rotated μ' times [38, 39] to the left. Other coefficients of the message $m(x)$ are also rotated correspondingly.

Sample Extraction [38, 39]:

The goal of the sample extraction technique is to extract an LWE instance from a given RLWE instance. This technique simply chooses some coefficients of the polynomials of the RLWE instance and arranges them in a certain order to create the output LWE instance. This technique is a straightforward one and any coefficient of the polynomial which was encrypted in the RLWE ciphertext can be extracted efficiently [38, 39].

We would like to emphasize that in our proposed constructions for homomorphic computation of PRFs, we use these building blocks as black boxes and denote them as $\mathsf{BlindRot}$ and $\mathsf{Extract}$. More detailed description of these operations can be found in [38, 39] and other relevant references.

5.2.4 RLWE/LWE Conversion

As mentioned earlier, we utilize three different instatiations of FHE schemes, namely, LWE, RLWE, and RGSW. For performing homomorphic computations, it is sometimes necessary to convert the samples of one itstantiation to another. Particularly, two conversions have been well-studied, i.e. LWE-to-RLWE and RLWE-to-LWE conversions.

Converting an RLWE instance to an LWE instance, which is equivalent to extracting an LWE instance from an RLWE instance, is fairly straightforward and computationally efficient. For converting a given RLWE ciphertext to an LWE ciphertext it suffices to apply the sample extraction technique, i.e., Extract, on the given RLWE ciphertext [37].

The reverse operation, i.e., converting an LWE instance to an RLWE instance [37] and [34], is more involved and computationally intensive. In order to perform this conversion, one approach is to use an evaluation key, which has the encrytions of the secret key, and the extrenal product of RGSW and RLWE ciphertexts. Assuming we want the RLWE ciphertext to be in $R_q = \mathbb{Z}_q[x]/(x^N + 1)$, this approach requires N external products. When N is a large number, e.g., $N = 1024$, this naive approach is not efficient.

Recently, researchers in [34] have proposed an elegant approach that takes advantage of the properties of ring of polynomails and field extensions. The proposed approach [34] homomorphically evaluates the automorphisms of the Galois group over the tower of finite fields. As a result, it improves the conversion logarithmically and reduces the computational cost from $O(N)$ to $O(\log(N))$. We would like to remark that for LWE to RLWE conversion we have implemented an approach similar to [34] and used it in our constructions for homomorphic computation of PRFs.

5.3 POLYNOMIAL TRANSFORMATIONS AND CIPHERTEXT EMBEDDING

The advantage of RLWE-based homomorphic encryption schemes is that they are based on the ring of polynomials, which have really useful properties. There are several transformations that have shown to be very useful when working with the ring of polynomials. A well-known transformation is the number theoretic transformation (NTT) [135], which has been used in FHE implementations for fast polynomial multiplication. Another transformation that has been utilized in FHE-related research is the Nussbaumer transformation [130].

In this paper, we use an elegant ciphertext embedding technique which is in spirit of the Nussbaumer transformation [130]. The embedding technique uses a polynomial packing technique that allows us to embed a vector of N polynomials from R_q to a polynomial in R'_q. Recall that for a given N, we deal with two ring of cyclotomic polynomials, namely $R_q := \mathbb{Z}_q[x]/(x^N+1)$ and $R'_q := \mathbb{Z}_q[x]/(x^{N^2}+1)$. The ciphertext embedding technique works as follows and we use Enc-to-Enc' to refer this technique. For a vector of RLWE ciphertexts $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1, \ldots, \mathsf{ct}_{N-1}) \in (R_q^2)^N$, we define:

$$\mathsf{ct}' = (\mathsf{ct}'_0, \mathsf{ct}'_1) \in R'_q \times R'_q \tag{5.3}$$

where $\mathsf{ct}'_j = \sum_{i=0}^{N-1} \tau(\mathsf{ct}_{i_j})x^i$ for $j = 0$ and 1. Moreover, the τ function (which is in spirit of the Nussbaumer transformation [130]) is defined as follow:

$$\begin{aligned} \tau : &R_q \longrightarrow R'_q \\ &x \longmapsto x^N \end{aligned} \tag{5.4}$$

Intuitively, this embedding technique packs the polynomials of a vector of ciphertexts in a smaller ring and maps them to a ciphertext in a bigger ring, so that computations can be performed more efficiently. We use the notation Enc-to-Enc' to refer to this ciphertext embedding technique.

5.4 HOMOMORPHIC METHODS AND CONSTRUCTIONS

In this section we provide our proposed constructions for homomorphic computation of pseudorandom functions PRFs. Our constructions use two different instantiations of pseudorandom functions, namely LWR-based PRF and LPN-based PRF (as defined in section 5.2.2).

5.4.1 Homomorphic Computation of LWR-based Weak PRFs

Our first construction is a technique for homomorphic computation of weak PRFs based on the learning with rounding LWR problem [15], as defined in equation 5.1. The following procedure simulates the homomorphic evaluation of an LWR-based PRF:

Construction I (LWR-based weak PRF):

Input: $\{\mathsf{RGSW.Enc}(s_i)\}_{i=0}^{n-1}$, where $s_i \in \mathbb{Z}_2$, and $\vec{\boldsymbol{a}} \in \mathbb{Z}_q^n$.

Output: $\mathsf{LWE.Enc}(F(\vec{\boldsymbol{a}}, \vec{\boldsymbol{s}}))$

- $\mathsf{RLWE.ct} = \mathsf{FHE.BlindRot}(x^{\sum_{i=0}^{n-1} a_i . s_i})$
- Return $\mathsf{LWE.ct} := \mathsf{FHE.Extract}(\mathsf{RLWE.ct})$

We would like to highlight that the security of our proposed first construction is inherited from the underlying computationally hard problem, i.e., the learning with rounding problem LWR [15]. We analyze the security level of this construction in section 5.5.

5.4.2 Homomorphic Computation of LPN-based Bounded-Query-Weak PRFs

Our second construction is for homomorphic computation of bounded-query-weak pseudorandom functions (PRFs) that were instantiated based on the ring-LPN problem [83], as shown in equation 5.2. The procedure for homomorphic evaluation of

LPN-based bounded-query-weak PRFs is as follows:

Construction II (LPN-based bounded-query-weak PRF):

Input: $\{\mathsf{RGSW.Enc}(s_{\alpha_i})\}_{i=0}^{t-1}$, $\{\mathsf{RLWE.Enc}(x^{\alpha_i})\}_{i=0}^{t-1}$ and $a \in R'$.

Output: $\mathsf{LWE.Enc}'(s.a)$

- use the Enc-to-Enc′ mapping to convert $\{\mathsf{RGSW.Enc}(s_{\alpha_i})\}_{i=0}^{t-1}$ and $\{\mathsf{RLWE.Enc}(x^{\alpha_i})\}_{i=0}^{t-1}$ to $\mathsf{Enc}'(s)$.
- Return $\mathsf{RLWE.Enc}'(s).a$

The security of construction II relies on the hardness of the ring-LPN problem [83], which is a computationally hard problem. More accurate analysis regarding the security level of this construction is provided in the following section (section 5.5).

5.5 SECURITY ANALYSIS

In this section we discuss the security level of our constructions. We also analyze the boundaries on the amount of noise that can be used in the underlying FHE instantiations utilized in our constructions. Furthermore, we highlight the secret key distributions of the FHE instantiations and provide the recommended parameters set that our constructions work with.

First of all, for two FHE instantiations that we use, i.e., for LWE and RLWE, the amount of noise must be less than $\frac{q}{2\times p}$, where q is the modulus for the ciphertext domain and p is the modulus (or cardinality) of the plaintext space. For the decryption process of the RGSW instantiation to work properly, the amount of noise must be less that $\frac{q}{2\times B^{K-1}}$, where B is the base in the gadget matrix and K is the number of non-zero elements in a row of the gadget matrix. The boundary on the amount of the noise for the RGSW is different (from that of the other two FHE instatiations) because of the structure of the gadget matrix G. It should be mentioned that the same boundary on noise is applicable for the LWE and RLWE instantiations if instead of $\frac{q}{p}$ the value B^{K-1} is used as their rounding threshold.

Generally speaking, the distribution of secret key for the FHE instatioantions should be Gaussian, with a standard devitation of at least $\sigma = \sqrt{n}$, where n is the dimension of the lattice [118]. In our constructions we used random secret keys uniformly sampled from a binary domain, i.e., $\{0, 1\}$. It should be noted that a ternary domain, e.g., $\{-1, 0, 1\}$, can also be used with little modifications in the constructions.

The security level that our constructions provide are as follows. Our proposed constructions for homomorphic computation of PRFs are based on FHE instatioations that support different configurations. The dimension of the lattice, i.e., n, can be a power of two, e.g., 512, 1024 or 2048. In our implementation and for experimental results we set $n = 1024$. The degree of the cyclotomic polynomials, i.e., N, was also set to 1024 for the first construction and $1024^2 = 1048576$ for the second construction. The modulus Q can be a 30 bit prime number for 32-bit integer arithmetic; and a 62 bit prime number for 64-bit integer arithmetic. In our implementation and for evaluation results, we used the modulus Q which was chosen by the NFLlib C++ library [5] (such a mudulous allows fast polynomial multiplication thanks to the NTT transformation [135]). The base B in the gadget matrix, G [117], can be a positive power of two; and we used the values provided in Table 5.1 (also included in Table 5.2 and Table 5.3 in section 5.6). Our constructions work with the parameters set provided in Table 5.1.

n	N or N'	$\log_2 Q$	B
1024	1024 or 1048576	30	2^4 or 2^7
1024	1024 or 1048576	62	2^7, 2^{14}, 2^{20} or 2^{30}

Table 5.1: Parameters Set for Constructions I and II

5.6 EVALUATION AND EXPERIMENTAL RESULTS

We have implemented our constructions for homomorphic evaluation of pseudorandom functions (PRFs) in C++. Our implementation supports different configurations, including 32-bit and 64-bit integers and different polynomial degrees as long as the degree is a power of two. Particularly, our RGSW implementation supports different bases (B) in the gadget matrix G [117], that gives us the flexibility to choose different dimensions for RGSW ciphertexts. For polynomial arithmetic we used the NFLlib library [5], which is a fast C++ library for lattice-based cryptography and supports NTT-based polynomial multiplication [135].

Table 5.2 shows the runtime of our first construction along with different parameters that we used. There are a couple of points that should be highlighted. First, in Table 5.2 the measured runtimes are in milliseconds (ms) and averaged over multiple executions. Moreover, the numbers are rounded to the closet integer. Second, some configurations (e.g., 32-bit integers and base $B = 2^{10}$) are not supported in our implementation. In such cases we have put an 'NA' in the table. The reason that some configurations are not supported is because we use the number theoretic transformation (NTT) [135] for efficient polynomial multiplication and this imposes some limitations on the parameters that can be used (for instance, the degree N and modulus q must satisfy the requirement $q \equiv 1 \bmod 2N$). Moreover, the underlying NFLlib C++ library that we use for polynomial arithmetic, uses only 30 and 62 bits of 32-bit and 64-bit integers respectively (and sacrifices two or sometimes more bits for optimization).

l	$B = 2^4$	$B = 2^7$	$B = 2^{14}$	$B = 2^{20}$	$B = 2^{30}$
32-bit	360 ms	240 ms	NA	NA	NA
64-bit	2200 ms	550 ms	440 ms	280 ms	230 ms

Table 5.2: Runtime Results for Construction I ($N = 1024$)

The experimental results for our second construction are provided in Table 5.3. The runtime results in Table 5.3 are in seconds, where they have been averaged over multiple executions. Moreover, they were rounded to the closest decimal or integer numbers. Similar to construction 1, we have used the NFLlib C++ library [5] for fast polynomial multiplication. As it is seen in the table, with using larger bases (B) the runtime decreases. This is because if we uses larger bases, the dimension of RGSW ciphertexts reduces and thus less computations are needed. More interestingly, as it can be seen using the LPN-based PRFs along with the ciphertext embedding technique improves the runtime significantly. For instance, for 64-bit integer and with base $B = 2^{20}$ the runtime of the LPN-based PRF is 6.2 seconds. However, this approach encodes $N = 1024$ polynomials and this runtime corresponds to multiplication of and encrypted polynomial with a public polynomials both in $R'_q = \mathbb{Z}[x]/(x^{N^2} + 1)$. Therefore, the amortized runtime comparing to the LWR-based PRF is around 6 milliseconds.

l	$B = 2^4$	$B = 2^7$	$B = 2^{14}$	$B = 2^{20}$	$B = 2^{30}$
32-bit	7.8 s	5.5 s	NA	NA	NA
64-bit	NA	11.0 s	8.0 s	6.2 s	5.6 s

Table 5.3: Runtime Results for Construction II ($N' = 1024^2$)

5.7 CONCLUDING REMARKS

In this article we proposed two constructions for homomorphic computation of weak-pseudorandom functions (PRFs). The proposed constructions are based on the LWR and the ring variant of the LPN problems; and utilize three instatiations of fully hmomprphic encryption FHE schemes, i.e., LWE, RLWE, RGSW instatiations. We also used an interesting ciphertext embedding technique which is in spirit of the Nussbaumer transformation [130] and based on the number theoretic transformation

(NTT [135]).

We implemented our constructions in C++ and using the NFLlib C++ library that provides fast polynomial arithmetic operations thanks to the NTT transformation [135]. Our experimental results and security analysis shows that homomorphic computation of weak PRFs can be done in hundreds of milliseconds and can provide a security of 128 or 256 bits based on the homomorphic encryption standard [6]. We hope our constructions for homomorphic computation of PRFs will be utilized for secure evaluation of pseudrandom functions which are a common building blocks in secure multiparty computation protocols, e.g., the SPDZ secure multiparty computation protocol and compiler [52, 53].

CHAPTER 6
SECURE TRUST EVALUATION (STE) USING MULTIPATH AND REFERRAL CHAIN METHODS

In this chapter we provide several protocols for secure evaluation of trust values in a network. The proposed secure trust evaluation techniques are based on an interesting framework for modeling and measuring trust values. The framework is called information theoretic framework for modeling trust [156]. Moreover, our proposed secure protocols rely on secure computation based on the Shamir's secret sharing [150] approach. It should be mentioned that this part of the dissertation is based on the student's research during his PhD program; and the research was published in [137].

6.1 INTRODUCTION

Trust and reputation are common social concepts/mechanisms that have been used in different contexts, including in human interactions, economics, multiagent systems and computer networks. These social mechanisms are now well-studied and have been integrated into electronic applications/services, e.g., Amazon and eBay websites, in search engines, e.g. Google's PageRank algorithm, and in social networks. These social mechanisms can also be used in peer-to-peer (P2P) networks [1, 92, 163], collaborative environments, multiagent systems, autonomous vehicles, vehicle-to-vehicle (V2V) networks [176], and multiparty computation scenarios to provide better quality and more trustworthy services.

Trust and reputation are sometimes considered as soft security measures that compliment hard security measures, e.g. cryptography and secure multiparty computation protocols. It should be mentioned that using soft security measures alongside hard security measures can provide more secure and trustworthy systems and networks [138, 171]. In other words, integrating these social concepts into data and computation infrastructures can provide more reliable, secure and trustworthy systems and services [90]. In fact, trusted computing is a term that reffers to this idea and has been used in the IT security jargon [90] and [179]. However, there has been challenges with modeling and utilizing these social concepts, i.e. trust and reputation.

First, these concepts are highly subjective [81], in the sense that different people have different impressions about them. It should also be mentioned that these concepts are very contextual-based and time-dependent [81]. Fortunately, there has been significant attempts for modeling and measuring trust and reputation. In the computer science literature, Marsh [114] is among the first who tried to provide a computational model for trust. Thereafter, other models, methods and metrics have been defined for measuring trust and reputation quantitatively. In a nutshell, there are different theories/approaches for modeling and evaluating trust and reputation concepts. Some of the well-known approaches for measuring trust include subjective logic [88, 89], fuzzy logic [113], entropy-based models [156], Demster-Shafer theory [166]. Reputation is usually evaluated based on the trust values. Some of the well known reputation systems use simple summation, average and weighted average of the trust values [90]. Other reputation systems utilize the Beta probability density function and Bayesian networks [90].

Second, there are studies [144] discussing that users are usually unwilling to provide honest feedbacks (ratings) in trust and reputation systems, mainly due to the fear of retaliation for negative ratings. For a trust and reputation system (TRS) to be welcome by users, it is important that such a system keeps the users' data private while allowing them to perform the desired computations on their private data (here specifically the trust value (rating score) of users in one another). There are different approaches for providing such a system. One approach is to use decentralized reputation systems. Such reputation systems do not rely on any centralized reputation management authority, and thus are safer [73]. Another approach is to use secure computation approaches, e.g., secure MPC, in trust and reputation models.

6.1.1 Our Contribution

This paper aims at addressing data security and privacy issues in trust and reputation systems. We use secure multiparty computation (secure MPC), a.k.a., secure function evaluation (SFE), to provide secure trust evaluation (STE) methods. Our proposed methods [137] are based on the information theoretic framework for modeling trust and two approaches that trust propagate in a network [156]. These two approaches are: the idea of referral chains in social networks and multipath trust propagation in a network. We provide two protocols that enable the nodes in a network to securely evaluate their trust value in one another. As an application, we use our proposed STE method to provide a secure network routing protocol. Our protocols can be based on any secret sharing scheme, e.g., the Shaimr's (t, n)-threshold secret sharing scheme [150]. We would like to emphasize that our protocols do not rely on any trusted third parties. In other words, the nodes in a network can perform the required computations for measuring their trust securely and by themselves. Using secure trust evaluation methods will result in more secure and trustworthy network-based systems and services.

In section 7.2, we review the existing works related to secure trust and reputation models. In section 6.3, we provide the necessary preliminaries for our secure trust evaluation method. These include secure MPC based on secret sharing, and an encoding approach that allows performing secure computations on real-valued numbers. We use floating-point representation of real numbers to perform secure computations on such numbers. Note that in our model, trust values are real numbers in $[-1, 1]$

interval. In section 6.4, we provide our main contribution. We propose two secure protocols that are in fact the building blocks of our STE method. We also provide a secure network routing protocol as an appealing application of the proposed STE method. Our discussion is presented in section 6.5. The paper is rounded off by the conclusion in section 6.7.

6.2 RELATED WORKS

Data security and privacy are important issues in trust and reputation systems (TRS). Different approaches have been used to address such issues in the TRS systems. Among others, we can point out approaches based on secure MPC techniques and those based on decentralized computation frameworks. In what follows, we review the previous works related to secure trust and reputation models. For comprehensive surveys related to trust and reputation systems in general, interested readers may refer to [90] and [81].

In [166], two schemes for preserving the privacy of trust evidence providers were proposed. The proposed schemes use two non-colluding service parties, called Authorized Proxy (AP) and Evaluation Party (EP), to manage the aggregated evidences and process the collected data in encrypted format. The proposed schemes are based on public key cryptography, e.g. RSA and digital signature, and additive homomorphic encryption, e.g. Paillier scheme [131]. Centralized trust and reputation systems (TRS) can take advantage of their users' data. To address such an issue, the authors in [12] proposed a privacy-preserving distributed reputation mechanism based on the notion of *mailboxes*. Malicious-k-shares protocol, a decentralized privacy-preserving reputation system, was proposed in [78]. The protocol is based on the Paillier cryptosystem [131] and uses source managers (e.g., the Chord distributed hash table [155]) to share the data among k agents and perform privacy-preserving distributed computations.

The privacy-preserving version of the P2PRep [13] reputation-management mechanism, called 3PRep (privacy preserving P2PRep), was proposed in [125]. 3PRep enhances the P2PRep mechanism by adding two new protocols to preserve votes' privacy using semantically secure homomprphic encryption scheme, e.g. Paillier scheme [131]. Three different schemes for privacy-preserving computation of reputation values were presented in [73]. Two of the proposed schemes use a trusted third party to calculate the reputation. The third scheme does not rely on any trusted third party. Pavlov et al. argued that supporting perfect privacy in a decentralized reputation system is impossible [132]. That being said, they proposed three probabilistic schemes that are able to support partial privacy in decentralized additive reputation systems. The proposed schemes use secret splitting and secret sharing schemes, e.g., the Pederson secret sharing [133].

There are other works related to privacy-preserving reputation systems. In [77], the authors provided the k-shares protocol, which was inspired by the protocol of [132]. The advantage of k-shares protocol is that it has a lower message complexity compared to the protocol proposed in [132], i.e. $O(n)$ versus $O(n^2)$. The authors in [43] introduced dynamic preivacy-preserving reputation systems (Dyn-PDRS). Dyn-PDRS is able to deal with the dynamic structure of some decentralized reputation systems wherein nodes (users) in the network leave and join the network constantly. The authors in [179] discussed how a trust mechanism can be used in the blockchain technology for developing a decentralized personal data management system.

6.3 PRELIMINARIES

6.3.1 Secure Multiparty Computation (secure MPC)

Secure multiparty computation (secure MPC) is a computational model in which a group of parties can evaluate a public function on their private data without revealing their data. This idea was first introduced by Andrew Yao [169]. Secure MPC, a.k.a., secure function evaluation (SFE) [116], can be realized using cryptographic primitives such as secret sharing schemes, homomorphic encryption techniques and Yao's Garbled circuits. In secret sharing-based MPC, a secret sharing scheme, e.g. the Shamir's (t, n)-threshold secret sharing [150], is used to generate and distribute the shares of secrets (private data) among the participating parties. The computations are then carried out on the shares of those secrets. At the end of the computations, an appropriate technique, e.g., the Lagrange interpolation, is used to obtain the result.

Secure MPC based on Secret Sharing

Secure MPC based on the Shamir's secret sharing scheme works as follows. First of all, it should be noted that in secure multiparty computation usually there are n parties and each has a private value (which can be considered as a secret). Moreover, the computations are performed in a finite field such as Z_p, where p is a prime number. In order to perform a computation (evaluate a function) using secure MPC, each party first selects a polynomial $f(x) \in Z_p[x]$ whose coefficient are random values in Z_p and its constant term is the party's secret (private value). Mathematically speaking, each party P_i selects a polynomial as follows:

$$f_i(x) = \alpha_i + a_{i,1}x + a_{i,2}x^2 + ... + a_{i,t-1}x^{t-1}. \tag{6.1}$$

where α_i is the secret of party P_i, for $i = 1, 2, ..., n$ and $a_{i,1}$, $a_{i,2}$ ..., $a_{i,t-1}$ are random numbers in Z_p. Moreover, t is the threshold of the secret sharing scheme. Each party then evaluates its polynomial on n points, such as $1, 2, ..., n$ to generate the shares of its secret. The parties then distribute the shares of their secrets among each other. To evaluate a function securely, the parties perform the required computations on the shares of their data. They finally carry out a Lagrange interpolation on their updated shares to obtain/reconstruct the result of their computation (i.e., the function value).

6.3.2 Floating-Point Representation of Real (Rational) Numbers

Secure computation techniques mostly work on integer numbers, i.e. on finite field elements. In our secure trust evaluation method, the trust values are rational numbers in $[-1, +1]$ interval. Therefore, we need to use secure MPC techniques on real numbers. There are different encoding approaches, e.g., floating-point representation [8], that allow secure computation techniques to be used on real numbers. In this paper, we use floating-point representation [8] of real numbers in our proposed protocols, although other approaches can be used.

Floating-point representation is a way of representing real numbers using a fixed-precision significand v and an exponent p. The exponent p specifies how the real number should be scaled in a given base. For instance, when the base is 2, the representation would be of the form $v \,.\, 2^p$. In order to have a workable representation, the authors in [8] used a 4-tuple (v, p, z, s) with base 2 to represent each real value u. In this representation, v is an l-bit significand and p is a k-bit exponent. Moreover, z is a binary value which is 1 if and only if $u = 0$. Furthermore, s is the sign bit. The sign bit s is set when the value u is negative. For a real value u represented as above, we have $u = (1-2s)(1-z)v \,.\, 2^p$. In other words, the equation $u = (1-2s)(1-z)v \,.\, 2^p$ is used to convert a floating-point representation to a real number and vice versa.

6.3.3 Information Theoretic Framework for Modeling Trust [156]

The concept of trust (in human interactions or social networks) is very related to the concept of uncertainty in information theory. This subtle connection was formalized in [156], wherein an information theoretic framework for modeling trust was introduced. Due to the similarity between trust and uncertainty, trust can be measured by entropy, which is a well-accepted concept in information theory. Having said that, two trust models were proposed in [156], an entropy-based trust model and a probability-based trust model. For the probability-based trust model, two approaches were studied, a

Binomial distribution and a Bayesian approach. The authors [156] discussed that the Bayesian approach captures the concept of uncertainty more appropriately.

The information theoretic framework for modeling trust works based on the observations of nodes of each other. In what follows we briefly explain how trust is evaluated in this framework. Assume a network is given and node A in the network wants to evaluate its trust (for performing an action, e.g., packet forwarding) in another node, say node X. To do so, the past behaviors of node X regarding that specific action is considered. In the trust model based on the Bayesian approach, first the probability of node X performing that action is calculated. If node X has been asked to perform an action N times and among them node X has performed that action k times, the probability of performing that action in the next request, i.e., the $N+1$-th request, is defined as follows [156]:

$$Pr(V(N+1)) = \frac{k+1}{N+2} \tag{6.2}$$

wherein k is the number of times that node X has performed a specific action upon N total requests. In fact, $Pr(V(N+1))$ is the probability that node X will perform that specific action in the $(N+1)^{th}$ request. Note that $V(i)$ is the random variable of performing an action at the i-th request [156]. In the information theoretic framework for modeling trust, trust can also be calculated as entropy (which in fact measures the uncertainty). Having the probablity-based trust value, the entropy-based trust value of node A in node X for performing an action is defined/calculated as follows [156]:

$$T(A:X, action) = \begin{cases} 1 - H(p), & \text{for } 0.5 \leq p \leq 1 \\ H(p) - 1, & \text{for } 0 \leq p < 0.5 \end{cases} \tag{6.3}$$

where $H(p) = -p\log_2(p) - (1-p)\log_2(1-p)$ and p is the probability as defined in equation 6.2. The information theoretic framework [156] is an elegant way of modeling the concept of trust. There are a few points that should be emphasized. The trust values in the information theoretic framework can be represented as probability-based values or entropy-based values. Equation 6.3 shows the relation between these two types of trust values and how they can be converted to each other. It is also important to note that probability-based trust values are in $[0, 1]$ interval, whereas entropy-based trust values vary within the $[-1, 1]$ interval. In our STE method, the trust values are in $[-1, 1]$ interval.

6.4 THE PROPOSED SECURE TRUST EVALUATION (STE) TECHNIQUES

In this section we describe our proposed secure trust evaluation techniques. We would like to emphasize that our proposed protocols rely on the information theoretic framework [156] for modeling and measuring trust values. Moreover, for evaluating the trust values in a secure or privacy-preserving way our protocols rely on secure computation based on Shamir's secret sharing [150].

6.4.1 Secure Trust Evaluation using Multipath Trust Propagation

Trust in a network can propagate in different ways. In this section, we briefly discuss how a node can evaluate its trust in another node using the multipath trust propagation approach. In the multipath trust propagation, a node (say node A_1) wants to evaluate its trust in another node (say node B). For this purpose, node A_1 asks (or sends trust recommendation requests to), other nodes say nodes A_2, A_3, ..., A_n, in the network to give their opinions about node B. Figure 6.1 shows a sample multipath trust propagation in a network.

After receiving the trust values (from other nodes, i.e., A_2, A_3, ..., A_n), node A_1 calculates its trust in node B as follows [156]:

$$T_{A_1B} = Trust(A_1 : B) = \sum_{i=1}^{n} w_i T_i \tag{6.4}$$

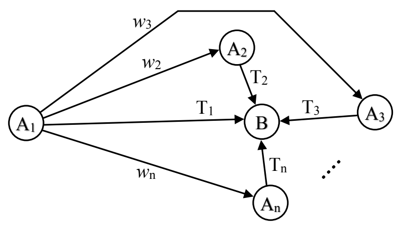

Figure 6.1: Multipath trust propagation [137] (see also [156])

where T_1 is the trust value of node A_1 in node B (i.e., based on direct observation) and w_1 is the weight that node A_1 considers for its direct trust value in node B. Moreover, T_i, for $i = 2, ..., n$, is the trust value (opinion) of node A_i in node B, which is returned from node A_i to node A_1. w_i is the weight that node A_1 considers for its trust in node A_i for $i = 2, ..., n$ (see figure 6.1). Note that the weights w_i's are selected by node A_1, such that $0 \leq w_i \leq 1$ and $\sum_{i=1}^{n} w_i = 1$. Note that since $0 \leq w_i \leq 1$ and $-1 \leq T_i \leq 1$ we have $-1 \leq T_{A_1B} \leq 1$. The maximum value that T_{A_1B} can get is when $T_i = 1$. In that case, $T_{A_1B} = \sum_{i=1}^{n} w_iT_i \leq \sum_{i=1}^{n} w_i = 1$. The minimum value that T_{A_1B} can get is when $T_i = -1$, where we have $T_{A_1B} = \sum_{i=1}^{n} w_iT_i \geq \sum_{i=1}^{n} w_i(-1)$ $= -1 \times \sum_{i=1}^{n} w_i = -1$. Recall that it is assumed $\sum_{i=1}^{n} w_i = 1$.

We now propose a protocol that allows a node in a network, e.g., node A_1, to evaluate its trust in another node, e.g., node B, using the multipath trust propagation approach (see also Fig. 6.1). The idea is that the nodes on the multipath perform their computation using secure multiparty computation. To do so, the nodes use the Shamir's secret sharing scheme to share their secrets (in this case their trust values in each other) and perform the computations in a secure fashion. Note that the nodes on the multipath illustrated in figure 6.1 need to securely evaluate the function represented in equation 6.4. In equation 6.4, w_i's are private values of node A_1, while T_i is the private value of node A_i for $i = 2, ..., n$. Protocol 1 shows the secure trust evaluation procedure using the multipath trust propagation approach.

Protocol 1: Secure Trust Evaluation using the Multipath Approach [137]

```
Require: Trust values $\{T_1, T_2, ..., T_n\}$ and weights $\{w_1, w_2, ..., w_n\}$.
Ensure: Calculates $T_{A_1B} = \sum_{i=1}^{n} w_i T_i$ using secure MPC.
1: Each party (node) $A_i$, for $i = 1, 2, ..., n$, uses floating-point representation
   to encode its input into a single finite field element.
2: Each party $A_i$ uses the Shamir's secret sharing scheme to generate the
   shares of its input $T_i$. Party $A_i$ selects a polynomial as follows:
       $f_i(x) = T_i + a_{i,1}x + a_{i,2}x^2 + ... + a_{i,t-1}x^{t-1}.$
   where $T_i$ is the trust value of node $A_i$ in node $B$.
3: Party $A_1$ uses the Shamir's secret sharing scheme to generate the shares
   of its weights, i.e., $w_i$'s. $A_1$ selects a polynomial as follows:
       $g_i(x) = w_i + b_{i,1}x + b_{i,2}x^2 + ... + b_{i,t-1}x^{t-1}.$
   where $w_i$ is the weight that party $A_1$ considers for node $A_i$.
4: Each party distributes the shares of its input among all the parties. The
   share-exchange matrix (wherein party $A_i$ generates the $i$-th row and
   receives the $i$-th column) is as follows:
       $E_f = \begin{bmatrix} f_1(1) & f_1(2) & \dots & f_1(n) \\ \vdots & \vdots & \ddots & \vdots \\ f_n(1) & f_n(2) & \dots & f_n(n) \end{bmatrix}$
```

```
5: Party $A_1$ distributes the shares of its weights $w_i$'s, for $i = 1, ..., n$. The
   share-exchange matrix is as follows:
```

$$E_g = \begin{bmatrix} g_1(1) & g_1(2) & \dots & g_1(n) \\ \vdots & \vdots & \ddots & \vdots \\ g_n(1) & g_n(2) & \dots & g_n(n) \end{bmatrix}$$

```
6: Party $A_i$, for $i = 1, 2, ..., n$, performs the following computation:
```

$$T^i_{A_1B} = \sum_{k=1}^{n} g_k(i) \times f_k(i).$$

```
   where $g_k(i)$ is the share of $w_i$ that party $A_i$ has received from party $A_1$ and
   $f_k(i)$ is the share of $T_k$ that party $A_i$ has received from party $A_k$. Moreover,
   $T^i_{A_1B}$ means the share of party $A_i$ of the trust value $T_{A_1B}$. Note that after
   each multiplication, $g_k(i) \times f_k(i)$, the participating parties must perform a
   degree reduction procedure [128].
7: Each party $A_i$ for $i = 2, 3, ..., n$ sends its result of the computation in the
   previous step to party $A_1$.
8: Party $A_1$ uses Lagrange interpolation to obtain the result (i.e., $T_{A_1B}$):
```

$$T_{A_1B} = \sum_{i=1}^{n} (\prod_{\substack{k=1 \\ k \neq i}}^{n} \frac{k}{k-i} \times T^i_{A_1B})$$

6.4.2 Secure Trust Evaluation using Referral Chains

The idea of using referral chains (referral graphs) in trust and reputation systems was introduced in [171] and further studied in [126, 172]. Yu and Singh [171] defined a referral chain as follows. Given the graph representation of a network (e.g., a social network), a referral chain from node A_0 to node A_n is basically a path between the two nodes. Such a referral chain is represented as $\chi = \langle A_0, A_1, ..., A_n \rangle$, where A_i is a neighbor of A_{i+1}.

The concept of referral chain in a network can capture the notion of trust propagation in a good way. In [171], the authors used this concept for estimating the quality of nodes in a trust net. In a trust net [171], the trust value of a node (say node A) in another node (say node B) is measured based on three factors [126, 171]: A's direct observation of B, the B's neighbors opinion about B and the A's opinion about the neighbors of B. Having the trust values of the nodes on a referral chain, the trust over the referral chain propagates according to the trust propagation operator (see definitions 5 and 6 of [171]). In our secure trust evaluation method, we consider a general case of a referral chain consisting of n nodes as illustrated in Fig. 6.2:

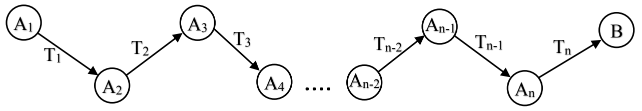

Figure 6.2: A referral chain in a network [137] (see also [156])

The trust value of node A_1 in the last node on the referrl chain, i.e., node B, is calculated as follows [171]:

$$T_{A_1B} = T_{A_1A_2} \otimes T_{A_2A_3} \otimes ... \otimes T_{A_nB} \tag{6.5}$$

where $T_{A_iA_{i+1}}$, for $i = 1, 2, ..., n-1$, is the trust value of node A_i in node A_{i+1} and it is represented as T_i in Fig. 6.2. Moreover, $\otimes$ represents the trust propagation operator, which is defined as follows [171]:

Definition 1 ([171])**.** $x \otimes y :=$ if $(x \geq 0 \wedge y \geq 0)$ then $x \times y$; else $-|x \times y|$.

The trust propagation on a referral chain is defined as follows [171]:

Definition 2 ([171])**.** For any k, where $k \in \{1, 2, ..., n\}$, the trust of A_1 in A_k is defined as: $T_{A_1 A_k} = T_{A_1 A_2} \otimes T_{A_{k-1} A_k}$.

In the following, we propose a protocol that enables a node in a network to evaluate its trust in another node through a referral chain. The main idea is that the nodes on the referral chain use secure MPC based on secret sharing to carry out the trust evaluation computations (i.e., the function in equation 6.5) in a secure fashion. The procedure of secure trust evaluation on a referral chain is described in Protocol 2. Note that in Protocol 2 the trust value of node A_i in node A_{i+1} is represented as T_i, where $i = 1, ..., n$. That is, $T_i = T_{A_i A_{i+1}}$.

Note that in the trust propagation operator, i.e., $\otimes$ in definition 1 and equation 6.5, before the multiplication of each pair of trust values, the trust values are compared with zero (i.e. if $x \geq 0 \wedge y \geq 0$). Thus, in order to carry out the trust propagation operator in Protocol 2, each pair of trust values need to be securely compared with zero. This can be done in different ways. One solution is to use a secure comparison protocol, e.g. a protocol from Table 3.1 in section 3.3.1 of chapter 3. Another approach is to use secure MPC for determining the sign of the final trust value, i.e., $T_{A_1 B}$, as follows. Each party (node) A_i encodes and shares the sign of its trust value T_i: If A_i's trust value is positive (i.e. $0 \leq T_i < 1$), then A_i shares 0 among all the parties. If A_i's trust value is negative (i.e. $-1 \leq T_i < 0$), then A_i shares 1 among all the parties. All the parties exchange and add their shares and send their result to party A_1. By obtaining the final result (using the Lagrange interpolation), party A_1 can determine the sign of the final trust value as follows: If the final result is 0, the sign of the final trust value (i.e. $T_{A_1 B}$) is positive. If the final result is a non-zero value, the sign of the final trust value (i.e. $T_{A_1 B}$) is negative.

Protocol 2: Secure Trust Evaluation using Referral Chain Approach [137]

Require: Trust values $\{T_1, T_2, ..., T_n\}$, where $T_i = T_{A_i A_{i+1}}$, for $i = 1, .., n$.

Ensure: Calculates $T_{A_1 B} = T_1 \otimes ... \otimes T_n$ using secure MPC.

1: Each party A_i (i.e., each node on the referral chain) uses floating-point representation to encode its input T_i into a single finite field element.

2: Each party A_i uses the Shamir's secret sharing scheme to generate the shares of its trust value T_i. Party A_i selects a polynomial as follows:

$$f_i(x) = T_i + a_{i,1}x + a_{i,2}x^2 + ... + a_{i,t-1}x^{t-1}.$$

where T_i is the trust value of node A_i in node A_{i+1} on the chain.

3: Each party A_i distributes the shares of its trust value among all the parties. The share-exchange matrix (wherein party A_i generates the i-th row and receives the i-th column) is as follows:

$$E_f = \begin{bmatrix} f_1(1) & f_1(2) & \dots & f_1(n) \\ \vdots & \vdots & \ddots & \vdots \\ f_n(1) & f_n(2) & \dots & f_n(n) \end{bmatrix}$$

4: Each party A_i multiplies its received shares:

$$T^i_{A_1 B} = \prod_{k=1,...,n} f_k(i)$$

where $f_k(i)$ is the share that party A_i has received from party A_k where $k = 1, 2, ..., n$. Moreover, $T^i_{A_1 B}$ means the share of party A_i of the trust value $T_{A_1 B}$. Note that after each multiplication the participating parties must perform a degree reduction procedure [128].

```
5: Each party $A_i$ for $i = 2, 3, ..., n$ sends its result of the multiplication, in the
   previous step, to party $A_1$.
6: Party $A_1$ uses Lagrange interpolation to obtain the result (i.e., $T_{A_1B}$):
```

$$T_{A_1B} = \sum_{i=1}^{n} \Big(\prod_{\substack{k=1 \\ k \neq i}}^{n} \frac{k}{k-i} \times T^{i}_{A_1B} \Big)$$

We would like to note that a disadvantage of the referall chain approach [171] is that on long chains the trust propagation operator fades the trust value of node A_1 in node B [126]. An alternative solution [126] for the referral chain approach is to use a weighted average of the trust values on the chain, where the weights decrease monotonically, i.e., $1 \geq w_1 > w_2 > ... > w_n \geq 0$. Note that w_i's are selected such that $\sum_{i=1}^{n} w_i = 1$. In such a monotonically-decreasing weighted referral chain, the trust value T_{A_1B} can be securely evaluated using Protocol 1.

6.4.3 Secure Network Routing

The concept of trust, as a soft security mechanism, can be used for improving the quality of network services in different ways. For instance, trust models can improve network routing protocols and provide malicious node detection capability [156]. An important thing in most networks is the security and privacy of the nodes' data. It is important for the nodes in a network to not reveal their private data, e.g., their trust values in each other [144] and [156]. This is because if trust values are revealed, nodes with high trust values may be compromised by adversaries. This can reduce the trustworthiness of the whole network.

In this section, we use our proposed protocols (Protocol 1 and Protocol 2) to provide a secure network routing protocol. By using the secure network routing protocol, a node in a network can find a high quality route in a network while the nodes' private data is not revealed. Secure network routing protocols can provide more secure and trustworthy protocols in the sense that adversaries are not able to figure out how an action, e.g., packet forwarding in a network, is carried out. We first need to define the quality of a route in a network.

The Quality of a Route in a Network. Assume a network is given and node A and node N_{dest} are two nodes in that network. Moreover, suppose node A wants to perform an action in the network, e.g., to forward a packet to node N_{dest}. There are usually different routes in the network for performing such an action. In order to perform the packet forwarding action with a higher chance of success, node A can determine the quality of each route prior to forwarding its package to the destination. One approach to define the quality of a route in a network is based on the trust value of the nodes on that route [156]. Suppose R is a route in a network and $\{N_i\}$ represents the set of all nodes on route R. We define and calculate the quality of route R as follows [156]:

$$Quality(R) = \begin{cases} \prod_i T_i & \text{if } T_i > 0 \ \forall \text{ nodes } N_i \text{ on } R \\ \min\{T_i\} & \text{otherwise} \end{cases} \tag{6.6}$$

where T_i is the trust value of node A in node N_i on route R. Equation 6.6 is basically the multiplication of the trust values of the nodes on route R. In cases that there are nodes with negative trust values on the route, we define the quality of route as the minimum trust value (i.e., as the smallest negative trust value).

We now propose a protocol that enables a node in a network to evaluate the quality of a route in a secure manner. Our proposed secure network routing protocol works as follows. Assume node A wants to evaluate the quality of route R. Node A evaluates the trust value of each node on the route using the secure trust evaluation protocols (Protocol 1 and Protocol 2). Then, node A calculates the quality of the route using equation 6.6. To find a high quality route, node A must calculates the quality of different possible routes (to its desired destination) and finds the route with the highest quality. The secure network routing protocol is provided in Protocol 3. Note that in the protocol we assumed that each node, including node A, has a trust record on which the trust values are stored.

Protocol 3: Secure Network Routing Protocol [137]

```
Require: Nodes' trust records, i.e. nodes observations from each other.
Ensure: A high quality route in the network, from node $A$ to node $N_{dest}$.
1: Let $\{S_i\}$ denotes the set of all the nodes on all possible routes between
   node $A$ and node $N_{dest}$ in the network.
2: for any node $S_i$: do
       if node $A$ has a trust record about node $S_i$ then
           Node $A$ uses that trust record.
       else
           Node $A$ sends trust recommendation request about node $S_i$ to
           other nodes. Node $A$ collaboratively with other nodes use Protocol 1
           and Protocol 2 to securely evaluate its trust value in node $S_i$.
3: Let $R$ denote a particular route in the network and let $\{N_i\}$ represent
   the set of all the nodes on route $R$. Let $T_i$ denote the trust value of
   node $A$ in node $N_i$. Node $A$ calculates the quality of route $R$ as:
```

$$Quality(R) = \begin{cases} \prod_i T_i & \text{if } T_i > 0 \ \forall \text{ nodes } N_i \text{ on } R \\ \min\{T_i\} & \text{otherwise} \end{cases}$$

Note that the above multiplication is performed localy by node A. However, each T_i is computed securely when node A does not have a trust record about node N_i (see step 4).

```
4: Let $\{R_i\}$ denote the set of routes from node $A$ to node $N_{dest}$ in the network
   among which node $A$ wants to find a good quality route. Node $A$ selects a
   route which has a good quality, e.g., larger than a threshold or the route
   with the maximum quality, as follows:
       $R^* = argmax_{R_i}\{Quality(R_i)\}$
5: Node $A$ updates its trust records using the recent observations and
   calculated trust values.
6: Node $A$ initiates its desired action on the high quality route, i.e. route $R^*$.
```

6.5 DISCUSSION

In this paper, we introduced a secure trust evaluation (STE) method. Our proposed method is based on the information theoretic framework for modeling trust and two approaches of trust evaluation in a network: the multipath trust propagation and the idea of referral chains in a network. There are a few comments that we discuss in the following.

The Beta reputation system [91] is a specific case of the information theoretic framework for modeling trust. To see this, note that the trust value in the information theoretic framework is measured using equation 6.2 in section 6.4. In the Beta reputation system, the reputation of a user is calculated as $\frac{r+1}{r+s+2}$ (see [91] and [94]). The Beta reputation system is one of the commonly referred reputation systems in the literature. Thus, our secure trust evaluation method can be used wherever the Beta reputation system is applicable. For instance, our proposed protocols can be used in computer networks and Internet-based services that use the Beta reputation system.

Another important point is that our proposed STE method is a decentralized system. This has its own advantages and makes a network more reliable and trustworthy, because the nodes in a network do not reveal their private values to any third party or any other nodes. Recall that the secure protocols (Protocol 1 and Protocol 2) in our trust evaluation method use secure MPC and secret sharing schemes, e.g., the Shamir's (t, n)-threshold secret sharing scheme, which are powerful tools for secure function evaluation.

An overlooked fact in many privacy-preserving trust and reputation systems (TRS) is that regardless of using the cryptographic primitives or any other privacy measure, a ratee in a TRS can figure out the impact of a rater's feedback (rate) on its reputation [94]. This is because a feedback is usually provided after a transaction is completed; thus the ratee knows when the rater leaves his feedback. The ratee then can see the impact of that feedback on their reputation. Although the ratee might not be able to figure out the exact feedback rate, it is, at least, able to figure out whether the feedback was positive or negative.

Our proposed secure trust evaluation method addresses the above-mentioned issue appropriately. In our model, when a node (say node A) in a network wants to evaluate its trust in another node (say node B), node A asks other nodes for their rating about node B. The process of evaluating the trust value of node A in node B is carried out in such a way that the latter node, i..e node B, may even not notice that its trust is evaluated by other nodes. This makes sense because in a decentralized trust and reputation system, the nodes are witnesses for each others' behavior. Recall that in our trust model, the trust value of a node is evaluated as a weighted average of other nodes' ratings in the network (see equations 6.4).

6.6 SECURITY AND COMPLEXITY ANALYSIS

The security analysis of the protocols is inherited from the security of the underlying secret sharing scheme, which is the Shamir's scheme [150]. In our proposed protocols, the participating parties use the Shamir's secret sharing scheme to generate distribute the shares of their secrets (i.e. trust values). They then perform their computations on the shares of trust values, rather than on the trust values directly. Note that our protocols work in a semi-honest (passive) adversarial model. In other words, we assumed that the nodes in the network are honest-but-curious. In a passive adversarial model, the participating parties act honestly and follow the protocols' rules, but they are curious to learn other parties' private data.

We now discuss the complexity analysis of our proposed protocols. Recall that there are n parties in each protocol. In Protocol 1, the distribution of shares (in step 4 and 5) takes two rounds of communication, each one with $n-1$ exchanges of shares. Each party also performs a multiplication of its shares (in step 6 of Protocol 1) and performs $n-1$ additions. Note that for each multiplication one degree reduction is carried out. Each degree reduction consists of two rounds of communication, and n multiplication in total. Thus, in total $O(n^2)$ multiplication is carried out. Another round of communication is carried out in step 7 of Protocol 1. In total, the complexity of Protocol 1 is five rounds of communication and $O(n^2)$ multiplication of finite field elements.

For Protocol 2, the parties distribute their secrets (i.e. trust values) in one round of communication (in step 3 of Protocol 2). Then they carry out $n-1$ multiplications of finite field elements. For each multiplication, one degree reduction (with two rounds of communication and n multiplication) is carried out. This results in a computation complexity of $O(n^2)$. One round of communication is also carried out in step 5 of Protocol 2. In total, the complexity of Protocol 2 is three rounds of communication and $O(n^2)$ multiplication of finite field elements.

The complexity of the network routing protocol (Protocol 3) depends on the complexity of Protocol 1 and Protocol 2. Assuming there are r routes in a network and each route has at most m nodes, the complexity of the proposed routing protocol is $O(r \times m)$ rounds of communication and $O(r \times m \times n^2)$ multiplication of finite field elements, where n is the maximum number of participating parties in secure multi-party computation. It should be noted that in general the route discovery problem in a network has an exponential communication complexity [156]. However, in Protocol 3 we assume that the routes in the network are already given.

6.7 CONCLUSION

In this paper, we introduced a secure trust evaluation (STE) method. Our STE method consists of two protocols that allow the nodes a network to securely evaluate their trust in one another. The proposed protocols in our STE method use secure multiparty computation (secure MPC) based on the Shamir's secret sharing scheme to guarantee the security and privacy of the parties private data. As an application, we also proposed a secure network routing protocol that shows how our proposed STE method can be used for improving network protocols.

Our proposed STE method can be used in different networks, for providing more secure and trustworthy networks/services. Our STE method relies on the information theoretic framework for modeling trust is a powerful trust model. As such, our STE method captures other trust and reputation systems, e.g., the Beta reputation system and the weighted average reputation model. Indeed, soft security measures (e.g., trust and reputation mechanisms) can compliment hard security measures (e.g., cryptography and secure MPC) to provide more reliable and trustworthy network.

CHAPTER 7
SECURE TRUST EVALUATION (STE) USING AGGREGATION OF TRUST EVIDENCE

In this chapter we provide a secure trust evaluation (STE) technique based the aggregation of trust evidence [166]. Our proposed STE can use any secret sharing scheme and we used Shamir's secret sharing scheme [150] as an example. Compared to the previous work of [166], that uses two non-colluding third parties, our proposed STE scheme does not rely on any third parties. That is to say, the parties in a trust and reputation system (TRS) are considered as a peer-to-peer (P2P) network and can evaluate their trust values in each other collaboratively.

7.1 INTRODUCTION

In the last couple of decades, numerous models and mechanisms have been proposed for evaluating social mechanisms such as trust and reputation. These social mechanisms are believed to enhance the reliability and security of systems and services. As such, they have been studied and applied in different areas and application domains [90] including in Internet of Things (IoT) [74, 167], cloud computing [84], C2C e-commerce [177], social networks [151], wireless sensor networks (WSN) [75], peer-to-peer (P2P) networks [1, 92, 163], vehicle-to-vehicle (V2V) networks [176], ad hoc networks [14, 41] etc.

Social mechanisms such as trust and reputation are considered as soft security measures that complement hard security measures [171]. In other words, using mechanisms such as trust and reputation alongside cryptographic techniques enables us to provide more trustworthy and reliable services and systems. While these mechanisms altogether provide better security than what they provide solely, utilizing trust and reputation mechanisms has its own challenges and opens up doors for other vulnerabilities [106]. For example, the cheating entities in trust and reputation systems (TRS) can fake themselves as trusted entities or a group of entities can attack trusted entities to take advantage of them. Another example is that the entities might not be willing to participate in trust and reputation assessment due to the fear of retaliation [76, 106, 132, 144, 147].

Secure multiparty computation (secure MPC) is a good workaround that can address the above-mentioned challenges and relieve such vulnerabilities to some good extent [76]. Secure MPC enables a group of (possibly untrusted) parties to evaluate a public function on their private data without revealing their input data [166]. This computational paradigm provides promising solutions for privacy-preserving computation and secure function evaluation (SFE). Secure MPC has already been utilized in different application domains including in privacy-preserving trust evaluation [166], secure trust evaluation (STE) [137] and other domains such as privacy-preserving data mining [105], privacy-preserving scientific computation [57] etc.

Motivated by the applications of trust and reputation systems in different application domains, in this article we use secure MPC and secure function evaluation (SFE) techniques to provide a secure trust evaluation (STE) method. Our proposed STE technique can be deployed in different internet-based applications such as Internet of Things (IoT), cloud computing as well as other network-based systems to provide more secure and trustworthy services.

7.1.1 Our Contribution

In this article we propose a secure trust evaluation (STE) technique that enables the entities in a TRS system to securely assess their trust values in each other. More specifically, we use Shamir's secret sharing scheme [150] to provide a STE technique that enables a group of entities in a TRS system to securely evaluate their trust values in each other without relying on other third parties. Our proposed STE scheme is based on the aggregation (summation) of trust evidences and adapts the trust evaluation approach of [166].

The STE technique that we propose provides a couple of improvements compared to the schemes of [166]. First, the schemes of [166] use two non-colluding third parties (i.e., servers) in order to preserve the privacy of the entities in the trust evaluation system. Our proposed STE method is decentralized and does not rely on any sort of third parties. In other words, in our STE scheme the parties involved in trust evaluation securely evaluate their trust in each other without using any third parties. An advantage of our STE technique is that as it is based on the summation of trust evidences, it is a good candidate to be efficiently implemented with secret sharing schemes. Second, the schemes of [166] are based on Paillier encryption scheme [131], thus they provide computational security. Our proposed STE scheme is based on Shamir's secret sharing scheme [150], thus provides information-theoretical security.

7.2 RELATED WORKS

Trust and reputation have been vastly studied and investigated in different application domains. One of the early works on providing a computational model for trust was done by Marsh [114]. After that, there has been various studies on trust and reputation models and mechanisms [40, 74, 90, 121]. There are also numerous surveys that have discussed different aspects of trust and reputation in various domains, including in Internet of Things (IoT) [74, 167], cloud computing [40, 84], grid computing [100], internet applications [71], social networks [151], intelligent transportation system (ITS) [110], vehicular ad-hoc networks(VANETs) [176], mobile ad hoc networks (MANETs) [41]. Briefly speaking, there are a couple of approaches for modeling the notions of trust and reputation. Some commonly-used approaches for modeling trust include subjective logic [88, 89], fuzzy logic [113], entropy-based models [156], Dempster-Shafer theory of evidence [56, 70, 149, 165].

While significant amount of works has been done on studying trust and reputation systems, previous works have rarely taken into account the security and privacy of trust evidences during evidence aggregation and trust evaluation [166]. In what follows we briefly review the works that are related to our work.

The authors in [137] used secure multiparty computation based on Shamir's secret sharing to propose secure trust evaluation techniques based on multipath and referral chains trust propagation. Yan et al. [166] proposed two privacy-preserving trust evaluation schemes. The proposed schemes use two non-colluding trusted third parties and the Paillier homomorphic encryption scheme [131] to preserve the privacy of the entities in trust evidence aggregation. In [12], the authors used the notion of *mailbox* agents to propose protocols for privacy-preserving distributed reputation systems. A distributed privacy-preserving reputation mechanism, called Malicious-k-shares protocol, was proposed in [78]. The proposed machanism relies on the Paillier homomorphic encryption scheme [131] to carry out reputation computation in a decentralized approach.

The authors in [125] proposed the 3PRep protocol, which is the privacy-preserving version of the P2PRep reputation system [13]. The 3PRep protocol uses the Paillier homomorphic encryption scheme [131] to enhance the entities privacy in the P2PRep system. Pavlov et al. [132] used secret sharing schemes, e.g., the Pederson secret sharing [133], to preserve the privacy of the entities in decentralized additive reputation management systems. The researchers in [77] proposed the k-shares protocol. The k-shares protocol relies on the protcol of [132], but has a smaller communication complexity. Gudes et al. [73] used secure summation [132] and secure dot product [11] techniques to compute the reputation of entities in reputation systems in a privacy-preserving manner.

7.2.1 Privacy-Preserving Trust Evaluation Schemes of [166]

In [166], the authors proposed two privacy-preserving trust evaluation (PPTE) schemes that allow a node in a network to calculate its trust value in another node without revealing their trust values. In this section we briefly review the overall idea of the proposed schemes and how they work. Since the two schemes of [166] are similar and have minor differences, we briefly review their main idea in the computation part of trust evaluation.

The proposed PPTE schemes [166] consider a period of J time slots (which are represented by $t = \{t_1, t_2, ..., t_J\}$). The idea is that first the aggregation of trust evidences is calculated in each time slot (see equation 7.1). Then these aggregated trust evidences are fed into a trust evaluation function which produces the final trust value (see equation 7.2). The schemes, however, carry out the computation in a privacy-preserving fashion. To do so, they rely on the Paillier homomorphic encryption scheme [131] and use two trusted third parties (i.e., servers): an Authorize Proxy (AP) server and Evaluation Party (EP) server. The AP server is responsible for access control and managing collected trust evidences. The EP server, on the other hand, is responsible for processing and doing computation on the trust evidences. In particular, EP performs computation on encrypted data which were encrypted using AP's public-key for homomorphic encryption (i.e., HPK_{AP}). Figure 7.1, equation 7.1 and equation 7.2 demonstrate how to idea of trust evaluation using the aggregation of evidences works.

In the schemes of [166], the aggregation of trust evidences is carried out using the Paillier homomorphic encryption scheme [131] by the AP and EP servers [166]:

$$Enc\Big\{HPK_{AP}, te_j(y)\Big\} = \prod_{i=1}^{N_j} Enc\{HPK_{AP}, te(z_i, y)\}. \tag{7.1}$$

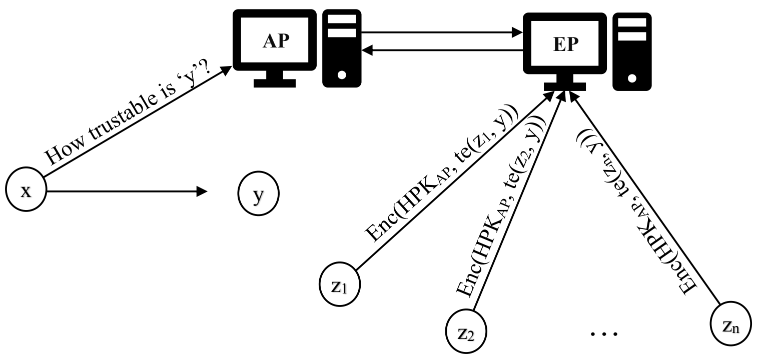

Figure 7.1: Aggregation of Trust Evidences using two Trusted Third Parties, i.e., AP and EP (the proposed model in [166])

wherein $te_j(y) = \sum_{i=1}^{N_j} te(z_i, y)$ denotes the aggregation of trust evidences in the j-th time slot. $te(z_i, y)$ denotes the trust evidence provided by node z_i about node y. Moreover, N_j indicates the number of provided trust evidences (also the number of evidence providers) in the j-th time slot.

When a node (e.g., node x) wants to evaluate its trust in another node (e.g., node y), node x first asks the EP server for trust pre-evaluation. EP then collects trust evidences about node y from trust evidence providers (i.e., nodes z_i). The evidence provider nodes (e.g., nodes z_i, for i = 2, ..., N_j) encrypt their trust evidences using the public key of the AP server (i.e., HPK_{AP}) and send the results to the EP server. EP calculates the summation of the evidences by multiplying the encrypted trust evidences (see equation 7.1). Since the schemes use the Paillier encryption scheme [131], which is an additive homomorphic encryption scheme, the multiplication result is an encryption of the summation of trust evidences. After finding the summation of trust evidences, EP sends the result along with the statistics of trust evidences $s_j(y)$ (e.g., the number of evidences N_j) to the AP server. AP then checks node x's access control and verifies whether node x's secret key is valid. If node x is eligible to get the result, AP first decrypts the result. AP then re-encrypts the result with node x's public key and sends the result along with the statistics of trust evidences (e.g., the number of evidences) to node x. Finally, node x decrypts the result and gets the summation of trust evidences.

By having the aggregation of trust evidences, node x can then use a trust evaluation function to calculate its trust value in node y. The trust evaluation function of [166] is given in equation 7.2. The inputs of this function are as follows. $Tv'(y)$ denotes node x's personal (direct) trust in node y. Moreover, $te(y)$ denotes the set of aggregated trust evidences in all J time slots. That is, $te(y) = \{te_j(y) | j = 1, ..., J\}$ where $te_j(y) = \sum_{i=1}^{N_j} te_j(z_i, y)$ for $j = 1, ..., J$. Furthermore, $s(y) = \{s_j(y) | j = 1, ..., J\}$ represents the set of all statistics of trust evidences about node y, where $s_j(y)$ is the statistics of trust evidences at time slot t_j (e.g., the number of collected trust evidences N_j). Lastly, t_c denotes the evaluating time (i.e., the time whereat trust evaluation is performed) [166].

$$\begin{aligned} Tv(y) &= F(Tv'(y), te(y), s(y), t_c) \\ &= \alpha \times Tv'(y) + \beta \times \frac{1}{S} \sum_{j=1}^{J} \theta(s_j(y)) \times \frac{te_j(y)}{s_j(y)} \\ &\times (1 - |\frac{te_j(y)}{s_j(y)} - Tv'(y)|) \times e^{-\frac{|t_j - t_c|^2}{\tau}} \end{aligned} \tag{7.2}$$

where the weight parameters α and β are defined as $\beta = \frac{\eta}{J} \sum_{j=1}^{J} (1 - |\frac{te_j(y)}{s_j(y)} - Tv'(y)|)$ and $\alpha = 1 - \beta$. Moreover $S = \sum_{j=1}^{J} \theta(s_j(y)) \times (1 - |\frac{te_j(y)}{s_j(y)} - Tv'(y)|) \times e^{-\frac{|t_j - t_c|^2}{\tau}}$. Further, η is a parameter for adjusting the contribution of collected trust evidences. The function θ is the Rayleigh cumulative distribution function which is defined as follows [166]:

$$\theta(I) = 1 - exp(-I^2/2\sigma^2) \tag{7.3}$$

wherein $\sigma > 0$ is a parameter which adversely governs how the number I influences the increment of $\theta(I)$.

In summary, the schemes of [166] use the Paillier sncryption scheme [131] and rely on two non-colluding trusted third parties (i.e., the AP and EP servers) in order to enable node x to evaluate its trust in node y in a privacy-preserving fashion. The summation of trust evidences is calculated using equation 7.1 by the two servers and is handed out to node x along with the statistics of trust evidences (e.g., the number of collected trusted evidences). Node x then uses equation 7.2 to calculate the final trust value about node y.

7.3 SECURE TRUST EVALUATION USING AGGREGATION OF TRUST EVIDENCE

The proposed schemes of [166] use two trusted (semi-trusted) non-colluding servers (i.e., AP and EP servers) that perform trust evaluation in a privacy-preserving fashion. While this approach preserves the privacy of the nodes to a good extent, in reality finding such two non-colluding parties might not be easy. In particular, since such two third parties need to communicate in the setup phase in order to exchange their cryptographic keys, they can also simply reveal nodes' trust evidences to each other and violate nodes' data privacy. In fact, in most cases it might be in the two parties best interest to reveal nodes' data to each other.

Secure multiparty computation (secure MPC) based on Shamir's secret sharing [150] provides promising solutions for secure function evaluation (SFE), e.g., secure trust evaluation (STE). This approach of secure trust evaluation has two main advantages. First, there is no need for the two trusted third parties (i.e., the AP and EP servers). In other words, the nodes in a trust evaluation system can evaluate their trust in each other by themselves and without relying on any third parties. Second, secure trust evaluation based on Shamir's secret sharing provides information-theoretical security, whereas the schemes of [166] provide computational security. However, it should be mentioned that these improvements are achieved at the cost of more communication (which is created as a result of using secret sharing schemes rather than relying on homomorphic encryption techniques).

7.3.1 The Proposed Secure Trust Evaluation Technique

In this section we propose a secure trust evaluation (STE) technique that enables the entities in a TRS system to securely evaluate their trust in each other without relying on any third parties. More specifically, we use secure multiparty computation based on Shamir's secret sharing [150] to securely evaluate the aggregation of trust evidences and the trust evaluation function.

Our STE method works as follows. We consider a general trust evaluation system and treat it as a network (of nodes). Examples of such networks can be online social networks, networks of IoT objects, nodes in a wireless sensor network etc. We assume that there are two nodes x and y, where node x is interested in evaluating its trust in node y. In other words, node x wants to find out how trustable node y is. In order to measure the trust value of node x in node y, similar to the approach of [166] we consider J time slots. In each time slot t_j, for $j = 1$, ..., $j = J$, node x asks other nodes, who had interactions with node y in the past, to provide their trust evidences about node y. The evidence provider nodes are represented by z_i for $i = 2$, ..., n. Figure 7.2 shows nodes x, y and z_i's in a trust evaluation system, where node x wants to assess the trust value of node y and nodes z_i's, for $i = 2$, ..., n, are the evidence provider nodes.

Node x can get the trust evidences from nodes z_i and evaluate its trust in node y using equation 7.4 and equation 7.2. However, nodes z_i's are not willing to reveal their trust values about y. Our proposed STE method enables node x and nodes z_i to collaboratively evaluate its trust value in node y in a secure way.

$$te_j(y) = \sum_{i=1}^{N_j} te(z_i, y) \tag{7.4}$$

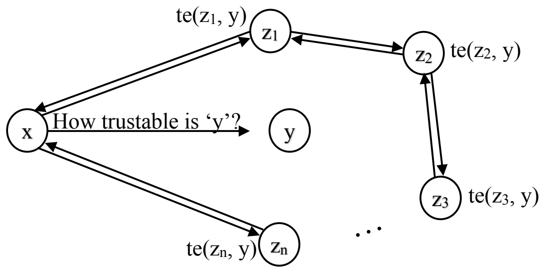

Figure 7.2: Aggregation of Trust Evidences without any Trusted Third Parties (our proposed model)

In order to securly evaluate the trust of node x in node y, nodes x and nodes z_i, for i = 2, ..., n, use Shamir's threshold secret sharing scheme [150] to generate and exchange the shares of their trust values among themselves. For the sake of simplicity, we assume that $x = z_1$ and other evidence provider nodes are represented by z_i for i = 2, ..., n, where n can be any positive integer which denotes the number of evidence provider nodes.

After exchanging the shares of trust evidences, node x and each node z_i, for i = 2, ..., n, adds up their shares locally. Nodes z_i, for i = 2, ..., n, then send their result to node x. Finally, node x performs a Lagrange interpolation on the received shares and gets the summation of trust evidences. After calculating the summation of trust evidences, node x uses equation 7.2 to calculate its trust in node y. Protocol 4 shows the step-by-step procedure of our proposed STE method.

7.4 COMPLEXITY ANALYSIS

In this section, we provide the security and complexity analysis of our proposed secure trust evaluation (STE) technique.

The security of our STE technique relies on secure MPC based on Shamir's secret sharing scheme [150]. Secure MPC enables a group of parties to securely evaluate a public function on their private data while the parties do not reveal their inputs to

Protocol 4: Secure Trust Evaluation using Aggregation of Evidences

```
Require: Trust evidences $\{te(z_1, y), te(z_2, y), ..., te(z_n, y)\}$.
Ensure: Calculates $Tv(y)$ (i.e., trust value of node $x$ in node $y$) using secure
    MPC based on Shamir's secret sharing scheme [150].
1: Let $te(y) \leftarrow \{\}$; and $s(y) \leftarrow \{\}$;
2: for $j = 1$ to $J$ do
3:   Each party $z_i$, for $i = 1, ..., N_j$, uses the Shamir's secret sharing scheme
     to generate the shares of its input $te(z_i, y)$. Party $z_i$ selects a polynomial
     as follows:
         $$f_i(x) = te(z_i, y) + a_{i,1}x + a_{i,2}x^2 + ... + a_{i,t-1}x^{t-1}.$$
     where $te(z_i, y)$ is the trust evidence provided by node $z_i$ about $y$.
4:   Each party distributes the shares of its input among all the parties. The
     share-exchange matrix (wherein party $z_i$ generates the $i$-th row and
     receives the $i$-th column) is as follows:
         $$E_f = \begin{bmatrix} f_1(1) & f_1(2) & \dots & f_1(N_j) \\ \vdots & \vdots & \ddots & \vdots \\ f_{N_j}(1) & f_{N_j}(2) & \dots & f_{N_j}(N_j) \end{bmatrix}$$
5:   Party $z_i$, for $i = 1, ..., N_j$, performs the following computation:
         $$er_{j,i}(y) = \sum_{k=1}^{N_j} f_k(i).$$
     where $f_k(i)$ is the share of $te(z_k, y)$ that party $z_i$ has received from
     party $z_k$. Moreover, $er_{j,i}(y)$ means the share of party $z_i$ from the
     aggregation of trust value $te_j(y) = \sum_{k=1}^{n} te(z_k, y)$ in time slot $t_j$.
   end
```

```
6: for $j = 1$ to $J$ do
7:     Each party $z_i$ for $i = 2, ..., N_j$ sends its result of the computation in the
       previous step to party $x$ (i.e., party $z_1$).
8:     Party $x$ uses Lagrange interpolation to obtain the aggregation of trust
       evidences in the $j$-th time slot (i.e., $te_j(y)$):
           $te_j(y) = \sum_{i=1}^{N_j}(\prod_{k=1, k\neq i}^{N_j} \frac{k}{k-i} \times er_{j,i}(y))$
       $te(y) \leftarrow te(y) \cup \{te_j(y)\}$; and $s(y) \leftarrow s(y) \cup \{s_j(y)\}$.
   end
9: Party $x$ uses his trust evidence $Tv'(y)$, statistics $s(y)$, the set of all
   aggregated trust evidences $te(y)$ and evaluating time $t_c$ to obtain
   the final trust value using the trust evaluation function $F$ [166]:
       $Tv(y) = F(Tv'(y), te(y), s(y), t_c)$
       $= \alpha \times Tv'(y) + \beta \times \frac{1}{S}\sum_{j=1}^{J}\theta(s_j(y)) \times \frac{te_j(y)}{s_j(y)}$
       $\times (1 - |\frac{te_j(y)}{s_j(y)} - Tv'(y)|) \times e^{-\frac{|t_j - t_c|^2}{\tau}}$
```

each other. Briefly, in secure MPC based on Shamir's secret sharing scheme, each party selects a polynomial with random coefficients and put his secret input as the constant term of the polynomial. Each party then generates the shares of his secret by evaluating his polynomial on different points and distribute the generated shares among the parties. To evaluate a function on the parties' secret inputs, the parties perform the function evaluation on the shares of the secret inputs. Finally, the parties perform a Lagrange interpolation on their updated shares and get the function value (output). Since the shares of secret values are completely random numbers and the computations are performed on the shares of inputs, no private data is revealed during the computation. Secure MPC based on Secret sharing is a well-known technique for secure function evaluation and it has been utilized in many application domains.

Our proposed protocol consists of J time slots (which are represented by $t = \{t_1, t_2, ..., t_J\}$). In each time slot, the nodes (participating parties) use Shamir's secret sharing scheme [150] to securely evaluate a node's trust value (e.g., node x) in another node (e.g., node y). As such, the round complexity of our STE technique is J rounds. In time slot t_j, where $j = 1, ..., J$, there are N_j nodes (parties) that participate in trust evaluation. In each time slot, the nodes cooperatively calculate the aggregation (summation) of their trust evidences. For each summation the nodes use the addition gate of secure MPC based on Shamir's secret sharing. Therefore, in total, $\sum_{j=1}^{J} N_j$ addition operations are required for calculating the summation of all trust evidences in all J time slots. It should be mentioned that the evaluation of the trust function (i.e., function F in equation 7.2) is performed by node x on node x's computational device.

7.5 CONCLUSION

Trust and reputation are social mechanisms that can increase the security and trustworthiness of internet-based and network-based services and systems. These measures have been vastly studied and utilized in different areas and application domains. In spite of that, previous works have rarely considered the privacy of entities involved in trust and reputation systems (TRS). In this article, we proposed a secure trust evaluation (STE) technique using secure multiparty computation based on Shamir's secret sharing scheme. The proposed technique is based on the aggregation (summation) of trust evidences. It enables the nodes in a TRS system to securely evaluate their trust in each other without revealing their trust values and without relying on any trusted third parties.

CHAPTER 8
CONCLUDING REMARKS

Secure multiparty computation (secure MPC) is a branch of modern cryptography that has attracted significant attention during the last couple of decades. In this dissertation we focused on different approaches for secure computation, namely fully homomorphic encryption (FHE) and secret sharing schemes.

In particular, we studied fully homomorphic encryption schemes based on the LWE [142] and RLWE [109] problems and provided a C++ implementation of the ring variant of a third generation FHE scheme, called the approximate eignevector method (a.k.a., the GSW scheme) [67] ([58] and [38]). We utilized our implementation for homomorphic evaluation of pseudorandom functions (PRFs), that can be used for improving other secure MPC protocols such as the SPDZ protocol and compiler [53]. Furthermore, we provided several novel protocols for secure trust evaluation (STE) [137]. Our proposed STE protocols, which are based on Shamir's secret sharing scheme [150], can be used for improving trust and reputation systems (TRS).

As secure computation becomes a reality in the digital era, we hope our research can contribute to this interesting area of research and can be utilized in technologies that rely on secure computation. Particularly, our proposed FHE-based constructions and secure protocols can be used for providing more secure and robust data/computation infrastructures.

BIBLIOGRAPHY

[1] Karl Aberer and Zoran Despotovic. Managing trust in a peer-2-peer information system. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 310–317, 2001.

[2] Damiano Abram and Peter Scholl. Low-communication multiparty triple generation for spdz from ring-lpn. In *IACR International Conference on Public-Key Cryptography*, pages 221–251. Springer, 2022.

[3] Mark Abspoel, Niek J Bouman, Berry Schoenmakers, and Niels de Vreede. Fast secure comparison for medium-sized integers and its application in binarized neural networks. In *Cryptographers Track at the RSA Conference*, pages 453–472. Springer, 2019.

[4] Abbas Acar, Hidayet Aksu, A Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (CSUR)*, 51(4):79, 2018.

[5] Carlos Aguilar-Melchor, Joris Barrier, Serge Guelton, Adrien Guinet, Marc-Olivier Killijian, and Tancrede Lepoint. Nfllib: Ntt-based fast lattice library. In *Cryptographers Track at the RSA Conference*, pages 341–356. Springer, 2016.

[6] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, et al. Homomorphic encryption standard. In *Protecting Privacy through Homomorphic Encryption*, pages 31–62. Springer, 2021.

[7] M. Aliasgari, M. Blanton, Y. Zhang, and A. Steele. Secure computation on floating point numbers. *In Network and Distributed System Security Symposium - NDSS 2013. Internet Society*, 2013.

[8] Mehrdad Aliasgari, Marina Blanton, Yihua Zhang, and Aaron Steele. Secure computation on floating point numbers. 2013.

[9] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In *Annual Cryptology Conference*, pages 297–314. Springer, 2014.

[10] Peter S Nordholt ALX, Nikolaj Volgushev ALX, Prastudy Fauzi AU, Claudio Orlandi AU, Peter Scholl AU, Mark Simkin AU, Meilof Veeningen PHI, Niek Bouman TUE, and Berry Schoenmakers TUE. D1. 1 state of the art analysis of mpc techniques and frameworks.

[11] Artak Amirbekyan and Vladimir Estivill-Castro. A new efficient privacy-preserving scalar product protocol. In *Proceedings of the sixth Australasian conference on Data mining and analytics-Volume 70*, pages 209–214, 2007.

[12] Emmanuelle Anceaume, Gilles Guette, Paul Lajoie-Mazenc, Nicolas Prigent, and V Viet Triem Tong. A privacy preserving distributed reputation mechanism. In *Communications (ICC), 2013 IEEE International Conference on*, pages 1951–1956. IEEE, 2013.

[13] Roberto Aringhieri, Ernesto Damiani, Sabine De Capitani Di Vimercati, Stefano Paraboschi, and Pierangelo Samarati. Fuzzy techniques for trust and reputation management in anonymous peer-to-peer systems. *Journal of the American Society for Information Science and Technology*, 57(4):528–537, 2006.

[14] Marianne A Azer, Sherif M El-Kassas, Abdel Wahab F Hassan, and Magdy S El-Soudani. A survey on trust and reputation schemes in ad hoc networks. In *2008 Third International Conference on Availability, Reliability and Security*, pages 881–886. IEEE, 2008.

[15] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 719–737. Springer, 2012.

[16] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway. Efficient garbling from a fixed-key blockcipher. *in IEEE SP, pp. 478-492*, 2013.

[17] Assaf Ben-David, Noam Nisan, and Benny Pinkas. Fairplaymp: a system for secure multi-party computation. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 257–266. ACM, 2008.

[18] I. F. Blake and V. Kolesnikov. Strong conditional oblivious transfer and computing on intervals. *In Advances in Cryptology - ASIACRYPT'04, volume 3329 of LNCS, pp. 515-529. Springer*, 2004.

[19] I. F. Blake and V. Kolesnikov. One-round secure comparison of integers. *Journal of Mathematical Cryptography, vol. 3, no. 1*, 2009.

[20] George Robert Blakley. Safeguarding cryptographic keys. In *1979 International Workshop on Managing Requirements Knowledge (MARK)*, pages 313–318. IEEE, 1979.

[21] Avrim Blum, Merrick Furst, Michael Kearns, and Richard J Lipton. Cryptographic primitives based on hard learning problems. In *Annual International Cryptology Conference*, pages 278–291. Springer, 1993.

[22] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM)*, 50(4):506–519, 2003.

[23] Dan Bogdanov, Sven Laur, and Jan Willemson. Sharemind: A framework for fast privacy-preserving computations. In *European Symposium on Research in Computer Security*, pages 192–206. Springer, 2008.

[24] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser. Machine learning classification over encrypted data. *NDSS*, 2015.

[25] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. In *NDSS*, volume 4324, page 4325, 2015.

[26] F. Boudot, B. Schoenmakers, and J. Traore. A fair and efficient solution to the socialist millionaires' problem. *to appear in Discrete Applied Mathematics, Special Issue on Coding and Cryptography, Elsevier*, 2000.

[27] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector ole. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 896–912, 2018.

[28] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent ot extension and more. In *Annual International Cryptology Conference*, pages 489–518. Springer, 2019.

[29] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators from ring-lpn. In *Annual International Cryptology Conference*, pages 387–416. Springer, 2020.

[30] Lennart Braun, Daniel Demmler, Thomas Schneider, and Oleksandr Tkachenko. Motion–a framework for mixed-protocol multi-party computation. *ACM Transactions on Privacy and Security*, 25(2):1–35, 2022.

[31] Martin Burkhart, Mario Strasser, Dilip Many, and Xenofontas Dimitropoulos. Sepia: Privacy-preserving aggregation of multi-domain network events and statistics. *Network*, 1(101101), 2010.

[32] C. Cachin. Efcient private bidding and auctions with an oblivious third party. *In: Proceedings of the 6th ACM conference on computer and communications security. ACM, pp 120-127*, 1999.

[33] O. Catrina and S. de Hoogh. Improved primitives for secure multiparty integer computation. *In Security and Cryptography for Networks (SCN), pages 182-199*, 2010.

[34] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient homomorphic conversion between (ring) lwe ciphertexts. In *International Conference on Applied Cryptography and Network Security*, pages 460–479. Springer, 2021.

[35] Hao Chen, Ran Gilad-Bachrach, Kyoohyung Han, Zhicong Huang, Amir Jalali, Kim Laine, and Kristin Lauter. Logistic regression over encrypted data from fully homomorphic encryption. *BMC medical genomics*, 11(4):81, 2018.

[36] Hao Chen, Kim Laine, and Rachel Player. Simple encrypted arithmetic library-seal v2. 1. In *International Conference on Financial Cryptography and Data Security*, pages 3–18. Springer, 2017.

[37] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster packed homomorphic operations and efficient circuit bootstrapping for tfhe. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 377–408. Springer, 2017.

[38] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Tfhe: fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, 2020.

[39] Ilaria Chillotti, Marc Joye, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. Concrete: Concrete operates on ciphertexts rapidly by extending tfhe. In *WAHC 2020–8th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, volume 15, 2020.

[40] Jin-Hee Cho, Kevin Chan, and Sibel Adali. A survey on trust modeling. *ACM Computing Surveys (CSUR)*, 48(2):1–40, 2015.

[41] Jin-Hee Cho, Ananthram Swami, and Ray Chen. A survey on trust management for mobile ad hoc networks. *IEEE Communications Surveys & Tutorials*, 13(4):562–583, 2010.

[42] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, pages 383–395. IEEE, 1985.

[43] Michael R Clark, Kyle Stewart, and Kenneth M Hopkinson. Dynamic, privacy-preserving decentralized reputation systems. *IEEE Transactions on Mobile Computing*, 16(9):2506–2517, 2017.

[44] G. Couteau. New protocols for secure equality test and comparison. *n: Preneel B., Vercauteren F. (eds) Applied Cryptography and Network Security. ACNS 2018. Lecture Notes in Computer Science, vol 10892. Springer, Cham*, 2018.

[45] Giovanni Di Crescenzo. Private selective payment protocols. *Financial Cryptography and Data Security, FC 2000, LNCS 1962, pp. 72-89. Springer-Verlag*, 2000.

[46] Giovanni Di Crescenzo, Rafail Ostrovsky, and Sivaramakrishnan Rajagopalan. Conditional oblivious transfer and time-released encryption. *Advances in Cryptology - CRYPTO 99, LNCS 1592, pp. 74-89. Springer-Verlag*, 1999.

[47] I. Damgard, M. Fitzi, E. Kiltz, J.B. Nielsen, and T. Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. *In Proc. 3rd Theory of Cryptography Conference, TCC 2006, volume 3876 of Lecture Notes in Computer Science, pages 285-304, Berlin, Springer-Verlag*, 2006.

[48] I. Damgard, M. Geisler, and M. Kroigard. Efficient and secure comparison for on-line auctions. *In Australasian Conference on Information Security and Privacy (ACISP'07), volume 4586 of LNCS, pages 416-430. Springer*, 2007.

[49] I. Damgard, M. Geisler, and M. Kroigard. A correction to efficient and secure comparison for on-line auctions. *International Journal of Applied Cryptography, 1(4):323-324*, 2009.

[50] Ivan Damgård, Martin Geisler, Mikkel Krøigaard, and Jesper Buus Nielsen. Asynchronous multiparty computation: Theory and implementation. In *International Workshop on Public Key Cryptography*, pages 160–179. Springer, 2009.

[51] Ivan Damgard, Martin Geisler, and Mikkel Kroigard. Homomorphic encryption and secure comparison. *International Journal of Applied Cryptography*, 1(1):22–31, 2008.

[52] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P Smart. Practical covertly secure mpc for dishonest majority–or: breaking the spdz limits. In *European Symposium on Research in Computer Security*, pages 1–18. Springer, 2013.

[53] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Annual Cryptology Conference*, pages 643–662. Springer, 2012.

[54] Bernardo David, Rafael Dowsley, Raj Katti, and Anderson CA Nascimento. Efficient unconditionally secure comparison and privacy preserving machine learning classification protocols. In *International Conference on Provable Security*, pages 354–367. Springer, 2015.

[55] Daniel Demmler, Thomas Schneider, and Michael Zohner. Aby-a framework for efficient mixed-protocol secure two-party computation. In *NDSS*, 2015.

[56] Arthur P Dempster. A generalization of bayesian inference. *Journal of the Royal Statistical Society: Series B (Methodological)*, 30(2):205–232, 1968.

[57] Wenliang Du and Mikhail J Atallah. Privacy-preserving cooperative scientific computations. In *csfw*, page 0273. Citeseer, 2001.

[58] Léo Ducas and Daniele Micciancio. Fhew: bootstrapping homomorphic encryption in less than a second. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 617–640. Springer, 2015.

[59] Yael Ejgenberg, Moriya Farbstein, Meital Levy, and Yehuda Lindell. Scapi: The secure computation application programming interface. *IACR Cryptology EPrint Archive*, 2012:629, 2012.

[60] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

[61] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft. Privacy-preserving face recognition. *in Privacy Enhancing Technologies, pp. 235-253*, 2009.

[62] Uri Feige, Joe Killian, and Moni Naor. A minimal model for secure computation. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 554–563. ACM, 1994.

[63] M. Fischlin. A cost-effective pay-per-multiplication comparison method for millionaires. *Cryptographers track at the RSA conference. Springer, Berlin, pp 457-471*, 2001.

[64] Mohammad G Raeini and Mehrdad Nojoumian. Privacy-preserving big data analytics: from theory to practice. In *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*, pages 45–59. Springer, 2019.

[65] J. A. Garay, B. Schoenmakers, and J. Villegas. Practical and secure solutions for integer comparison. *In Public Key Cryptography (PKC'07), volume 4450 of LNCS, pages 330-342. Springer*, 2007.

[66] Craig Gentry et al. Fully homomorphic encryption using ideal lattices. In *Stoc*, volume 9, pages 169–178, 2009.

[67] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Annual Cryptology Conference*, pages 75–92. Springer, 2013.

[68] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game - a completeness theorem for protocols with honest majority. *In 19th STOC, pages 218-229*, 1987.

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

[70] Jean Gordon and Edward H Shortliffe. The dempster-shafer theory of evidence. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, 3:832–838, 1984.

[71] Tyrone Grandison and Morris Sloman. A survey of trust in internet applications. *IEEE Communications Surveys & Tutorials*, 3(4):2–16, 2000.

[72] D. Grigoriev and V. Shpilrain. Yao's millionaires' problem and decoy-based public key encryption by classical physics. *J. Foundations Comp. Sci. 25, 409-417*, 2014.

[73] Ehud Gudes, Nurit Gal-Oz, and Alon Grubshtein. Methods for computing trust and reputation while preserving privacy. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 291–298. Springer, 2009.

[74] Jia Guo, Ray Chen, and Jeffrey JP Tsai. A survey of trust computation models for service management in internet of things systems. *Computer Communications*, 97:1–14, 2017.

[75] Guangjie Han, Jinfang Jiang, Lei Shu, Jianwei Niu, and Han-Chieh Chao. Management and applications of trust in wireless sensor networks: A survey. *Journal of Computer and System Sciences*, 80(3):602–617, 2014.

[76] Omar Hasan. *Privacy preserving reputation systems for decentralized environments*. PhD thesis, Lyon, INSA, 2010.

[77] Omar Hasan, Lionel Brunie, and Elisa Bertino. Preserving privacy of feedback providers in decentralized reputation systems. *Computers & Security*, 31(7):816–826, 2012.

[78] Omar Hasan, Lionel Brunie, Elisa Bertino, and Ning Shang. A decentralized privacy preserving reputation protocol for the malicious adversarial model. *IEEE Transactions on Information Forensics and Security*, 8(6):949–962, 2013.

[79] Marcella Hastings, Brett Hemenway, Daniel Noble, and Steve Zdancewic. Sok: General purpose compilers for secure multi-party computation. In *2019 IEEE symposium on security and privacy (SP)*, pages 1220–1237. IEEE, 2019.

[80] Nolan Hedglin, Kade Phillips, and Andrew Reilley. Building a fully homomorphic encryption scheme in python. 2019.

[81] Ferry Hendrikx, Kris Bubendorfer, and Ryan Chard. Reputation systems: A survey and taxonomy. *Journal of Parallel and Distributed Computing*, 75:184–197, 2015.

[82] Wilko Henecka, Ahmad-Reza Sadeghi, Thomas Schneider, Immo Wehrenberg, et al. Tasty: tool for automating secure two-party computations. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 451–462. ACM, 2010.

[83] Stefan Heyse, Eike Kiltz, Vadim Lyubashevsky, Christof Paar, and Krzysztof Pietrzak. Lapin: an efficient authentication protocol based on ring-lpn. In *International Workshop on Fast Software Encryption*, pages 346–365. Springer, 2012.

[84] Jingwei Huang and David M Nicol. Trust mechanisms for cloud computing. *Journal of Cloud Computing: Advances, Systems and Applications*, 2(1):9, 2013.

[85] I. Ioannidis and A. Grama. An efcient protocol for yao's millionaires problem. *Proceedings of the 36th Hawaii international conference on system science. IEEE Press, Piscataway, pp. 1-6*, 2003.

[86] M. Jakobsson and M. Yung. Proving without knowing: On oblivious, agnostic and blindfolded provers. *Crypto 96, Lecture Notes in Computer Science, Vol. 1109, Springer-Verlag, pp. 186-200*, 1996.

[87] Leqi Jiang, Yi Cao, Chengsheng Yuan, Xingming Sun, and Xiaoli Zhu. An effective comparison protocol over encrypted data in cloud computing. *Journal of Information Security and Applications*, 48:102367, 2019.

[88] Audun Josang. An algebra for assessing trust in certification chains. In *Proceedings of the Network and Distributed Systems Security Symposium (NDSS99). The Internet Society*, 1999.

[89] Audun Jøsang. A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(03):279–311, 2001.

[90] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision support systems*, 43(2):618–644, 2007.

[91] Audun Jsang and Roslan Ismail. The beta reputation system. *Proceedings of the 15th bled electronic commerce conference. Vol. 5*, pages 2502–2511, 2002.

[92] Sepandar D Kamvar, Mario T Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 640–651, 2003.

[93] Marcel Keller. MP-SPDZ: A versatile framework for multi-party computation. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pages 1575–1590, 2020.

[94] Florian Kerschbaum. A verifiable, centralized, coercion-free reputation system. In *Proceedings of the 8th ACM workshop on Privacy in the electronic society*, pages 61–70. ACM, 2009.

[95] Florian Kerschbaum and Orestis Terzidis. Filtering for private collaborative benchmarking. *in International Conference on Emerging trends in information and communication security*, 2006.

[96] V. Kolesnikov, A.-R. Sadeghi, and T. Schneider. How to combine homomorphic encryption and garbled circuits - improved circuits and computing the minimum distance efficiently. *in 1st International Workshop on Signal Processing in the EncryptEd Domain (SPEED'09)*, 2009.

[97] V. Kolesnikov, A. R. Sadeghi, and T. Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. *In CANS 09, LNCS 5888, pages 1-20. Springer*, 2009.

[98] V. Kolesnikov and T. Schneider. Improved garbled circuit: Free xor gates and applications. *In ICALP 2008, Part II, LNCS 5126, pages 486-498. Springer*, 2008.

[99] Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In *International Conference on Cryptology and Network Security*, pages 1–20. Springer, 2009.

[100] P Suresh Kumar, P Sateesh Kumar, and S Ramachandram. Recent trust models in grid. 2011.

[101] Qiqi Lai, Feng-Hao Liu, and Zhedong Wang. Almost tight security in lattices with polynomial moduli–prf, ibe, all-but-many ltf, and more. In *IACR International Conference on Public-Key Cryptography*, pages 652–681. Springer, 2020.

[102] Éric Levieil and Pierre-Alain Fouque. An improved lpn algorithm. In *International conference on security and cryptography for networks*, pages 348–359. Springer, 2006.

[103] S. Li, Y. Guo, S. Zhou, J. Dou, and D. Wang. Efficient protocols for the general millionaires' problem. *Chin. J. Electron., vol. 26, no. 4, pp. 696-702*, 2017.

[104] H. Y. Lin and W.G. Tzeng. An efficient solution to the millionaires problem based on homomorphic encryption. *In: International conference on applied cryptography and network security, Springer, Berlin, pp. 456-466*, 2005.

[105] Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. In *Annual International Cryptology Conference*, pages 36–54. Springer, 2000.

[106] Jinshan Liu and Valérie Issarny. Enhanced reputation mechanism for mobile ad hoc networks. In *International Conference on Trust Management*, pages 48–62. Springer, 2004.

[107] X. Liu, S. Li, X. Chen, G. Xu, X. Zhang, , and Y. Zhou. Efficient solutions to two-party and multiparty millionaires problem. *Secur. Commun. Netw.*, 2017.

[108] X. Liu, S. Li, J. Liu, X. Chen, and G. Xu. Secure multiparty computation of a comparison problem. *SpringerPlus, vol. 5, no. 1, p. 1489*, 2016.

[109] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 1–23. Springer, 2010.

[110] Shuo Ma, Ouri Wolfson, and Jie Lin. A survey on trust management for intelligent transportation system. In *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Computational Transportation Science*, pages 18–23, 2011.

[111] A. Maitra, G. Paul, and A. K. Pal. Millionaires problem with rational players: a unified approach in classical and quantum paradigms. *In: arXiv preprint arXiv:1504.01974, https://arxiv.org/abs/1504.01974*, 2015.

[112] Dahlia Malkhi, Noam Nisan, Benny Pinkas, Yaron Sella, et al. Fairplay-secure two-party computation system. In *USENIX Security Symposium*, volume 4, page 9. San Diego, CA, USA, 2004.

[113] Daniel W Manchala. Trust metrics, models and protocols for electronic commerce transactions. In *Proceedings. 18th International Conference on Distributed Computing Systems (Cat. No. 98CB36183)*, pages 312–321. IEEE, 1998.

[114] Stephen Paul Marsh. Formalising trust as a computational concept. 1994.

[115] RJ McEliece and DV Sarwate. On sharing secrets and reed-solomon codes. *Communications of the ACM*, 24(9):583–584, 1981.

[116] Silvio Micali and Phillip Rogaway. Secure computation. In *Annual International Cryptology Conference*, pages 392–404. Springer, 1991.

[117] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 700–718. Springer, 2012.

[118] Daniele Micciancio and Yuriy Polyakov. Bootstrapping in fhew-like cryptosystems. In *Proceedings of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pages 17–28, 2021.

[119] Michele Minelli. *Fully homomorphic encryption for machine learning.* PhD thesis, PSL Research University, 2018.

[120] Benjamin Mood, Debayan Gupta, Henry Carter, Kevin Butler, and Patrick Traynor. Frigate: A validated, extensible, and efficient compiler and interpreter for secure computation. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 112–127. IEEE, 2016.

[121] Lik Mui, Mojdeh Mohtashemi, and Ari Halberstadt. A computational model of trust and reputation. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, pages 2431–2439. IEEE, 2002.

[122] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. *in Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms, pp. 448-457*, 2001.

[123] Takashi Nishide and Kazuo Ohta. Constant-round multiparty computation for interval test, equality test, and comparison. *IEICE Transactions 90-A (2007), pp. 960-968*, 2007.

[124] Takashi Nishide and Kazuo Ohta. Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. *In PKC 2007, volume 4450 of LNCS, pages 343-360, Springer*, 2007.

[125] Rishab Nithyanand and Karthik Raman. Fuzzy privacy preserving peer-to-peer reputation management. *IACR Cryptology ePrint Archive*, 2009:442, 2009.

[126] Mehrdad Nojoumian and Timothy C Lethbridge. A new approach for the trust calculation in social networks. In *International Conference on E-Business and Telecommunication Networks*, pages 64–77. Springer, 2006.

[127] Mehrdad Nojoumian and Douglas R. Stinson. Unconditionally secure first-price auction protocols using a multicomponent commitment scheme. In *12th International Conference on Information and Communications Security (ICICS)*, volume 6476 of *LNCS*, pages 266–280. Springer, 2010.

[128] Mehrdad Nojoumian and Douglas R. Stinson. On dealer-free dynamic threshold schemes. *Advances in Mathematics of Communications (AMC)*, 7(1):39–56, 2013.

[129] Mehrdad Nojoumian and Douglas R. Stinson. Efficient sealed-bid auction protocols using verifiable secret sharing. In *10th International Conference on Information Security Practice and Experience, ISPEC'14*, volume 8434 of *LNCS*, pages 302–317. Springer, 2014.

[130] Henri Nussbaumer. Fast polynomial transform algorithms for digital convolution. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(2):205–215, 1980.

[131] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 223–238. Springer, 1999.

[132] Elan Pavlov, Jeffrey S Rosenschein, and Zvi Topol. Supporting privacy in decentralized additive reputation systems. In *International Conference on Trust Management*, pages 108–119. Springer, 2004.

[133] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual International Cryptology Conference*, pages 129–140. Springer, 1991.

[134] Krzysztof Pietrzak. Cryptography from learning parity with noise. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 99–114. Springer, 2012.

[135] John M Pollard. The fast fourier transform in a finite field. *Mathematics of computation*, 25(114):365–374, 1971.

[136] Luis Bernardo Pulido-Gaytan, Andrei Tchernykh, Jorge M Cortés-Mendoza, Mikhail Babenko, and Gleb Radchenko. A survey on privacy-preserving machine learning with fully homomorphic encryption. In *Latin American High Performance Computing Conference*, pages 115–129. Springer, 2020.

[137] Mohammad G Raeini and Mehrdad Nojoumian. Secure trust evaluation using multipath and referral chain methods. In *International Workshop on Security and Trust Management*, pages 124–139. Springer, 2019.

[138] Lars Rasmusson and Sverker Jansson. Simulated social control for secure internet commerce (position paper). 1996.

[139] Aseem Rastogi, Matthew A Hammer, and Michael Hicks. Wysteria: A programming language for generic, mixed-mode multiparty computations. In *2014 IEEE Symposium on Security and Privacy*, pages 655–670. IEEE, 2014.

[140] Aseem Rastogi, Nikhil Swamy, and Michael Hicks. Wys*: : A dsl for verified secure multi-party computations. In *International Conference on Principles of Security and Trust*, pages 99–122. Springer, 2019.

[141] Irving S Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.

[142] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.

[143] T. Reistad. Multiparty comparison - an improved multiparty protocol for comparison of secret-shared values. *In International Conference on Security and Cryptography (SECRYPT), pages 325-330*, 2009.

[144] Paul Resnick and Richard Zeckhauser. Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system. In *The Economics of the Internet and E-commerce*, pages 127–157. Emerald Group Publishing Limited, 2002.

[145] M Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M Songhori, Thomas Schneider, and Farinaz Koushanfar. Chameleon: A hybrid secure computation framework for machine learning applications. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 707–721. ACM, 2018.

[146] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[147] Alexander Schaub, Rémi Bazin, Omar Hasan, and Lionel Brunie. A trustless privacy-preserving reputation system. In *IFIP International Information Security and Privacy Conference*, pages 398–411. Springer, 2016.

[148] B. Schoenmakers and P. Tuyls. Practical two-party computation based on the conditional gate. *In Advances in Cryptology-ASIACRYPT '04, volume 3329 of Lecture Notes in Computer Science, pages 119-136, Berlin, Springer-Verlag*, 2004.

[149] Glenn Shafer. *A mathematical theory of evidence*, volume 42. Princeton university press, 1976.

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

[151] Wanita Sherchan, Surya Nepal, and Cecile Paris. A survey of trust in social networks. *ACM Computing Surveys (CSUR)*, 45(4):1–33, 2013.

[152] L. Shundong, W. Daoshun, D. Yiqi, and L. Ping. Symmetric cryptographic solution to yao's millionaires' problem and an evaluation of secure multiparty computations. *Information Sciences, Vol.178, No.1, pp. 244-255*, 2008.

[153] Patrícia R Sousa, Luís Antunes, and Rolando Martins. The present and future of privacy-preserving computation in fog computing. In *Fog Computing in the Internet of Things*, pages 51–69. Springer, 2018.

[154] Springer. *Secret sharing comparison by transformation and rotation*, 2007.

[155] Ion Stoica, Robert Morris, David Karger, M Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review*, 31(4):149–160, 2001.

[156] Yan Lindsay Sun, Wei Yu, Zhu Han, and KJ Ray Liu. Information theoretic framework of trust modeling and evaluation for ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 24(2):305–317, 2006.

[157] T. Toft. Sub-linear, secure comparison with two non-colluding parties. *In PKC 2011, LNCS 6571, pages 174-191, Springer*, 2011.

[158] T. Veugen. Improving the DGK comparison protocol. *In Information Forensics and Security (WIFS), 2012 IEEE International Workshop on, pages 49-54. IEEE*, 2012.

[159] T. Veugen, F. Blom, S.J. de Hoogh, and Z. Erkin. Secure comparison protocols in the semi-honest model. *IEEE Journal of Selected Topics in Signal Processing 9(7) (2015) 1217-1228*, 2015.

[160] Thijs Veugen. Comparing encrypted data. *Multimedia Signal Processing Group, Delft University of Technology, The Netherlands, and TNO Information and Communication Technology, Delft, The Netherlands, Tech. Rep*, 2011.

[161] Alexander Viand, Patrick Jattke, and Anwar Hithnawi. Sok: Fully homomorphic encryption compilers. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1092–1108. IEEE, 2021.

[162] Nikolaj Volgushev, Malte Schwarzkopf, Ben Getchell, Mayank Varia, Andrei Lapets, and Azer Bestavros. Conclave: secure multi-party computation on big data. In *Proceedings of the Fourteenth EuroSys Conference 2019*, page 3. ACM, 2019.

[163] Yao Wang and Julita Vassileva. Trust and reputation model in peer-to-peer networks. In *Proceedings Third International Conference on Peer-to-Peer Computing (P2P2003)*, pages 150–157. IEEE, 2003.

[164] Alexander Wood, Kayvan Najarian, and Delaram Kahrobaei. Homomorphic encryption for machine learning in medicine and bioinformatics. *ACM Computing Surveys (CSUR)*, 53(4):1–35, 2020.

[165] Ronald R Yager. On the dempster-shafer framework and new combination rules. *Information sciences*, 41(2):93–137, 1987.

[166] Zheng Yan, Wenxiu Ding, Valtteri Niemi, and Athanasios V Vasilakos. Two schemes of privacy-preserving trust evaluation. *Future Generation Computer Systems*, 62:175–189, 2016.

[167] Zheng Yan, Peng Zhang, and Athanasios V Vasilakos. A survey on trust management for internet of things. *Journal of network and computer applications*, 42:120–134, 2014.

[168] A. Yao. Protocols for secure computation. *In Proc. 23rd Annual Symp. on Foundations of Computer Science (FOCS), pages 160-164. IEEE*, 1982.

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

[170] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 162–167. IEEE, 1986.

[171] Bin Yu and Munindar P Singh. A social mechanism of reputation management in electronic communities. In *International Workshop on Cooperative Information Agents*, pages 154–165. Springer, 2000.

[172] Bin Yu and Munindar P Singh. An evidential model of distributed reputation management. In *Proceedings of the first international joint conference on Autonomous Agents and Multiagent Systems: Part 1*, pages 294–301. ACM, 2002.

[173] Ching-Hua Yu. Sign modules in secure arithmetic circuits. *IACR Cryptology ePrint Archive*, 2011:539, 2011.

[174] S. Zahur, M. Rosulek, and D. Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. *LNCS, pages 220-250. Springer*, 2015.

[175] Samee Zahur and David Evans. Obliv-c: A language for extensible data-oblivious computation. *IACR Cryptology ePrint Archive*, 2015.

[176] Jie Zhang. A survey on trust management for vanets. In *2011 IEEE International Conference on Advanced Information Networking and Applications*, pages 105–112. IEEE, 2011.

[177] Jingan Zhang and Xiane Guo. Trust evaluation model based on fuzzy logic for c2c e-commerce. In *2009 International Symposium on Information Engineering and Electronic Commerce*, pages 403–407. IEEE, 2009.

[178] Yihua Zhang, Aaron Steele, and Marina Blanton. Picco: a general-purpose compiler for private distributed computation. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 813–826. ACM, 2013.

[179] Guy Zyskind, Oz Nathan, et al. Decentralizing privacy: Using blockchain to protect personal data. In *Security and Privacy Workshops (SPW), 2015 IEEE*, pages 180–184. IEEE, 2015.

[180] Guy Zyskind, Oz Nathan, and Alex Pentland. Enigma: Decentralized computation platform with guaranteed privacy. *arXiv preprint arXiv:1506.03471*, 2015.