ADVANCEMENTS IN PUBLIC-KEY CRYPTOGRAPHY: CRAFTING NOVEL CONSTRUCTIONS TO ADDRESS EMERGING DEMANDS

BORUI GONG

PhD

The Hong Kong Polytechnic University

2023

The Hong Kong Polytechnic University

Department of Computing

Advancements in Public-Key Cryptography: Crafting Novel Constructions to Address Emerging Demands

Borui Gong

A thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

May 2023

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

Borui Gong (Name of student)

Abstract

Since its emergence as a formal science in the 1970s, modern cryptography has experienced remarkable progress. A notable milestone is the introduction of public-key cryptography by Diffie and Hellman in 1976, which has revolutionized secure communication and computation. It is now an indispensable aspect of our digital life, providing a range of security and privacy solutions.

Modern public-key schemes are designed to meet diverse functionality and security demands, and must be rigorously validated within security models that accurately reflect real-world attack scenarios. However, with the continual advancement of computer science and its widespread applications, there is a dual effect: it provides personalized services and convenience, but also brings new challenges in security and functionality for existing public-key cryptographic systems. For example, today's growing complexity in interaction and deployment environments enables adversaries to gain additional information and launch novel attacks on existing protocols. Therefore, it becomes essential to develop enhanced security models that consider the influence of these additional entities. Furthermore, the extensive collection and use of personal information by various companies and organizations, aimed at improving service quality and convenience, raise critical security and privacy challenges when making use of sensitive and distributed

data.

This dissertation focuses on making public-key cryptography more practical in the face of functionality and security challenges raised in real-world applications. At the same time, we assess the overheads associated with integrating cryptographic protocols into systems, ensuring efficient deployment in practical settings. Our research concentrates on three representative areas of public-key cryptography: digital signatures, zero-knowledge proofs, and blockchain applications.

In more detail, we address new security demands by investigating enhanced models within the context of strong designated verifier signature (SDVS) schemes and propose a generic framework that meets these enhanced models. Addressing functionality demands in two-party data analysis, we introduce a zero-knowledge argument of knowledge protocol for the Paillier cryptosystem, offering active security in data aggregation. Lastly, we explore the development of a fully decentralized electronic voting system, integrating blockchain technology and other public-key primitives to reduce dependency on trust and ensure comprehensive security and functionality.

More specifically, we present the following results.

- We introduce two enhanced models in strong designated verifier signatures that account for potential security influences from more entities, namely, multi-user and multi-user$^+$. We also provide a generic construction utilizing a

key encapsulation mechanism and a pseudorandom function, proving its security under our new models. Additionally, we offer several instantiations. Each is based on different security assumptions, allowing us to achieve distinct characteristics. Furthermore, diverse key encapsulation mechanisms can be employed to tailor SDVS schemes to specific needs.

- We propose an efficient zero-knowledge argument of knowledge system for the Paillier cryptosystem. Our system features sub-linear proof size, low verification cost, and manageable proof time, while also supporting batch proof generation and verification. We instantiate our system in various scenarios and conduct comprehensive experiments to assess its practicality. Scenario 1 is Paillier with packing. When we pack 25.6K bits into 400 ciphertexts, a proof that all these ciphertexts are correctly computed is 17 times smaller and is 3 times faster to verify compared with the naive implementation: using 25.6K OR-proofs without packing. Furthermore, we can prove additional statements almost for free, e.g., one can prove that the sum of a subset of the witness bits is less than a threshold t. Another scenario is range proof. To prove that each plaintext in 200 Paillier ciphertexts is of size 256 bits, our proof size is 10 times smaller than the state-of-the-art. Results demonstrate that

our system is asymptotically more efficient than the state-of-the-art and is particularly well-suited for situations involving a large number (over 100) of Paillier ciphertexts, which frequently occur in real-world applications.

- We present an electronic voting system based on blockchain technology that features fully distributed authorities. To distribute trust in the registration process, we employ threshold blind signatures while maintaining the anonymity of the voters. We also utilize a threshold decryption scheme to distribute authorities in the tallying phase. By integrating these techniques with using a blockchain as the public bulletin board, our system attains verifiability, eligibility, fairness, and anonymity properties. We also implement our system to evaluate its efficiency and overall performance. Our experimental results show that our proposed system is efficient enough for real deployment while maintaining the common security guarantees required in an e-voting system.

Publications Arising from the Thesis

- **Borui Gong**, Man Ho Au, Haiyang Xue. Constructing Strong Designated Verifier Signatures from Key Encapsulation Mechanisms. In 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), Rotorua, New Zealand, 2019, pages 586-593.

- **Borui Gong**, Xingye Lu, Lau Wang Fat, Man Ho Au. Blockchain-Based Threshold Electronic Voting System. In Security and Privacy in Social Networks and Big Data (SocialSec 2019), pages 238–250.

- **Borui Gong**, Wang Fat Lau, Man Ho Au, Rupeng Yang, Haiyang Xue, Lichun Li. Efficient Zero-Knowledge Arguments For Paillier Cryptosystem. (To appear in *S&P* 2024).

Acknowledgements

My Ph.D. journey at PolyU has been a winding path filled with contemplation, self-doubt, reading, discussions, and hard work. Fortunately, I encountered numerous incredible individuals throughout this journey, making it an unforgettable experience. This thesis would not have been possible without the love, support, guidance, inspiration, encouragement, and companionship of countless people, to whom I would like to express my deepest gratitude and sincerest thanks.

First and foremost, I would like to express my deepest gratitude to my primary Ph.D. advisor, Prof. Au Man Ho Allen, for his exceptional guidance, unwavering patience, and enthusiastic encouragement throughout my Ph.D. journey. I truly appreciate every instance in which he has engaged in research discussions with me, regardless of the hour. His dedication to understanding the fundamental knowledge underlying each question has profoundly influenced me, and these qualities will undoubtedly benefit me for a lifetime. He has consistently been supportive, teaching me how to articulate questions clearly, providing insightful suggestions, allowing me the freedom to work on intriguing projects without pressure, and offering opportunities to attend conferences, winter schools, and visits. Throughout this journey, I have learned a great deal from Allen – from writing research papers and delivering presentations to conducting scientific research, building research and development

teams, and much more in life. Allen has been, and will continue to be, my role model as a researcher, advisor, and colleague in the pursuit of creating truly impactful research contributions for the community.

I would like to extend my gratitude to my Ph.D. co-advisor, Prof. Luo Xiapu Daniel. I appreciate all the support and opportunities that he has provided during my Ph.D. study at PolyU. I would also like to appreciate my TA's supervisor, Prof. Liu Yan Wang Dennis, for his supervision in teaching and communication. Furthermore, I would like to take this opportunity to thank Prof. Zhang Fengwei , for offering me the opportunity to experience a fantastic visiting period at the Southern University of Science and Technology. I am grateful for his outstanding mentorship on conducting world-class research and for providing an exceptional study environment on campus.

I would also like to express my appreciation to my thesis committee members, Prof. Chen Jie, Prof. Wei Puwen and Prof. Wu Xiaoming for their time and efforts in providing valuable and constructive comments on my thesis. Their thoughtful feedback has greatly improved the quality of this dissertation.

Furthermore, I wish to take this opportunity to extend my thanks to all the mentors, co-authors, colleagues and lab-mates, who directly or indirectly contributed to this thesis. I would like to thank: Zuoxia Yu, Xingye Lu, Haiyang Xue, Rupeng Yang, Wang Fat Lau, Mengling Liu, Chengru Zhang, Xiao Yang, Kang Li, Jiazhuo Lyu, Xiaoyi Yang, Jinghui Liao, Jingjing Fan,

Yilei Wang, On Na Cheng, and many others. I still remember the time that we carefully discuss research problems, the time that I ask some even "stupid" and naive questions, and the time that we hang out and share wonderful food together. It was a great pleasure to meet all of you during this journey. In addition, as I have taken a visiting at the Southern University of Science and Technology, I would also like to give my thanks to all the professors, colleagues, and lab-mates that I've met there. Thanks to Xian Qing, Nian Liu, Qingxia Li, Jinting Wu, Yiming Zhang, Kun Lu, Haroon, Zhanbo Wang, Lei Zhou, Yu Wang, Yunjie Deng , Jiaxin Zhan, and many others for all the memorable happiness, help, and discussions that happen in the Lab 441A. Wishing the COMPASS lab be better and all of you a bright future.

Moreover, beyond the realm of research, I would like to express my gratitude to my friends, including Ling Zhu, Jiayue Sun, Yujing Luan, Chenlu Wei, and so on, who have consistently provided me with happiness, positivity, and support throughout both good and challenging times. Furthermore, I offer special thanks to Kaini Zhou and Feiyu Wang, who have taught me the importance of having a friend by my side. Although we cannot always meet in person, it is always a joyous and gratifying experience when we connect online to discuss and share our recent lives. I send my best wishes to all of you and hope that we will remain in each other's lives for many years to come, even when we are unable to walk smoothly. I also want

to thank Hao Xu for the companionship and driving me home when I'm struggling for the DDLs.

Last but not least, I would like to extend my heartfelt thanks to my whole family. I am incredibly fortunate to receive so much unconditional support and unwavering love from the best parents in the world! I am grateful for their encouragement and advice whenever I encounter problems and for their unwavering support in every decision I have made. I would also like to express my gratitude to my grandparents, who always treat me like a little girl, eagerly anticipate seeing me, and make the most special and delicious homemade comfort food for me. Their love makes me feel secure and never alone. Furthermore, I want to thank my aunt, uncle, and sister for all the happy times we have shared together. A special thanks goes to Niuniu, who is my first, best, and dearest fluffy friend! I consider her to be like a little sister to me. Thanks to her companionship, happiness, and even mischievousness that she brings to my life. Niuniu is so smart that she listens to me, understands what I'm saying, and even comforts me when I'm feeling down. Having Niuniu in my life is one of the best decisions I have ever made. I wish her all the best and hope that we can continue to be there for each other for the next ten years. Finally, I want to thank my other furry friends, including Doudou, Taotao, Oreo, Zhima, Danhuang, Mocha, and the stray cats that I have fed, such as Dangdang, Maomao, Dahuang, and Naonao. Spending time with them is so much

easier than with any two-legged creature. They always provide me with happiness and comfort, as long as I feed them.

Table of Contents

Abstract **iii**

Publications Arising from the Thesis **vii**

Acknowledgements **ix**

List of Figures **xix**

List of Tables **xxi**

1 Introduction **1**

1.1 Generic Constructions for Strong Designated Verifier Signature Schemes 21

1.2 Efficient Zero-knowledge Argument of Knowledge for Paillier 24

1.3 Distributed Electronic Voting in Blockchain . . . 26

1.4 Related Works 27

1.5 Thesis Organization and Derived Publications . 39

2 Preliminaries **41**

2.1 Notations and Cryptographic Assumptions . . . 41

2.1.1 DL Assumption on Composite Group . . 41

2.1.2 DCR Assumption 42

2.1.3 GDH Groups 43
2.2 Key Encapsulation Mechanism 43
2.3 Pseudorandom Function 45
2.4 Designated Verifier Signature 46
2.5 Pedersen Commitment 47
2.6 Paillier Encryption 48
2.7 Zero-knowledge Argument of Knowledge (ZKAoK) 49
2.8 Threshold Blind Signature 52
2.9 Threshold ElGamal Decryption Scheme 55

3 Strong Designated Verfiable Signature from Key Encapsulation Mechanism 59
3.1 Our Contribution 61
3.2 Our Strengthened Models 62
3.3 Our Construction 66
3.3.1 Overview of Our Construction 67
3.3.2 Details of Our Generic Construction . . . 67
3.4 Security Analysis of Our Construction 69
3.5 Instantiation and Comparison 77

4 Zero-knowledge Arguments for Paillier 83
4.1 Our Contribution 85
4.2 Technical Overview of Our Results 87
4.3 Our Main Protocol ZKAoK* 97
4.3.1 Constraints for A Valid Paillier Message Ciphertext Pair 98
4.3.2 Our Main Protocol ZKAoK* 99

4.3.3 An Extended Protocol ZKAoK$^+$ 100
4.4 A Range Proof Protocol 104
4.5 Security Proof 105
4.6 Performance Evaluation 114
4.6.1 Communication Efficiency 115
Micro-benchmarks 115
Total and Average Proof Size 116
Estimate Communication Cost 117
Compare with A Baseline: Honest-but-curious Model 118
Compare with the Second Baseline: Exponential ElGamal Encryption . 119
4.6.2 Computation Efficiency 120
Time Consumed by Each Side 120
Estimate Poof and Verification Time . . . 122
4.6.3 End-to-end Performance 123
4.6.4 Comparison 124
Compare ZKAoK* with An OR-Proof . . . 124
Compare ZKAoK$'$ with Paillier Range Proofs [Lin17, Bou00] 126
Compare ZKAoK$^+$ with Existing Work . . 127
4.7 Potential Optimizations and Discussions 128
4.7.1 Potential Optimizations. 128
4.7.2 Discussions. 129

5 Blockchain-based Threshold Electronic Voting Systems133

5.1 Our Contribution 134
5.2 Overview of Our Solution 135
5.2.1 Architecture of Our System 137
5.2.2 Work Process of Our System 138
5.3 Syntax and Security Requirements 139
Syntax 140
Security Requirements 140
A Building Block Π^{sig} 141
5.4 Our Construction and Security Analysis 141
Entities and Their Notations 142
Detailed Construction 142
5.4.1 Security Analysis 144
5.5 Implementation and Performance 146
5.5.1 Implementation Setup 146
5.5.2 Performance Evaluation 146

6 Conclusions and Future Research Directions 149

References 153

List of Figures

4.1 An Arithmetic Circuit 88
4.2 Our Main Protocol ZKAoK* for Relation $\mathcal{R}^*$ . . . 101
4.3 Total and Average Proof Size. 116
4.4 Total Time Consumed by the Prover and Verifier. 121

5.1 Architecture of our blockchain-based voting system 138
5.2 Voting Process Diagram of our blockchain-based voting system 139

List of Tables

1.1 Existing Zero-knowledge Proofs and Our Main Protocol Customized for Paillier Cryptosystem. . 33

3.1 Differences Between Existing Models and Our Models in SDVS. 62

3.2 Comparison Between Our Instantiations and Existing SDVS schemes 81

4.1 Summarization and Comparison of Our Protocol with the State-of-the-art Approaches in Different Scenarios with $|N| = 2048$ bits. 87

4.2 Possible values of b_i 106

4.3 Notations and Definitions. 115

4.4 Parameter Sets Used in Our Experiment. 115

4.5 Micro-benchmarks. 116

4.6 Comparison of Real and Estimated Proof Cost. . 118

4.7 Comparison Between the Cost for Encrypting and Proving One Bit Using Our Protocol in Large Scenarios. 119

4.8 Comparison between Real and Estimated Time Cost for One Bit in Both Prover and Verifier Sides. 123

4.9 End-to-end Performance Evaluation with $\mathcal{N}_p = 800$. 123
4.10 Compare Ours with A Standard Or-proof. 124
4.11 Compare Ours with [Lin17, Bou00] for Range Proving 256-*bit* Paillier Plaintexts in $|N| = 2048$ Setting. 126

5.1 Total Time Consumed in Each Phase with $(7, 10)$ Threshold 146
5.2 Average Time Consumed in Each Phase with $(7, 10)$ Threshold 146

List of Abbreviations

PKC	Public-Key Cryptography
DVS	Designated Verifier Signature
SDVS	Strong Designated Verifier Signature
ZKP	Zero-Knowledge Proof Systems
HE	Homomorphic Encryption
AHE	Additive Homomorphic Encryption
E-Voting	Electronic Voting
TTP	Trusted Third Party
KEM	Key Encapsulation Mechanism
PRF	Pseudorandom Function
ZKAoK	Zero-knowledge Argument of Knowledge
DCR	Decisional Composite Residuosity
PPT	Probabilistic Polynomial Time
IND-CPA	Chosen Plaintext Indistinguishability
IND-CCA	Chosen Ciphertext Indistinguishability
zk-SNARK	Zero-knowledge Succinct Non-interactive Argument of Knowledge
UDVS	Universal Designated Verifier Signature
ID-based	Identity Based
RPC	Randomized Partial Checking
LRS	Linkable Ring Signature

Chapter 1

Introduction

Cryptography has been utilized for thousands of years. Initially, it relied predominantly on creativity rather than rigorous theory during the period known as "classical cryptography". However, starting in the 1970s, a rich body of theory emerged, enabling cryptography to evolve into a science, which we now refer to as "modern cryptography".

Public-key cryptography (PKC), introduced by Diffie and Hellman [Dif76] in 1976, is a prominent branch of modern cryptography. It is extensively used in our daily lives. Generally, it has a public and secret key pair, where the public key is disseminated publicly while the secret key is only known to the owner. PKC requires different keys in distinct phases of a public-key scheme, and popular PKC schemes include digital signatures, zero-knowledge proof (ZKP) systems, and encryptions. Additionally, blockchain, one of the most prevalent contemporary systems, extensively incorporates these PKC techniques in its architecture to attain various security assurances.

PKC has been extensively applied and has brought significant benefits to our everyday lives, particularly in this era dominated by digital computation. For instance, in blockchain systems, an individual's address is replaced by its public key, and a transaction links public keys of the payers and payees. Authorizing a transaction requires the owner to attach a corresponding signature using its secret key. However, as the number of transactions on the blockchain grows, it becomes infeasible to place all transactions on-chain. To address this scalability problem, zero-knowledge proofs can be utilized by placing the proof, instead of the whole transaction, on-chain, as exemplified in Zcash [Zca]. Furthermore, another recent application is to use zero-knowledge proof systems to prove the integrity of machine learning predictions. That is, the owner of a convolutional neural network model can prove that the prediction of a data sample is computed from the model without leaking additional information [LXZ21].

Modern cryptography necessitates that PKC schemes should achieve provable security. To ensure this, generally, we begin by identifying the security requirements that a desired scheme should satisfy. Subsequently, we propose a cryptographic primitive or protocol. Finally, we prove that the newly proposed construction indeed fulfills our predefined security criteria.

To be more specific, to define security in the first step, we usually define a game between an adversary and the challenger by abstracting potential attacks happen in the real life, together

with defining what kind of information that the attacker could obtain through their interaction. To prove the security, we usually do a security reduction by utilizing some mathematical problems that are widely believed to be hard. After proving its security correctly, the proposed scheme is provably secure under some computation assumptions as the underlying problem (e.g., the problem of prime factorization of large numbers) is believed to be infeasible to be solved in polynomial time.

However, as technology continues to advance and become more integrated into daily life, the deployment environments for PKC schemes are also undergoing significant changes. Consequently, new security and functionality demands are raised. For example, the increasing complexity of real-world deployment environments necessitates the development of new and enhanced security models to accurately capture these intricate interactions. Moreover, in an era where data is extensively collected by a variety of distributed sources, including companies, devices, and organizations, there emerges a critical demand to securely utilize this distributed data without compromising privacy. Consequently, the development of novel PKC schemes that cater to these evolving security models and functionality demands becomes imperative. Such advancements are crucial for enhancing the practicality and effectiveness of PKC schemes in our rapidly evolving digital landscape.

This thesis explores the evolving security and functionality

requirements in public-key cryptography, specifically concentrating on digital signatures, ZKPs, and blockchain systems. We introduce innovative constructions that are provably secure, tailored to meet the contemporary challenges posed by recent practical applications, to make PKC schemes more practical.

Digital Signatures. Digital signature schemes are one of the most widely adopted public-key cryptographic infrastructures, designed to guarantee the authenticity of a message. To sign a message, a secret key is required, while verifying the signature requires the associated public key. Since its introduction, digital signature schemes have found numerous applications in real-life scenarios, and a rich body of research has explored their different variants and security models.

Traditionally, a digital signature scheme allows anyone holding the public key to verify the validity of a signature signed by the corresponding private key. This property is advantageous in scenarios such as the dissemination of announcements where the more distributed the better. However, it may not be suitable in cases where the signature relates to commercially sensitive information. Direct transfer of such signatures may lead to industrial espionage.

Designated verifier signature (DVS) schemes [JSI96a] offer a solution to the limitation of traditional digital signature schemes where a signature can be easily transferred to any third party. A

DVS scheme allows the signer to convince a designated verifier that a message has been endorsed without the ability to transfer that conviction to any other party. This is achieved through the property that the signature can be generated by either the signer or the verifier, making it publicly verifiable. However, anyone can tell that a signature is generated from two potential signers (i.e., the signer and the designated verifier). To enhance signer's privacy, a variant of the DVS scheme called strong designated verifier signature (SDVS) scheme [JSI96a, SKM03a] is proposed, by disallowing public verification. This is accomplished by requiring the designated verifier's secret key to verify the signature's validity, ensuring that only the verifier can determine the real signer's identity.

The standard abstract model for a SDVS scheme assumes only one signer and one verifier. However, real-life applications may involve multiple signers and verifiers, which requires rethinking the security model and potential threats posed by multiple dishonest participants. Malicious actors may forge a dishonest signer or verifier, which can compromise the security of SDVS schemes designed for the previous model. Therefore, it is crucial to develop new SDVS schemes that are secure in the presence of multiple cheating participants.

Consider the following SDVS scheme. Let G, G_T be cyclic groups of the same order and $\hat{e} : G \times G \rightarrow G_T$ be a bilinear pairing between these groups. Let g be a generator of G. Further assume $H : \{0,1\}^* \rightarrow G$, be a hash function from $\{0,1\}^*$ to

G. The public-secret key pair of our example scheme is set as $(pk, sk) = (g^x, x)$.

When the signer (with public/secret key (pk_s, sk_s)) generates a signature σ for the designated verifier (with public/secret key (pk_v, sk_v)) on message m, it computes signature $\sigma = \hat{e}(pk_v^{sk_s}, H(m))$. This signature's validity can be verified by the verifier through checking if $\sigma \stackrel{?}{=} \hat{e}(pk_s^{sk_v}, H(m))$. It can be proven easily that the scheme is unforgeable in the single-user setting under the BDH hardness assumption, where H is a random oracle. In addition, it enjoys signer's privacy since identifying the actual signer implies solving the DBDH problem.

In practice, the signer may generate signatures for different designated verifiers. However, this introduces new security challenges. If an attacker gains control of the keys of the verifiers for which the signer has generated signatures, the security of the above example scheme may be compromised. To further illustrate, suppose the attacker aims to forge a signature on the message m intended for the verifier pk_v. The attacker may first creates a "rouge key" of the form $pk'_v = (pk_v)^k$ for some randomly chosen k. He may request a signature from the signer on message m with respect to pk'_v. The signature is of the form $\sigma = \hat{e}(pk_v'^{sk_s}, H(m))$. The attacker can compute $\sigma' = \sigma^{\frac{1}{k}}$ as a valid forgery on m under pk_v. Therefore, it is necessary to re-evaluate the security model of SDVS schemes in situations where multiple users are involved, as a scheme that is secure in the existing model may not be sufficient considering practical

applications.

To address the security challenges with having multiple potentially malicious signers and verifiers, raised in a more complex deployment environment in practice, it is imperative to enhance the security guarantees for SDVS schemes. Moreover, exploring novel constructions to satisfy these new security models and rigorously proving their security is essential. This advancement is crucial for ensuring the efficacy of SDVS schemes in real-world applications, where numerous users are involved, and the likelihood of malicious attacks is elevated.

Zero-knowledge Argument for Paillier Systems. In today's interconnected world, data is collected and distributed across different individuals, companies and organizations. On one hand, making use of all these distributed data will improve analysis quality and help to provide personalized service. On the other hand, since these data may include private information, disclosing them to other collaborators may break their privacy guarantees. The new demand for functionality requires collaborative use of all these distributed data to improve service quality, while maintaining privacy guarantees.

To address these concerns in cryptography, we can leverage homomorphic encryption (HE). Generally, it allows operations over encrypted messages without decryption or the knowledge

of secret keys. Additive homomorphic encryption (AHE) is often sufficient for practical deployment due to its efficiency, allowing for the addition operations. It is widely used in federated learning [ZLX+20, NWI+13], private information retrieval [GH19, FIPR05], oblivious transfer [Lin08], and electronic voting (e-voting) [KY04, DJ01a] for privacy guarantee. Its characteristic also facilitates data flow across mutually distrusted organizations. Based on it, many privacy-preserving data aggregation schemes[1] [JK12, RN10, SCR+11, PBBL11, JL13, SCR+11, BIK+17, CGB17, EDG14, MDDC15, ET12] have been proposed recently.

However, a malicious party may behave dishonestly, attempting to obtain private information by deviating from the prescribed protocol through interaction in real-world applications. To counteract such malicious behavior, one possible approach requires participants to prove that they have correctly completed each step. To achieve this objective without compromising privacy, a zero-knowledge proof (ZKP) system [MR+89] is typically employed to provide active security for the existing schemes.

ZKP systems serve as fundamental building blocks in numerous cryptographic protocols. They enable a prover to persuade a verifier of a statement's validity without revealing any

[1] We refer to a large number of aggregation scenarios here, including but not limited to computing sum, mean, minimum or maximum value, and counting frequency among other advanced statistics in machine learning.

additional information. A ZKP system must satisfy the following properties: 1) Completeness - the prover can convince a verifier of a true statement with high probability; 2) Soundness - a malicious prover cannot persuade a verifier of a false statement, except with negligible probability; 3) Zero-knowledge - a malicious verifier cannot glean any extra information through the proof. Additionally, if the ZKP system achieves soundness regarding computationally bounded verifiers, it is called an *argument*; while a *proof* can achieve soundness without computational boundaries[2].

In this dissertation, we focus on a 2-party aggregation scenario involving two data providers, denoted as $\mathcal{A}$ and $\mathcal{B}$, where $\mathcal{A}$ and $\mathcal{B}$ hold a binary vector and weight for each entity, respectively. Specifically, data provider $\mathcal{A}$ holds a set of binary vectors, $\{\mathbf{b}_i\} = \{(b_{i,1}, b_{i,2}, \ldots, b_{i,F})\}$ while $\mathcal{B}$ holds $\{W_i\}$ for each entity i. The objective is for $\mathcal{A}$ to analyze its data with the assistance of $\mathcal{B}$. Such aggregation is fundamental in constructing secure machine learning algorithms [CGB17], network traffic statistics [EDG14], recommendation systems [MDDC15], and other applications.

We can consider $\mathcal{A}$ as a proxy possessing data collected from multiple users, organizations, or devices while treating $\mathcal{B}$ as a weight provider. We use $b_{i,j}$ to indicate a particular record of entity i towards unit j and W_i to denote 'weight' for each entity,

[2]In the following, we interchangeably use proof and argument without further explanation. Our construction is a zero-knowledge argument of knowledge protocol.

where $i \in [1, M]$ and $j \in [1, F]$. They aim to jointly and securely compute the weighted sum for each unit j as,

$$S_j = \sum_i b_{i,j} * W_i = b_{1,j} * W_1 + b_{2,j} * W_2 + \cdots b_{M,j} * W_M.$$

Here, we restrict $b_{i,j}$ to be Boolean and W_i to be an integer. If the i-th entity supports/owns the j-th unit, then $b_{i,j} = 1$; otherwise $b_{i,j} = 0$. Using a binary vector to represent the possession of attributes is quite common. It can be used, for example, to indicate whether a user has a certain disease [CGB17], whether a phone has installed an app [CGB17], whether one is in some country [EDG14], presence in a certain restaurant [PBBL11], browsing history and so on.

AHE for Secure Data Analytics - An Example. We take voter analysis (or an election exit/entrance poll analysis) as an example. Pollster company $\mathcal{A}$ (e.g., CNN and Fox News) conducts interviews with voters on their voting wills in different polling stations or through telephone interviews. The other data provider, $\mathcal{B}$, holds voters' personal information such as age and salary. $\mathcal{A}$ aims to gain an indication of the average age or the income group that opts for each candidate with the help of $\mathcal{B}$, without revealing data in-the-clear to each other. In this example, each entity indicates a voter while each unit indicates a candidate. $b_{i,j} = 1$ means that the i-th voter has the will to vote for the j-th candidate. W_i is used to denote age or salary of the i-th voter. $\mathcal{A}$ intends to compute S_j. This can be

viewed as frequency estimation [ZWC+22, BS15], which generalizes boolean predicate of attribute-weighted sum [AGW20] in which W_i is Boolean.

To preserve data privacy, the aforementioned voter analysis can be conducted securely by utilizing AHE. $\mathcal{A}$ generates the key of the AHE, encrypts $\{\boldsymbol{b_i}\}$ and sends the ciphertext to $\mathcal{B}$. $\mathcal{B}$ computes the weighted sum (in the ciphertext domain) and returns the result to $\mathcal{A}$. $\mathcal{A}$ decrypts the result to get the sum of the age of the voter for each candidate and then divides the sum by the number of voters who aims to support that candidate to get the average.

Paillier cryptosystem [Pai99] is a prominent example of AHE and has been standardized by ISO [Hom19]. Since Paillier supports a very large message space, typically 2048-bit, packing is often used [ABMR20, GZ07] to reduce ciphertext expansion. In more detail, assume that $M \cdot max\{W_i\} \leq U$. $\mathcal{A}$ packs all records associated with the same entity into one Paillier plaintext m_i such that $m_i := \sum_j 2^{U*(j-1)} b_{i,j}$.$\mathcal{A}$ encrypts each m_i into c_i and sends them to $\mathcal{B}$. $\mathcal{B}$ then computes $\bar{C} = \prod_i c_i^{W_i}$ and sends the result back. $\mathcal{A}$ decrypts $\bar{C}$ and obtains $\bar{m}$. It can parse $\bar{m}$ to obtain $S_j = \sum_i b_{i,j} \cdot W_i$.

The Need for Proofs of Well-formedness. While the above approach protects the privacy of $\mathcal{A}$'s data, it does not guarantee the privacy of $\mathcal{B}$'s data. Specifically, a malicious $\mathcal{A}^*$ can obtain the weight of a specific entity, say, the i^*-th entity, by biasing the message structure. $\mathcal{A}^*$ computes m_{i^*} as, $m_{i^*} = \sum_j 2^{U*(j-1)+U/2} b_{i^*,j}$.

Consequently, the value of W_{i^*} appears in the higher part of S_j (provided that $U >> S_j$). In other words, the above approach only provides honest-but-curious security. To guard against this kind of attack, $\mathcal{B}$ should require $\mathcal{A}^*$ to prove that all ciphertexts are well-formed. We listed three requirements regarded the well-formedness in our running example of voter analysis.

- *Packing with Binary Messages.* To support Paillier with packing, $\mathcal{A}$ needs to prove that each $b_{i,j}$ is binary and it should be located in the correct position.

- *Equality Proof for Sum of Records.* In a plurality-at-large election, there are multiple, say, t, seats to be elected. Each voter can vote for at most t candidates. This kind of electoral system is utilized for electing Senate nominees in Canada (Alberta) [nom], Federal Senate in Brazil [elec], Council of States in Switzerland (2019) [eleb], and the election committee in Hong Kong [elea]. To prove that c_i is the correct encryption of entity i's preference, $\mathcal{A}$ should prove that there are at most t 1's in each voter's vote m_i.

- *Range Proof for Sum of Units.* Even if the protocol is secure, weight may be leaked from the output of the analysis. For example, if there is only one voter who votes for candidate j, S_j reveals the weight of that voter. $\mathcal{B}$ will additionally requires a proof that there are more than T 1's in the records $\{b_{i,j}\}$ towards each candidate (i.e., unit) j.

The challenge here is that the proof should leak no information about $\mathcal{A}$'s data. Otherwise, it will compromise $\mathcal{A}$'s data privacy. Zero-knowledge proof/argument (ZKP) systems [GMR85] allow a prover to convince a verifier of the truth of a statement, without revealing additional information, making it an ideal tool to mitigate the above tension.

Limitations of Existing ZKPs for Paillier. Developing ZKPs for different relations among Paillier plaintexts is valuable. There has been a rich body of work focusing on range proofs [Lin17, LN18], proving the plaintext is 0 [CDN01], multiplication [DJ01a] and a sequence of power relations [HL09], and so on. However, none of the existing works focus on Paillier with packing. Furthermore, even without packing, proof size and verification time are linear in the number of entities, making them unfit for data analytics involving data from hundreds or thousands of entities. We note that in many cases, $\mathcal{A}$ may collaborate with multiple weight providers. Thus, it is desirable to have proof that is non-interactive, small, and efficient in verification.

One may consider utilizing existing zk-SNARKs. Significant progress has been made recently on constructing efficient ZKPs supporting statements expressed in arithmetic (over a prime field) or Boolean circuits [Gro09, PHGR13, BSCG+13, JKO13, Gro16, BCC+16, GMO16, BCG+17, BBB+18, MBKM19, HKR19, ZXZS20]. However, directly applying these ZKPs to our problem results in proof of unacceptable efficiency. The main problem is that an arithmetic or Boolean circuit representing Paillier

encryption is huge. Specifically, Paillier Encryption involves modular exponentiations over N^2, and it is unclear how to represent this operation efficiently using Boolean gates or addition/multiplication gates over a prime field. Thus, representing the well-formedness of Paillier ciphertext (with packing) will lead to an impractical circuit size. For example, proving one single plaintext-ciphertext pair is valid involves a circuit with 13335083 gates, even in the modest setting of $|N| = 1024$[3].

Therefore, there is a need to construct efficient zero-knowledge proof systems suitable for Paillier, providing active security towards the above (or other related) scenarios. Considering the previous scenario, as the provider $\mathcal{A}$ may re-analyze its data with the help of different $\mathcal{B}$'s that may empower distinct levels of computation resources, the proof size should be small and the verification cost should be low enough. In summary, the objectives of this proof system are 1). small proof size 2). low verification cost 3). affordable proof time.

Blockchain. Since Satoshi Nakamoto's seminal introduction of blockchain technology [Nak08] in 2008, it has emerged as an indispensable platform for a multitude of privacy-centric applications, owing to its decentralized and anonymous attributes. The rise in security requirements is evident, as numerous real-world applications are transitioning to blockchain-enabled systems. This trend is particularly notable in sectors where privacy

[3]We are unable to generate the circuit for $|N| = 2048$ on our PC with Intel Core i9-12900K CPU and 160 GB of memory (32 GB RAM and 128 GB swap).

and trust are paramount, such as healthcare systems [SJZ+19, GSM+20], finance [SLL+22, Tak20], IoT [MO18], transportation [LTK+22, WZZ+22], and many others. Blockchain provides an abstraction of a public append-only ledger that achieves full decentralization without requiring a single authority, as every on-chain transaction can be verified by its robust inherent consensus mechanism. By transitioning into blockchain systems, many applications can avoid traditional trusted intermediaries, thereby reducing security dependencies. Additionally, the decentralized framework of blockchain significantly mitigates the risks associated with single-node failures.

Voting, as a crucial social function, has also undergone a significant transformation in the digital era. Traditional voting approaches, including physically voting and mail-in ballots, suffer from various limitations. For example, in-person voting is not friendly for disabled individuals, while the mail-in ballots are susceptible to tampering during transit. Consequently, there has been a growing interest in devising an electronic voting (e-voting) system that preserves the security integrity of traditional voting methods. First proposed by David Chaum in 1981 [Cha81], the development of e-voting systems has rapidly evolved [Adi08, CGGI13, HKLD17]. The transition of e-voting to blockchain platforms is an active area of exploration, particularly for its potential to enable decentralized voting processes. This approach has garnered increased attention in recent years as a means to prevent COVID-19 transmission during elections.

In general, the fundamental security and functionality demands for an e-voting scheme involve privacy and accuracy [FOO93]. On one hand, it is not desirable to reveal the vote will of a voter before the tallying phase; On the other hand, the final tallying result should be accurate. To achieve the above requirements, e-voting schemes usually utilize distinct cryptographic techniques, for example, mix-net [Cha81], homomorphic encryption [RAD78], linkable ring signature [LWW04a] and blind signature [Cha83] schemes to provide various features.

A mix-net protocol[Cha81] is usually deployed by a series of mixers where its output is secretly permuted (i.e., shuffled) among them. It provides anonymity for e-voting schemes by removing the link between ballots and voters. Another approach, homomorphic encryption, provides privacy as it allows one to operate the ballots in ciphertext domain without decryption. Thus the single real voting will cannot be revealed to the taller. Besides, it enables fast tallying by adding all votes in encrypted forms together, followed by a single decryption to obtain the final result. However, a subtle issue involved in the use of above approaches is the need for a trusted third party (TTP) to produce proof of correctness for those hidden operations. Therefore their computation cost is usually very high.

A linkable ring signature [LWW04b] is also usually deployed to construct an e-voting system. As a ring signature, it allows a member to sign messages anonymously on behalf of a group

of signers, while hiding its actual identity. The linkability property guarantees that any two signatures generated by the same entity can be determined by anyone. Thus it can prevent unauthorized voters from casting ballots and malicious voters from casting multiple votes, while maintaining privacy of legitimate voters. However, it always requires at least one operation that has computation complexity linear to the size of anonymity set (i.e., the number of legitimate voters in the case of e-voting) in all existing schemes. Therefore when the number of legitimate voters is large, the resulting scheme will be inefficient in practice.

Blind signature [Cha83] is another typical approach to provide anonymity. It usually involves two parties in the protocol, allowing one party say, $\mathcal{A}$, to get a valid signature of his message from another party, $\mathcal{B}$, without revealing the actual content in the message to $\mathcal{B}$. When applying in the e-voting scenario, each voter (i.e., $\mathcal{A}$) can obtain its certified ballots from an authority (i.e., $\mathcal{B}$) in a privacy-preserving manner without disclosing the contents (its votes) to the authority. However, once the single authority becomes malicious, he can create as many ballots as he wish without being detected. That is, the successful run of an e-voting campaign also relies on a TTP (authority).

Besides, a public bulletin board is usually needed to publish the final results in an e-voting system and it has to be trusted by all participants. Blockchain, due to its decentralization nature, is usually deployed to instantiate this role. McCorry et al.

[MSH17] presented the first implementation of a decentralized and self-tallying e-voting protocol. Some companies like The Blockchain Voting Machine [Her], FollowMyVote [Ara] also proposed solutions of adopting blockchain as a ballot box. However, all these solutions are platform dependent. More recently, Yu et al. [YLS+18] proposed a new approach to construct platform-independent secure voting system based on blockchain. However, the need for a trusted third party remains.

Therefore, a truly decentralized e-voting system is required in order to eliminate the need of a trusted third party. Besides, the system should be platform-independent and efficient enough so that it can be compatible with real-life applications. The intended system also should utilize blockchain with various cryptographic schemes to offer different properties like verifiability, usability and so on.

In this dissertation, we aim to provide solutions to address the security and functionality demands that arise in real-world applications of strong designated verifier signatures, zero-knowledge proof systems for Paillier, and electronic voting schemes, to make public-key cryptographic schemes more practical. Our work is focused on the following aspects:

- Addressing the security demands of SDVS considering more complex real-world scenarios involving multiple signers and verifiers, and providing a generic and provably secure construction of SDVS that meets these enhanced security

guarantees.

- Addressing the functionality demand in two-party data aggregation scenarios and constructing an efficient ZKP system for Paillier system with low communication and verification costs, as well as manageable proof generation effort. We focus on scenarios involving hundreds of records, which are common in real-life applications.

- Addressing the security and functionality demands in e-voting and designing a fully decentralized e-voting system without reliance on a single trusted third party, while simultaneously providing privacy, accuracy, and efficiency guarantees for its deployment in reality.

We provide solutions to the aforementioned problems in this thesis, specifically:

- To address the security challenge of SDVS with multiple signers and verifiers in real-life scenarios, we first formalize two strengthened models: multi-user and multi-user$^+$. Then, we propose a generic construction of SDVS from Key Encapsulation Mechanism (KEM) and Pseudorandom Function (PRF) in the standard model. Our generic construction is secure in the multi-user setting if the underlying KEM and PRF are secure. We also provide instantiations based on the DDH and LWE assumptions, respectively

- To address the functionality demand of two-party aggregation, we introduce an efficient zero-knowledge argument of knowledge system customized for Paillier cryptosystem. Our system is based on a constraint system defined over the ring of residue classes modulo a composite number, and incorporates novel techniques designed for arguing binary values in this setting. With sub-linear proof size, low verification cost, and acceptable proof generation effort, our system supports batch proof generation/verification and is instantiated for various scenarios. Specifically, we consider Paillier with packing in Scenario 1, where we pack 25600 bits into 400 ciphertexts, resulting in a proof that all ciphertexts are correctly computed that is 17 times smaller and 3 times faster to verify than the naive implementation of using 25600 OR-proofs without packing. We conduct extensive experiments to demonstrate the practicability of our system.

- To address the security and functionality demand of an e-voting system, we propose a novel blockchain-based e-voting scheme featuring distributed authorities. We leverage threshold blind signature to distribute trust for registration and threshold ElGamal decryption to distribute trust in ballot tallying. By combining these techniques with decentralized blockchain technology, our system achieves

verifiability, eligibility, fairness, and anonymity with reduced trust. We conduct experiments to examine its efficiency and demonstrate the applicability of our approach. The results validate the effectiveness and practicality of our proposed e-voting scheme.

Next, we further illustrate the background and contribution of each result from Section 1.1 to Section 1.3. We give their related works in Section 1.4.

1.1 Generic Constructions for Strong Designated Verifier Signature Schemes

The concept of undeniable signature was first proposed by Chaum et al. [CVA90]. It consists of a signer named Alice and a verifier named Bob. When Bob wants to verify the signature created by Alice, he must interact with Alice through an interactive verification protocol. This means that the verifier cannot check the validity of signature by himself. In other words, the signer has complete control of the signature in order to avoid other undesirable verifiers from getting convinced of its validity. However, blackmailing [DY91] and mafia [DGB87] attacks have raised concerns about the security of undeniable signatures.

To address these issues, Jakobsson et al. [JSI96b] proposed a designated verifier signature (DVS) scheme with briefly discussing the concept of strong designated verifier signature (SDVS).

Their DVS scheme is the first non-interactive undeniable signature scheme by using designated verifier proof. In their scheme, only designated verifier can be convinced by the signature's validity or invalidity without requiring any interaction with the presumed signer. This scheme follows a very simple approach: each user holds two key pairs, one for generating signatures while the other for encrypting signatures. When Alice (signer) wants to generate a signature to Bob (verifier), she first uses her signing key to generate a signature, followed by encrypting it under Bob's encryption key. Once Bob receives the signature, he decrypts it first and verifies its validity. This simple approach requires an encryption followed by a verification, which is therefore less efficient than desired.

The concept of SDVS was first formalized by Saeednia et al. [SKM03b]. The idea of privacy of signer's identity was then formalized by Laguilaumie et at. [LV04], capturing the property of a strong designated verifier signature where no third party can distinguish which signer generates the signature without the verifier's secret key.

Since the introduction of SDVS, many schemes have been proposed. Huang et al. [HSMZ08] proposed the first short designated verifier signature scheme and its identity-based variant. Huang et al. [HYWS11] proposed the first SDVS scheme in the standard model, based on DDH problem. Subsequently, new schemes under various assumptions (e.g. DBDH, CDH, GDH, $\mathcal{R}$-SIS) have been proposed [TCZ+12, AVS13, CJZ+19].

However, the security of the existing SDVS schemes heavily relies on specific hardness assumptions and their security is only guaranteed in the single-user setting. In this setting, the attacker is given the public keys of the target signer and verifier and can issue queries with respect to these two entities. The existing models may not capture attacks in practical scenarios, as we have described in the previous section. Therefore, in this thesis, we aim to fill this gap by initiating the study of SDVS in the multi-user setting, which better captures the real-world scenarios and potential attacks

Specifically, we introduce two strengthened models. In the first model, the adversary can issue signature queries from a list of signers and verifiers. In the second model, the adversary can issue queries to the verifier as of its own choice. We also give a generic construction, based on *KEM* and *PRF* schemes. The security of our construction relies on the underlying *KEM* and *PRF* being secure. We provide four instantiations that are secure in both standard and quantum models and compare them with existing SDVS schemes. Our generic construction allows for the construction of diverse SDVS schemes that satisfy different security requirements based on distinct KEM schemes. Additionally, any progress made in KEM can be directly applied to improve SDVS schemes.

1.2 Efficient Zero-knowledge Argument of Knowledge for Paillier

We introduce an efficient zero-knowledge argument of knowledge system customized for Paillier cryptosystem. Our proposed system exhibits sub-linear proof size, low verification cost, and acceptable proof generation effort. Additionally, it facilitates batch proof generation and verification. In contrast, existing works specialized for Paillier cryptosystem are characterized by linear proof size and verification time. Employing existing sub-linear argument systems for generic statements (e.g., zk-SNARK) results in unaffordable proof generation cost since it involves translating the relations to be proven into an inhibitive large Boolean or arithmetic circuit over a prime order field. Our system does not suffer from these limitations.

At the heart of our argument systems lies a constraint system defined over the ring of residue classes modulo a composite number, augmented with innovative techniques specifically devised for arguing binary values in this context. We then build upon the approach presented by Jonathan et al. [BCC$^+$16] to transform the constraint system into a sub-linear argument system. Our constraint system is versatile and can be employed to express a variety of relations commonly associated with the Paillier cryptosystem, including range proof, correctness proof,

relationships between bits of plaintext, relationships of plaintexts among multiple ciphertexts, and more. Moreover, our argument enables batch proof generation and verification, with the amortized cost surpassing the performance of state-of-the-art protocols specialized for Paillier when the number of Paillier ciphertexts is on the order of hundreds.

In our study, we implement our system across several scenarios and perform comprehensive experiments. In Scenario 1, we focus on Paillier with packing. By packing 25,600 bits into 400 ciphertexts, we demonstrate that a proof confirming the correct computation of all these ciphertexts is 17 times smaller and 3 times faster to verify compared to a naive implementation that employs 25,600 OR-proofs without packing. Moreover, our system allows for the proof of additional statements with minimal extra effort. For instance, it is possible to prove that the sum of a subset of witness bits is less than a specified threshold t.

Another scenario we investigate is range proof. Our system proves that each plaintext in 200 Paillier ciphertexts has a size of 256 bits, with a proof size that is 10 times smaller than the state-of-the-art. Our analysis indicates that our system is asymptotically more efficient than existing protocols and is particularly well-suited for situations involving a large number of Paillier ciphertexts (more than 100), a common occurrence in data analytics applications.

1.3 Distributed Electronic Voting in Blockchain

In developing distributed blockchain-based e-voting schemes, we prioritize computation cost and efficiency to ensure practical applicability. Our e-voting system leverages blind signature, encryption, and blockchain with threshold techniques. In this construction, we carefully balance efficiency, anonymity, and the necessity to eliminate trusted parties entirely.

We employ threshold blind signature for voter registration, which proves more efficient than linkable ring signature for large group sizes. The trade-off lies in trusting a threshold number of registration authorities not to misuse their power or enable the registration of unauthorized voters. In our system, ballots are encrypted using ElGamal encryption, while the corresponding decryption key is distributed among a set of authorities using threshold techniques. Our approach achieves rapid tallying typically associated with systems based on homomorphic encryption.

We emphasize that our system's efficiency is practical enough for real-world deployment, as demonstrated by our experimental results.

In summary, this thesis presents an e-voting system that utilizes distributed blind signature, encryption, and blockchain technology. Our proposed system offers the following notable features.

- *Without a single trusted third party.* Our system employs a

threshold blind signature scheme, wherein the role of the registration authority is distributed among n organizers. Similarly, n^* tellers assume the role of the tallying authority. By integrating these two techniques, our system effectively eliminates the reliance on a single trusted third party.

- *Distributed.* The registration and tallying capabilities are distributed in a round-efficient manner, seamlessly aligning with the intrinsic decentralized nature of blockchain technology. Consequently, our system attains a truly distributed framework.

- *Anonymous.* We utilize blind signature to safeguard voters' identities. This ensures that even in the event of collusion among the set of registration authorities, they would still be unable to link a ballot to a registered voter.

- *Efficient.* We conduct a practical implementation of our system to assess its efficiency. The experimental results demonstrate that our system's performance is sufficiently efficient for real-world adoption. Notably, the time and complexity on the user side remain constant.

1.4 Related Works

In this section, we provide an overview of related works covering various research aspects addressed in this thesis.

Strong Designated Verifier Signature. In 1989, Chaum and Van introduced the concept of undeniable signatures to prevent unauthorized verifiers from determining the validity of signatures [CVA90]. This is achieved by requiring the signer's participation in the verification process, thus allowing the signer to maintain complete control over the signature. However, a drawback is that the signer may not always know the identity of the person he/she is interacting with, which led to the introduction of designated verifier signature (DVS) in [JSI96a].

DVS schemes provide message authentication without the non-repudiation property characteristic of traditional signature schemes. They are designed to convince only a single verifier, with no one else able to confirm the signature's validity. This is ensured by allowing the verifier to produce a signature that is indistinguishable from the one generated by the signer. The idea of strong designated verifier signature (SDVS) was also discussed in [JSI96a] and its definition was initially proposed by Shahrokh et al. [SKM03b] in 2003. The "strongness" property requires that no third party can identify the signature's originator when only given public keys, with the designated verifier's secret key needed for verification. This property was first formalized in [LV05a], where Laguillaumie and Vergnaud defined the "privacy of signer's identity" feature to capture it.

Since its introduction, numerous follow-up works have considered various additional features in the context of SDVS. Susilo et al. [SZM04] firstly introduced a variant in the identity-based

(ID-based) setting based on pairing. It was further enhanced in [HSMZ06, HSMZ08] basing on Diffie-Hellman key-exchange, with first proposing the notion of "short SDVS" and extending it to an identity-based version. Considering the property called "non-delegatability" where it requires that anyone who can produce a valid signature on behalf of the signer or the designated verifier, he must knows the secret key of any one of them. Huang et al. proposed the first ID-based SDVS with supporting non-delegatability property [HSW09] and a SDVS with non-delegatability [HYWS11]. [AVS13] proposed a non-delegatable SDVS scheme relying on a trusted third party with removing the need of bilinear pairing. [TL14] then proposed a short non-delegatable SDVS sceme with only requiring 2 elements in the signature while the concurrent SDVS signatures contain at least 3 elements without delegatability. Geotae [NJ16] presented the first lattice-based SDVS construction in the standard model in 2016.

A parallel research area is universal designated verifier signatures (UDVS) [SBWP03], which allow the holder of a signature to designate any desired verifier. This verifier can be convinced that the holder indeed possesses a signature, while being unable to transfer this conviction to any other party. Numerous works have been conducted in this area, such as [SWP04, ZFI05, BSNS05, SSN08]. Zhang et al. [ZSMC05] proposed the first ID-based UDVS scheme. In terms of non-delegatability, [HSMW06] presented the first provably secure UDVS without

delegatability. Hou et al. [HHL+15] introduced a designated designated verifier transitive signature, with additional features like non-delegatability considered in [LWY12, AVS13, TL14].

Zero-knowledge Proof Systems for Paillier. As we provide a ZKAoK protocol for Paillier, we now review and compare existing ZKPs customized for Paillier. Proof systems tailored for Paillier can be primarily divided into two categories: one focusing on proving the validity of an RSA modulus (i.e., Paillier public key) and the other on proving plaintext relations (including range). We use $\mathsf{pk} = N$ to denote a Paillier public key and $\mathsf{PL.Enc}_{\mathsf{pk}}(m; r)$ to denote Paillier encryption of message m with randomness r. We provide a detailed comparison of distinct relations related to the Paillier cryptosystem that existing works and ours aim to prove in Table 1.1.

Proving the Validity of A Paillier Public Key. [CM99] initially proposed a zero-knowledge proof for a number that is the product of two safe prime integers. It can be directly used for proving a valid Paillier public key. [HMR+19] also outlined a folklore method of proving the validity of an RSA modulus by demonstrating that $gcd(N, \phi(N)) = 1$. Although a standard Paillier public key is generated from two prime numbers, this statement still suffices to provide all Paillier properties (e.g., additive homomorphism) through this requirement (see [Lin17] Sec. 3.1). Moreover, [HL09] suggests combining methods in [BFL91] and [GP87] for proving an RSA composite (for more

detailed discussions, see [HL09]). This category of proofs is orthogonal to ours. Indeed, in our two-party aggregation scenario, party $\mathcal{B}$ may also request that $\mathcal{A}$ provides a proof that his/her Paillier encryption key is correctly formed.

Proving Paillier Plaintexts Relation. Another category primarily focuses on proving relations among Paillier plaintexts. [CDN01] provides a construction for proving knowledge of an encrypted plaintext. For proving that the plaintext is 0, [DJ01a] offers constructions, which are actually proofs of the Nth power in Paillier. To prove multiplicative relations among Paillier plaintexts, [DJ01a] presents constructions on Π_{mul}. In other words, we can prove that a message is the product of two other messages. For proving a more advanced relation—a sequence of powers—[HL09] constructs Π_{pow} based on Π_{mul}. In [Lin17], they bridged two different worlds, Paillier encryption and elliptic curve groups, proposing a zero-knowledge proof for language $\mathcal{R}_{\mathsf{pl-ec}}$. This proof shows that the message in a given Paillier ciphertext is the discrete log of a given elliptic curve point. Later, [LN18] gives constructions on Paillier and Pedersen commitment. It proves that the same value is used in encryption and commitment scheme. Additionally, [Lin17] and [LN18] also gave range proof that is customized for Paillier, where [Lin17] was adapted from the range proof in [Bou00]. However, to achieve efficiency, both of them can only gave inexact range proof (i.e., with slack).

To devise an exact range proof in Paillier, one may employ

tight range proof techniques, such as [Bou00, BBB+18], with some adaptations. Since these range proofs are not compatible with Paillier, an integer commitment scheme [FO97] is required as a bridge. In more detail, let $\mathsf{Enc}(x)$ represent the Paillier encryption of x. Let CMT_I denote an integer commitment scheme, and let CMT_R be the commitment scheme in which an efficient range proof exists. To prove that x lies within an exact range, the prover first generates commitments $c_1 := \mathsf{CMT}_I(x)$ and $c_2 := \mathsf{CMT}_R(x)$. The prover then engages in the following protocols: i) Π_{c1}: c_1 is a commitment to x; ii) Π_{c2}: the same value, x, is committed in c_2; iii) Π_{range}: the committed value in c_2 is within a certain range. Π_{range} can be achieved by invoking the existing range proof on c_2. Although the range proof is efficient, auxiliary commitments may impose a lower bound on range proof size.

Electronic Voting in Blockchain. In an e-voting scheme, vote secrecy and verifiability are two fundamental requirements [KV16]. Mix-net [Cha81], homomorphic encryption [RAD78], linkable ring signature [LWW04a] and blind signature [Cha83] are the common cryptographic approaches adopted to achieve these two properties (i.e., anonymity and verifiability) in e-voting systems.

Mix-net. One of the most popular anonymization approaches adopted is the mix-net scheme, which was first proposed by Chaum [Cha81] in 1981. Generally, the protocol consists of a

Table 1.1: Existing Zero-knowledge Proofs and Our Main Protocol Customized for Paillier Cryptosystem.

Paper	Proved Relation	Corresponding Protocol
[CM99]	$\mathcal{R}_{\mathsf{composite}} = \{(N,(p,q)) : N = p \cdot q \wedge p, q \text{ are primes}\}$	$\Pi_{\mathsf{composite}}$
[HMR+19]	$\mathcal{R}_{\mathsf{rsa}} = \{(N,\phi(N)) : gcd(N,\phi(N)) = 1\}$	Π_{rsa}
[DJ01a]	$\mathcal{R}_{\mathsf{zero}} = \{((c,pk),r) : c = \mathsf{PL.Enc}_{\mathsf{pk}}(0;r)\}$	Π_{zero}
	$\mathcal{R}_{\mathsf{mul}} = \{(c_1,c_2,c_3,pk),(m_1,r_1,m_2,r_2,r_3) : c_1 = \mathsf{PL.Enc}_{\mathsf{pk}}(m_1;r_1) \wedge c_2 = \mathsf{PL.Enc}_{\mathsf{pk}}(m_2;r_2) \wedge c_3 = \mathsf{PL.Enc}_{\mathsf{pk}}(m_1 \cdot m_2;r_3)\}$	Π_{mul}
[CDN01]	$\mathcal{R}_{\mathsf{enc}} = \{(c,pk),(m,r) : c = \mathsf{PL.Enc}_{\mathsf{pk}}(m;r)\}$	Π_{enc}
[HL09]	$\mathcal{R}_{\mathsf{pow}} = \{(c_1,c_2,\cdots,c_d,pk),(m,r_1,\cdots,r_d) : \forall i \in [1,d], c_i = \mathsf{PL.Enc}_{\mathsf{pk}}(m^i;r_i)\}$	Π_{pow}
[Lin17]	$\mathcal{R}_{\mathsf{pl.range}} = \{(c,pk),(m,r) : c = \mathsf{PL.Enc}_{\mathsf{pk}}(m;r) \wedge m \in \mathbb{Z}_q\}$	$\Pi_{\mathsf{pl.range}}$
	$\mathcal{R}_{\mathsf{pl-ec}} = \{(c,pk,\mathbb{G}_{ec},Q,G_{\mathsf{ec}},q),(m,r) : c = \mathsf{PL.Enc}_{\mathsf{pk}}(m;r) \wedge Q = m \cdot G_{\mathsf{ec}} \wedge m \in \mathbb{Z}_q\}$	$\Pi_{\mathsf{pl-ec}}$
[LN18]	$\mathcal{R}_{\mathsf{pl-ped}} = \{(c,pk,c',g,h,N'),(m,r,\rho) : c = \mathsf{PL.Enc}_{\mathsf{pk}}(m;r) \wedge c' = g^m \cdot h^\rho \mod N'\}$	$\Pi_{\mathsf{pl-ped}}$
	$\mathcal{R}_{\mathsf{pl.range'}} = \{(c,pk),(m,r) : c = \mathsf{PL.Enc}_{\mathsf{pk}}(m;r) \wedge x \in \mathbb{Z}_q\}$	$\Pi_{\mathsf{pl.range'}}$
ours	$\mathcal{R}^* = \{(\{c_i\}_{i\in[1,\mathcal{N}_p]},pk),(m_i,r_i,b^{(i)}_{32(s-1)})_{i\in[1,\mathcal{N}_p],s\in[1,64]} : \forall\, i \in [1,\mathcal{N}_p] \text{ and } s \in [1,64], c_i = \mathsf{PL.Enc}_{\mathsf{pk}}(m_i,r_i) \wedge$ $m_i = \sum_{s\in[1,64]} 2^{32(s-1)} \cdot b^{(i)}_{32(s-1)} \mod pk \wedge\ b^{(i)}_{32(s-1)} \in \{0,1\}\}$	ZKAoK^*
	$\mathcal{R}' = \{(\{c_i\}_{i\in[1,\mathcal{N}_p]},N,\beta),(m_i,r_i,b^{(i)}_k)_{i\in[1,\mathcal{N}_p],k\in[0,\lvert\beta\rvert-1]} : \forall\, i \in [1,\mathcal{N}_p],\ c_i = (1+N)^{m_i} \cdot r_i^N \mod N^2 \wedge m_i \leq \beta\}$	ZKAoK'

series of mixes, which can shuffle and re-mask the ballots to break the link between voters and their ballots. To avoid a single mixer maliciously modifying the output or refusing to participate in the protocol, Park, Itoh, and Kurosawa [PIK93] proposed an approach named re-encryption mixes, which has become the most widely studied research line. Its particular protocol was broken in [PP90, Pfi95] and later fixed in [OKST97]. Broadly speaking, this approach consists of two distinct stages: the first stage involves shuffling and re-encrypting input ciphertexts, while the second stage decrypts the outputs of the first stage.

The primary challenge in designing re-encryption mix-net schemes lies in achieving computational efficiency when proving the correctness of servers' operations. Some works [SK95, OKST97, Abe98] employ cut-and-choose zero-knowledge proofs. Although substantial efforts [Mas99, FS01, Nef01] have been made to improve the efficiency of these proofs, their computational cost remains quite high.

In 2002, Jakobsson et al. [JJR02] introduced a new technique called randomized partial checking (RPC) mix net to enhance the robustness of mix nets without requiring complete correctness proofs. This technique trades off some privacy for increased efficiency. The core idea is to have each server prove their operations' complete correctness using pseudo-randomly

selected subsets of input/output pairs. While the RPC technique is well-suited for e-voting systems, ensuring voter privacy and correct operation, generating and verifying such proofs can be challenging due to their computational demands. The first implementation of this technique was not realized until 2002 [FMM+03]. Although the efficiency of this approach was further improved in [FMS10], the time consumed in proving remains a bottleneck.

Golle et al. [GZB+02] proposed optimistic mixing, where they aim to verify that the product of all the inputs equals to the product of all the outputs. Besides, their protocol includes redundancy checks, which, in conjunction with product checking, ensure perfect correctness. Boneh and Golle [BG02] employed similar proof techniques while achieving different properties. The scheme presented in [BG02] results in the lowest total computational cost, although it only guarantees "almost entirely correct" mixing.

Homomorphic Encryption. Homomorphic encryption, which was first proposed in [RAD78], is another commonly deployed approach to maintain anonymity and privacy in voting systems. The application of homomorphic encryption in e-voting schemes began with the work of [BY86, CF85], and numerous subsequent studies have been conducted in [BFP+01, DJ01b, FPS01]. Generally, it enables the aggregation of multiple ciphertexts without decrypting each one. For instance, each voter encrypts his/her ballot using a homomorphic scheme, with the

public key published before voting. Suppose there are two voters with ballots b_1 and b_2, and corresponding ciphertexts c_1 and c_2. One can aggregate these encrypted ballots as $c' = c_1 * c_2$. After decrypting c', the final voting result, $m_1 + m_2$, can be obtained. The homomorphism property allows fast tallying without decrypting individual ballots, preserving anonymity and privacy. Protocols utilizing homomorphic encryption can be found in [BT94, CFSY96, CGS97, HS00].

Existing e-voting systems predominantly adopt Paillier [Rya08, XSHT08] and ElGamal [KY02, LK03] encrypting schemes. Building upon the models established by Benaloh et al. [CF85, BY86, Ben87], Cramer proposed a novel multi-authority protocol [CFSY96] that utilizes distributed authorities instead of a single one. The encrypted votes are shared among multiple authorities, who use verifiable secret sharing to ensure the posted shares genuinely represent the actual vote. This scheme was further enhanced in [CGS97], which introduced the first optimal solution for large-scale systems through threshold techniques.

However, this type of approach is not a general construction, as it requires voter choices to be binary (i.e., yes or no). Consequently, it cannot be easily applied to multi-choice voting systems due to the significant computational cost. Moreover, since no single ballot is decrypted, the cast ballots must be proven correct using zero-knowledge proof, which demands substantial computational effort. Therefore, when compared to

the mix-net technique, homomorphic encryption is more suitable for voting systems with a small number of candidates (e.g., yes/no voting) [ABDV03]. It can also be combined with other primitives to provide additional properties. For instance, in [SMPP10], homomorphic encryption was integrated with mix-net to enhance the efficiency of the voting system.

Linkable Ring Signature. Linkable ring signature (LRS) was proposed in [LWW04a] as a means to ensure the authenticity of ballots in e-voting systems. It incorporates the property of linkability into ring signatures [RST01]. In a ring signature scheme, any group member can anonymously sign a message on behalf of the group without the need to reveal its identity. Furthermore, an LRS scheme enables anyone to determine whether two ring signatures were signed by the same group member. In other words, users can maintain anonymity if they only sign once, but two ring signatures can be linked if signed by the same member. This property allows for the detection of double voting. Numerous e-voting schemes have been constructed based on LRS, such as [CLW08, LWW04b].

Generally, the linkability is achieved by adding a linkability tag to a ring signature, where the tag can be uniquely defined by the event identifier and member's signing key. Consequently, for the same event and the same voter, only one ballot can be cast; otherwise, anyone can detect double voting.

Early LRS constructions [LWW04b, TWC$^+$05] typically suffered from large signature sizes, which were linear to the group

size. Subsequent research aimed to reduce the signature size, as seen in works such as [DKNS04, TW05]. Some protocols have achieved constant signature sizes in ID-based schemes [CSY06, ALSY06] and PKI-based schemes [ACST06]. Despite the numerous improvements proposed, at least one operation consistently incurs a computation cost linear to the group size. Consequently, LRS is not suitable for large-scale voting scenarios.

Blind Signature. Blind signature, introduced by [Cha83], is commonly used in e-voting systems. It enables a user to obtain a signature on their message from the signer without revealing the content. Generally, blind signature is employed in e-voting systems in two ways. Specifically, during the registration phase of the e-voting system, a legitimate voter receives a blind signature on a random value. This signature-value pair can be utilized to prove the voter's authenticity. Blind signature is considered the most promising approach for constructing large-scale elections [Oka98] and offers greater computational efficiency [MZO+99]. This technique was first applied in a secret e-voting mechanism by Fujioka et al. in 1992 [FOO93] and later improved upon in [MZO+99].

Okamoto proposed the first practical receipt-free voting schemes for large-scale elections in [Oka98]. Two subtle issues should be noted when applying blind signature. First, an anonymous channel must be implemented when voters cast their ballots. Second, compared to linkable ring signature, blind signature

requires the signer to be trusted. If the signer is compromised, an attacker would be able to cast as many ballots as desired.

To address this issue, threshold techniques [DF90] can be applied to distribute authorities. Juang et al. [JLL02] proposed a scheme to support distributed trust by applying threshold techniques. Later, Mateu et al. [MSV13] introduced a threshold voting system that achieves the property of public verifiability.

1.5 Thesis Organization and Derived Publications

The publications derived from this thesis include the following,

- **Borui Gong**, Man Ho Au, Haiyang Xue. Constructing Strong Designated Verifier Signatures from Key Encapsulation Mechanisms. In 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), Rotorua, New Zealand, 2019, pages 586-593.

- **Borui Gong**, Xingye Lu, Lau Wang Fat, Man Ho Au. Blockchain-Based Threshold Electronic Voting System. In Security and Privacy in Social Networks and Big Data (SocialSec 2019), pages 238–250.

- **Borui Gong**, Wang Fat Lau, Man Ho Au, Rupeng Yang, Haiyang Xue, Lichun Li. Efficient Zero-Knowledge Arguments For Paillier Cryptosystem. (To appear in *S&P* 2024).

The rest of this thesis is organized in 6 chapters. The preliminaries are given in Chapter 2. In Chapter 3, we formally define two strengthened SDVS models and give a generic construction. We present our efficient ZKP systems customized for Paillier in Chapter 4. In Chapter 5, we present our e-voting system based on blockchain. Chapter 6 concludes this thesis and discusses potential research directions of future work.

Chapter 2

Preliminaries

In this chapter, we provide the preliminaries which will be utilized in the subsequent chapters. Specifically, we begin by defining notations and outlining the underlying cryptographic assumptions. Following this, we describe the syntax and security requirements necessary for various schemes.

2.1 Notations and Cryptographic Assumptions

We use λ to denote the security parameter and $\mathcal{A}$ to denote a PPT adversary as an interactive probabilistic polynomial time Turing Machine, whose running time is polynomial in λ. We use $r \leftarrow R$ and $r \xleftarrow{\$} R$ interchangeably to denote that r is randomly picked from a finite set R.

2.1.1 DL Assumption on Composite Group

Let *Setup* be an algorithm outputting $(\mathbb{G}, N^2, g)$, with input 1^λ. $\mathbb{G}$ is the description of a finite cyclic group with composite order N^2, where $N = pq$ is a RSA modulus, $\left|N^2\right| = \lambda$ and g is the

generator[1].

Definition 2.1.1. (Discrete Logarithm Relation Assumption on Composite Group). *This assumption holds if for all $n \geq 1$ and any non-uniform PPT adversaries $\mathcal{A}$,*

$$\Pr\left[\begin{array}{cc} \exists a_i \neq 0 & (\mathbb{G}, N^2, g) \leftarrow Setup(1^\lambda), \\ \text{and} & g_1, \ldots, g_n \leftarrow \mathbb{G}\ , \\ g^{a_0}\prod_{i=1}^{n} g_i^{a_i} = 1 & a_0, \ldots, a_n \leftarrow \mathcal{A}(\mathbb{G}, N^2, g, \{g_i\}_i) \end{array}\right] \approx 0,$$

relative to Setup. We say $g^{a_0}\prod_{i=1}^{n} g_i^{a_i} = 1$ a non-trivial discrete log relation between $g_1, \ldots, g_n$. It is known that this assumption is equivalent to the discrete logarithm assumption.

2.1.2 DCR Assumption

For the decisional composite residuosity assumption, roughly speaking, it means that given an RSA modulus N with an element $z \in \mathbb{Z}^*_{N^2}$, the adversary cannot distinguish (except with negligible probability) whether z is an N-th residue.

Definition 2.1.2. (Decisional Composite Residuosity (DCR) Assumption). *This assumption holds relative to the key generation algorithm* PL.Gen *in Paillier system (Def.* (2.6.1)*), if for all non-uniform PPT adversaries $\mathcal{A}$, the following holds,*

$$\Big|\Pr\left[\mathcal{A}(N, z_1) = 1 \,\middle|\, (N, p, q) \leftarrow \mathsf{PL.Gen}(1^\lambda), r_1 \leftarrow \mathbb{Z}^*_N, z_1 = r_1^N \mod N^2\right] - \Pr\left[\mathcal{A}(N, z_2) = 1 \,\middle|\, (N, p, q) \leftarrow \mathsf{PL.Gen}(1^\lambda), r_2 \leftarrow \mathbb{Z}^*_{N^2}, z_2 = r_2\right]\Big| \approx 0.$$

[1] To find such a group G, one can use methods specified in Chapter 4

2.1.3 GDH Groups

Let $\mathbb{G}$ be a cyclic group with prime order q and g is its generator. Let $a, b \leftarrow \mathbb{Z}_q^*$ be two random chosen elements from $\mathbb{Z}_q^*$. We consider the following two problems.

- **Computational Diffie-Hellman (CDH) Problem.** Given a triple (g, g^a, g^b) in group $\mathbb{G}$, find the element g^{ab}.
- **Decisional Diffie-Hellman (DDH) Problem.** Given a quadruple (g, g^a, g^b, g^c) in group $\mathbb{G}$, decide whether $c = ab$.

We call groups like $\mathbb{G}$ if DDH problem can be solved in polynomial time but no probabilistic algorithm can solve CDH problem, except with negligible advantage, the Gap Diffie-Hellmen (GDH) groups.

2.2 Key Encapsulation Mechanism

Here we review the syntax and the security requirements of a KEM scheme.

Definition 2.2.1 (*KEM*)**.** *A standard key encapsulation mechanism (*KEM*) consists of the following three PPT algorithms.*

- ***KeyGen:*** *The randomized key generation algorithm returns public/secret key pair (pk, sk) with input 1^λ, where λ is a security parameter. This algorithm can be expressed as, $KeyGen(1^\lambda) \rightarrow (pk, sk)$.*

- ***Encap:*** *The encapsulation algorithm takes public key as input, returning key K with its encapsulation C. It can be written as,* $Encap(pk) \rightarrow (K, C) \in K_{pk} \times C_{pk}$.

- ***Decap:*** *The decapsulation algorithm takes secret key sk and encapsulation C as input. It returns corresponding key K or outputs* $\perp$ *to indicate invalid encapsulation. It can be written as,* $Decap(C, sk) = K$ *or* $\perp$.

Definition 2.2.2 (*2-Phase KEM*)**.** *We call a KEM scheme as a 2-phase KEM if its Encap algorithm can be divided into the following two phases.*

- ***Encap***^{1} : *It will first choose a random value* $w \xleftarrow{\$} Q$ *and output C, it can be written as,* $Encap^1(w) \rightarrow C$.

- ***Encap***^{2} : *In the second phase, it takes C, public key pk and w as input. It finally returns K, whose encapsulation is C. It can be written as,* $Encap^2(C, pk, w) \rightarrow K$.

Definition 2.2.3 (*Security of KEM*)**.** *We call a* KEM *scheme is* (t, ϵ_{cpa})*-CPA (resp.* (t, q_d, ϵ_{cca})*-CCA) secure if there does not exist such a PPT adversary who can win the following game in time t with at least* ϵ_{cpa} *(resp.* ϵ_{cca}*) advantage (resp. after making* q_d *decryption queries). The game between a challenger* $\mathcal{C}$ *and an adversary* $\mathcal{A}$ *is as follows.*

1. **Setup**: By inputing security parameter λ, challenger $\mathcal{C}$ generates a pair of keys $(pk, sk) \leftarrow KeyGen(1^\lambda)$ and gives pk to the adversary $\mathcal{A}$.

2. **Phase 1** (Only in CCA game): In this phase, adversary $\mathcal{A}$ submits a string C_i to decapsulation oracle $\mathcal{O}_{dec}$. The oracle will return decapsulation result $dec_{sk}(C_i)$.

3. **Challenge**: In the challenge phase, $\mathcal{A}$ issues encapsulation queries to $\mathcal{C}$. Encapsulation oracle $\mathcal{O}_{enc}$ randomly selects $b \in \{0,1\}$ and computes $(C^*, K^*) \leftarrow Encap(pk)$. Challenger $\mathcal{C}$ will return (C^*, K^*) if $b = 0$; otherwise, it will return (C^*, K') where $K' \xleftarrow{\$} \{0,1\}^{|K^*|}$. ($C^*$ is called target ciphertext)

4. **Phase 2** (Only in CCA game): Phase 2 is the same as Phase 1 with the restriction that submitted encapsulation query C_i should not be identical to C^*.

5. **Guess**: $\mathcal{A}$ outputs a guess b' of b and wins the game if $b' = b$. The advantage of $\mathcal{A}$ in winning this game is defined as

$$\epsilon_{cpa}(resp.\ \epsilon_{cca}) = 2(\Pr[b' = b] - \frac{1}{2}).$$

   The scheme is secure if ϵ_{cpa} (resp. ϵ_{cca}) is negligible.

2.3 Pseudorandom Function

Definition 2.3.1 (*PRF*)**.** *Assuming that the inputs of the* pseudorandom function *(PRF) we considered here can be arbitrary. Let* $\{0,1\}^l$ *be its output. Let* $\mathbf{F} = \{PRF_\lambda\}_{\lambda \in N}$ *be a function set such that any variable* PRF_λ *assumes values in the set of* $\{0,1\}^* \rightarrow \{0,1\}^l$. **F**

is called an efficiently computable pseudorandom function ensemble if

1. *(*efficient computation*) I and V are PPT algorithms and there is a mapping function ϕ, mapping from strings to functions, such that $\phi(I(1^\lambda))$ and PRF_λ are identically distributed and $V(i,x) = (\phi(i))(x)$.*

2. *((t, ϵ_{prf})-*pseudorandomness*) For any PPT distinguisher $\mathcal{D}$, he can not distinguish a PRF function to a real random function with negligible probability.*

$$\left|\Pr[\mathcal{D}^{PRF_\lambda}(1^\lambda) = 1] - \Pr[\mathcal{D}^{RF_k}(1^\lambda) = 1]\right| < \epsilon_{prf}$$

where $\mathbf{R} = \{RF_k\}_{\lambda \in N}$ is the set involving RF_k. RF_k is uniformly distributed over $\{0,1\}^* \to \{0,1\}^l$.

2.4 Designated Verifier Signature

Definition 2.4.1 (*DVS*)**.** *A designated verifier signature (DVS) consists of the following three PPT algorithms.*

- ***KG:*** *The key generation algorithm takes 1^λ as input where λ is security parameter, followed by returning a public/secret key pair $(\mathrm{pk}, \mathrm{sk})$. This algorithm can be written as, $\mathrm{KG}(1^\lambda) \to (pk, sk)$.*

- ***Sign:*** *The signing algorithm takes message m, signer's public and secret keys (pk_s, sk_s) and designated verifier's public key*

pk_v as input. It will return signature σ of message m, which can be written as $\mathrm{Sign}(sk_s, pk_s, pk_v, m) \rightarrow \sigma$.

- ***Ver:*** *The verification algorithm takes signature σ, corresponding message m, verifier's public and secret keys (sk_v, pk_v) and signer's public key pk_s as input. It will output 1 if it is a valid signature, otherwise it will output 0. It can be written as* $Ver(sk_v, pk_v, pk_s, m, \sigma) \rightarrow b$ *(*b *is 1 if the signature is valid, otherwise* b *is 0).*

Correctness: The correctness of DVS requires that for any $KG(1^\lambda) \rightarrow (pk_s, sk_s)$, $KG(1^\lambda) \rightarrow (pk_v, sk_v)$ and any message $m \in \{0,1\}^*$, we have the following,

$$\Pr[Ver(sk_v, pk_v, pk_s, m, Sign(sk_s, pk_s, pk_v, m)) = 1] = 1.$$

2.5 Pedersen Commitment

Definition 2.5.1. (Pedersen Commitment Scheme). *A Pedersen commitment scheme contains 2 polynomial algorithms,* $\mathsf{PDC} = (\mathsf{PDC.Gen}, \mathsf{PDC.Com})$.

- $\mathsf{PDC.Gen}(1^\lambda) \rightarrow ck$. *With security parameter λ , the key generation algorithm outputs commitment key, ck, where $ck = (\mathbb{G}, g, h)$. $\mathbb{G}$ is a cyclic group of composite order satisfying Def.2.1.1, g is its generator and h is a randomly chosen element in this group where their discrete log are not known to the prover. Then ck specifies message space $\mathcal{M}_{ck}$, randomness space $\mathcal{R}_{ck}$ and ciphertext space $\mathcal{C}_{ck}$.*

- $\mathsf{PDC.Com}(ck, m) \rightarrow ct$. *On input ck and message* $m \in \mathcal{M}_{ck}$, *the algorithm randomly selects* $r \in \mathcal{R}_{ck}$ *and outputs commitment as,* $ct = g^m \cdot h^r \mod Q$, *s.t.,* $Q = f \cdot N^2 + 1$ *is a prime where* f *is a random small integer.*

In our protocol, what we used is a variant of the above form. The commitment key is set as $ck = (\mathbb{G}, g, g_1, g_2, \ldots, g_n)$ and a commitment of a series of messages, $(m_1, \ldots, m_n)$, has the form , $ct = g^r \cdot \Pi_{i=1}^{n} g_i^{m_i}$

2.6 Paillier Encryption

In the Paillier encryption, we utilize p and q of equal length, setting $N = pq$ and choosing $(1 + N)$ as the base.

Definition 2.6.1. (Paillier Encryption Scheme). *A Paillier cryptosystem,* PL, *contains 3 polynomial time algorithms,* (PL.Gen, PL.Enc, PL.Dec).

- $\mathsf{PL.Gen}(1^\lambda) \rightarrow (pk, sk)$. *With security parameter* λ *as input, the key generation algorithm outputs* $pk = N$ *and* $sk = (p, q)$. *They satisfy* $|p| = |q| = \lambda$ *(except with negligible probability in* λ*) and* $N = pq$.

- $\mathsf{PL.Enc}(pk, m) \rightarrow c$. *To encrypt a message* $m \in \mathbb{Z}_N$ *with respect to public key* N, *one chooses a randomness,* $r \leftarrow \mathbb{Z}_N^*$, *and outputs the ciphertext as,*

$$c = (1 + N)^m \cdot r^N \mod N^2.$$

When emphasizing randomness r, we sometimes write it as, $c = \mathsf{PL.Enc}_{\mathsf{pk}}(m; r)$ or $c = \mathsf{PL.Enc}(m; r)$ for simplicity.

- $\mathsf{PL.Dec}(sk, c) \to m$. *To decrypt a ciphertext c with $sk = (p, q)$, one can first compute $\lambda = lcm(p-1, q-1)$. Denote function $L(x)$ as, $L(x) = \frac{x-1}{N}$. Then he computes m as,*

$$m = \frac{L(c^{\lambda} \mod N^2)}{L((1+N)^{\lambda} \mod N^2)} \mod N.$$

Readers can refer to [Pai99] for its correctness proof and the security of this system follows based on DCR assumption in Definition 2.1.2.

2.7 Zero-knowledge Argument of Knowledge (ZKAoK)

We aim to construct a zero-knowledge argument of knowledge. Informally, a zero-knowledge proof of knowledge is a protocol, involving a prover and a verifier, where the prover can convince the verifier the truth of some statements without leaking any further secret information it holds. Typically, the system is referred to as an "argument" (i.e., ZKAoK) when its soundness property holds against computationally bounded adversaries, and as a "proof" (i.e., ZKPoK) when its soundness is against unbounded adversaries[2]. We now present the formal definitions.

In an argument, we consider two interactive algorithms, $(\mathcal{P}, V)$, which both run in probabilistic polynomial time. We use $tr \leftarrow \langle \mathcal{P}(s), V(t) \rangle$ to denote the transcript produced by $\mathcal{P}$ and V when

[2]Our protocol is actually an argument, and we do not specifically distinguish between these two notions in our paper.

interacting on their inputs s and t. We write $\langle \mathcal{P}(s), V(t) \rangle = b$ depending on whether verifier rejects, $b = 0$, or accepts, $b = 1$. Let $\mathcal{R} \in \{0,1\}^* \times \{0,1\}^*$ be a polynomial-time-decidable binary relation. For a statement u, we call w a witness of it if $(u, w) \in \mathcal{R}$.

Definition 2.7.1. (Argument of knowledge). *The pair $(\mathcal{P}, V)$ is called an argument of knowledge for relation $\mathcal{R}$, if it satisfies perfect completeness and statistical witness-emulation defined as below.*

Definition 2.7.2. (Perfect completeness). $(\mathcal{P}, V)$ *achieves perfect completeness if for all non-uniform polynomial time adversaries $\mathcal{A}$,*

$$\Pr\left[\ (u,w) \notin \mathcal{R} \text{ or } \langle \mathcal{P}(u,w), V(u) \rangle = 1 \quad (u,w) \leftarrow \mathcal{A}(1^\lambda) \ \right] = 1$$

Definition 2.7.3. (Statistical witness-extended emulation). *$(\mathcal{P}, V)$ has statistical witness-extended emulation if for all deterministic polynomial time $\mathcal{P}^*$, there exists an expected polynomial time emulator $\mathcal{E}$ such that for all interactive adversaries $\mathcal{A}$ we have,*

$$\Pr\left[\ \mathcal{A}(tr) = 1 \quad (u,s) \leftarrow \mathcal{A}(1^\lambda), tr \leftarrow \langle \mathcal{P}^*(u,s), V(u) \rangle \ \right] \approx$$
$$\Pr\left[\begin{array}{cc} \mathcal{A}(tr) = 1 \text{ and} & (u,s) \leftarrow \mathcal{A}(1^\lambda), \\ \text{if } tr \text{ is accepting then } (u,w) \in \mathcal{R} & (tr,w) \leftarrow \mathcal{E}^{\langle \mathcal{P}^*(u,s), V(u) \rangle}(u) \end{array} \right]$$

where $\mathcal{E}$ has access to oracle $\mathcal{O} =< \mathcal{P}^*(u,s), V(u) >$ which can be rewind to a specific point and resume with verifier picking fresh public coin challenges from this point onwards.

To define soundness, we apply the term, witness-extended emulation, which is used in [BCC+16] and defined in [GI08,

Lin03]. Witness-extended emulation implies both soundness and knowledge soundness. Intuitively, it means that given an adversary who can produce an acceptable argument with some probability, there exists an emulator who can produce a similar argument with the same probability, but can also produce a witness at the same time. In the definition, the value s can be considered as the internal state of $\mathcal{P}^*$, including randomness. Whenever $\mathcal{P}^*$, in state s, can make a convincing argument, $\mathcal{E}$ can extract a witness. Therefore, we obtain an argument of knowledge of w, such that $(u, w) \in \mathcal{R}$.

Definition 2.7.4. (Public coin). *An argument of knowledge $(\mathcal{P}, V)$ is called public coin if all messages sent from the verifier to the prover are chosen uniformly at random and independently of messages sent by the prover, i.e., the challenge values correspond to the verifier's randomness ρ.*

An argument of knowledge is zero-knowledge if it does not leak any other information about w beyond what can be deduced from the truth that $(u, w) \in \mathcal{R}$. We will present arguments of knowledge that have special honest verifier zero-knowledge. That is, given the verifier's challenge values in advance, it is possible to simulate the entire argument without knowing the witness.

Definition 2.7.5. (Perfect special honest verifier zero-knowledge). *A public coin argument of knowledge $(\mathcal{P}, V)$ is called a perfect special honest verifier zero knowledge (SHVZK) argument of knowledge for*

all $\mathcal{R}$ if there exists a probabilistic polynomial time simulator $\mathcal{S}$ such that for all interactive non-uniform polynomial time adversaries $\mathcal{A}$,

$$\Pr\left[\begin{array}{ll} (u,w)\in\mathcal{R} \text{ and} & (u,w,\rho)\leftarrow\mathcal{A}(1^\lambda), \\ \mathcal{A}(tr)=1 & tr\leftarrow\langle\mathcal{P}(u,w),V(u,\rho)\rangle \end{array}\right]$$

$$=\Pr\left[\begin{array}{ll} (u,w)\in\mathcal{R} \text{ and} & (u,w,\rho)\leftarrow\mathcal{A}(1^\lambda), \\ \mathcal{A}(tr)=1 & tr\leftarrow\mathcal{S}(u,\rho) \end{array}\right]$$

where ρ is the public coin randomness used by the verifier.

2.8 Threshold Blind Signature

The (t, n)-threshold blind signature [VZK02], based on GDH hard problem, containing the following four algorithms, namely, Setup algorithm TBU, Key Generation algorithm TBK, Signature Generation algorithm TBS and Signature Verification algorithm TBV, where TBK and TBS are two interactive algorithms. Let n players in this protocol denoted as $\{L_1, L_2, ..., L_n\}$.

1. *Setup Algorithm* TBU

   On input security parameter 1^λ, this algorithm outputs public parameters $\mathsf{param} = (\mathbf{G_1}, G_T, q, P, H, \hat{e})$. $\mathbf{G_1}$ is a GDH group with order q, P is its generator and $H : \{0,1\}^* \rightarrow \mathbf{G_1}$ denotes a one-way function. G_T denotes a pairing group and $\hat{e}$ denotes the pairing operation, i.e., $\hat{e}(\mathbf{G_1}, \mathbf{G_1}) \rightarrow G_T$. Looking ahead, these parameters will be uploaded to the blockchain in our e-voting system. It is an implicit input to the following algorithms.

2. *Key Generation Protocol* TBK

   The interactions between the players, $\{L_i\}_{i=1}^n$, are described as follows.

   - L_i conducts the following computations.
     (a) Picks up the parameters a_{ij} $(j = 0, 1, 2, .., t-1)$ randomly in the following polynomial $f_i(x)$:

     $$f_i(x) = a_{i0} + a_{i1}x + ... + a_{i,t-1}x^{t-1}.$$

     (b) Computes and broadcasts $P^{a_{ij}}$ for $j = 0, 1, ..., t-1$; sends $f_i(j)$ to each player L_j for $j = 1, 2, ..., n$; $j \neq i$.
   - L_i receives the information from other players.
     (a) After receiving $f_j(i)$ from L_j for $j = 1, 2, ..., n$; $j \neq i$, player L_i verifies if

     $$P^{f_j(i)} = \prod_{k=0}^{t-1} P^{a_{jk} \cdot i^k}.$$

     If the check fails, L_i broadcasts a complaint against L_j.
     (b) L_i computes the secret share $s_i = \sum_{k=1}^n f_k(i)$ and the public share $Q_i = P^{s_i}$, which will be broadcasted to other players. The public key in this algorithm will be set as $Q = \prod_{i=1}^n P^{a_{i0}}$, which can be computed using Q_i.

   After executing TBK protocol, the public key is set as $Q = P^s$ where the secret key is $s = \sum_{i=0}^n a_{i0}$, which is distributed

to the n players but does not appear explicitly in the protocol.

3. *Signature Generation Protocol* TBS

   This protocol allows user A to obtain a blind signature on message m from t signers. Let $S = \{L_i | 1 \leq i \leq t\}$ denote the set of t signers. For the ease of presentation, we use w_i to denote $\prod_{j \in S,\, j \neq i} \frac{j}{j-i}$.

   (a) User A randomly chooses $r \in \mathbb{Z}_q^*$ and blinds message m by computing $m' = H(m)^r$. A sends m' to every signer $L_i \in S$.

   (b) Signer L_i computes and sends σ_i to user A's address on the blockchain after receiving m', where

   $$\sigma_i = m'^{w_i s_i}.$$

   (c) User A validates σ_i by checking if the following equation holds.

   $$\hat{e}(\sigma_i, P) \stackrel{?}{=} \hat{e}(m'^{w_i}, Q_i).$$

   If this does not hold, A sends m' to the signer L_i again. Otherwise, A computes the signature σ on m as,

   $$\sigma = (\prod_{i \in S} \sigma_i)^{-r}.$$

4. *Key Verification Algorithm* TBV

   The signature σ on the message m is accepted if

$$\hat{e}(\sigma, P) = \hat{e}(H(m), Q).$$

2.9 Threshold ElGamal Decryption Scheme

The (t*,n*)-threshold ElGamal decryption scheme [Ped91] consists of the following four algorithms, namely, TEU, TEK, TEC, TED, where TEK and TED are interactive algorithms. It is based on the DDH problem. Let $\{T_1, T_2, ..., T_{n^*}\}$ denote the set of n^* players.

1. *Setup Algorithm* TEU:

   On input security parameter 1^λ, this algorithm outputs public parameters param $= (\mathbf{G_1}, G_T, q, P, \hat{e})$, where $\mathbf{G_1}$ is an elliptic curve group of order q with P as its generator. G_T is a pairing group and $\hat{e}$ denotes the pairing operation, i.e., $\hat{e}(\mathbf{G_1}, \mathbf{G_1}) \rightarrow G_T$. Looking ahead, param will be uploaded to blokchain and it is an implicit input of the following algorithms.

2. *Key Generation Protocol* TEK:

   Same as the TBK protocol of the threshold blind signature scheme, each user T_i randomly selects a_{ij} in the $(t^* - 1)$ degree polynomial $f_i(x)$, where

$$f_i(x) = a_{i0} + a_{i1}x + ... + a_{i,t^*-1}x^{t^*-1},$$

and sends $f_i(j)$ to T_j. The public share of T_i is $Q_i^* = P^{s_i^*}$, which will be broadcasted. Its corresponding secret is $s_i^* = \sum_{k=1}^{n^*} f_k(i)$. The resulting public key is set as $Q^* = \prod_{i=1}^{n^*} P^{a_{i0}}$ and secret key is $s^* = \sum_{i=0}^{n^*} a_{i0}$. The whole algorithm can be written as: $\mathsf{TEK} \rightarrow (Q^*, s^*, Q_i^*, s_i^*)$, for $i \in \{1, 2, .., n^*\}$.

3. *Encryption Algorithm* TEC:

   On input with message m and public key Q^*, the ciphertext is computed as,

$$C = (\mathbf{c}_1, \mathbf{c}_2) = (P^k, mQ^{*k}),$$

   where k is a random number: $k \xleftarrow{\$} \mathbb{Z}_q$.

4. *Decryption Protocol* TED:

   The decryption protocol takes ciphertext C as input. t^* players in list $S^* = \{T_i | 1 \leq i \leq t^*\}$ decrypt the ciphertext as follows together.

   (a) Each player computes and broadcasts $m_i = \mathbf{c}_1^{-w_i s_i}$, where

$$w_i = \prod_{j \in S^*,\, j \neq i} \frac{j}{j - i}.$$

   (b) After receiving m_i, verifies if

$$\hat{e}(m_i, P) = \hat{e}(\mathbf{c}_1^{-w_i}, Q_i).$$

   If the above equation does not hold, broadcasts a complaint on T_i.

(c) Finally, the resulting decrypted message is computed as

$$\mathbf{M} = \mathbf{c}_2 \cdot \prod_{i=1}^{t^*} m_i.$$

Correctness The decrypted message, $\mathbf{M}$, has the form,

$$\begin{aligned}\mathbf{M} = \mathbf{c}_2 \cdot \prod_{i=1}^{t^*} m_i &= \mathbf{c}_2 \cdot \prod_{i=1}^{n^*} \mathbf{c}_1^{-w_i s_i} \\ &= \mathbf{c}_2 \cdot \prod_{i=1}^{n^*} \mathbf{c}_1^{-\frac{j}{j-i} \cdot s_i} \\ &= \mathbf{c}_2 \cdot \mathbf{c}_1^{-s^*}\end{aligned}$$

When we derive the above result, we use the Lagrange interpolation and we can see that the decrypted message is correct.

Chapter 3

Strong Designated Verfiable Signature from Key Encapsulation Mechanism

The notion of strong designated verifiable signature (SDVS) was originally formalized by Saeednia et al. [SKM03b] to enhance the privacy of the signer in the (designated verifiable signature) DVS scheme. It allows the signer to generate a signature for a designated verifier without allowing the verifier to transfer the signature to any third party. Additionally, SDVS ensures that no third party can distinguish which party generated the signature without the verifier's secret key. However, existing constructions of SDVS rely on specific assumptions, and they only analyze their schemes considering one signer and one verifier.

We observe that existing models may not capture real attacks in practice. For instance, an adversary may have access to signatures generated for different designated verifiers. After collecting these signatures, the adversary may acquire auxiliary

information that helps him to forge the signature for the targeted verifier; Furthermore, an adversary may obtain useful information from signatures generated by other signers for the target verifier. In summary, an adversary may produce valid signatures through collaborating with dishonest signers or verifiers. However, these attacks are not captured by the existing models, where the adversary is only restricted to issue queries with respect to the target signer and verifier.

To capture more sophisticated situations that may occur in practice, an extended model is required, involving more than one target signer and verifier. With this new model, it is also necessary to construct new schemes that are suitable for this advanced model and prove their security.

Chapter Organization. We will start by providing an overview of our contributions in Section 3.1. We introduce our strengthened models in Section 3.2. In Section 3.3, we give our generic construction towards our model, based on KEM and PRF function. The security proof of our construction is covered in Section 3.4. Finally, in Section 3.5, we provide several instantiations based on different assumptions and compare them with the existing schemes.

3.1 Our Contribution

To address the above mentioned issues, we initiate the study of SDVS in the multi-user setting and propose a generic construction. Specifically, we strengthen existing models and propose two enhanced models, namely, multi-user and multi-user$^+$. In our first model (multi-user), the adversary can issue queries from given lists of signers and verifiers, and also corrupt them; In our second model (multi-user$^+$), the adversary can obtain signatures from the signer on any verifiers of its choice (i.e., the verifier's public keys are created by the adversary). We also propose a generic construction of SDVS based on *KEM* and *PRF*, which is proved to be secure in our enhanced models. In summary, we made the following contributions towards this problem,

- We proposed two enhanced security models of SDVS to model security requirements in the multi-user setting, namely, multi-user and multi-user$^+$.
- We proposed a generic construction of SDVS from *KEM* and *PRF*. We proved that our generic construction is secure, assuming the security of the underlying *KEM* and *PRF*.

Table 3.1 summarizes differences between the existing SDVS security models and our two strengthened models. Let S and V denote lists of signers and verifiers' public keys chosen by the

challenger. We can see from Table 3.1 that in the existing models, the adversary can only issue queries with respect to the specific challenge signer and verifier. In our enhanced models, the adversary can make additional queries beyond the challenge identities or can issue queries on the verifier chosen by himself adaptively. In other words, the adversary can access more information in our models than in the original model. Additionally, our models can corrupt queries, meaning that the adversary can request for private keys of any public keys in S and V except the challenge public keys (also chosen by the adversary). Therefore, our models have more stringent security requirements than the original model.

Table 3.1: Differences Between Existing Models and Our Models in SDVS.

S and V indicate signers and verifiers' public key lists chosen by the challenger. pk_s and pk_v indicate signer and verifier's public keys respectively.

	Challenge Public Keys (pk_s, pk_v)	Signature Queries (pk_s, pk_v)
Existing Model	$pk_s \in S, pk_v \in V$, $\|S\| = 1, \|V\| = 1$	$pk_s \in S, pk_v \in V, \|S\| = 1, \|V\| = 1$
Multi-user	$pk_s \in S, pk_v \in V$	$pk_s \in S, pk_v \in V$
Multi-user$^+$	$pk_s \in S, pk_v \in V$	$pk_s \in S$, no restriction on pk_v

3.2 Our Strengthened Models

In this section, we present our strengthened models, which allow the attacker to issue queries with respect to multiple verifiers, some of which may be corrupted or have keys chosen adversarially. The differences are summarized in Table 3.1. Formally, our strengthened models are defined as follows.

Definition 3.2.1 (*Unforgeability*)**.** *An SDVS scheme is* unforgeable *in multi-user (resp. multi-user$^+$) setting if no PPT adversary can forge a valid signature on a message of its choice without knowing the signer and verifier's secret key.*

The following game between challenger $\mathcal{C}$ and PPT adversary $\mathcal{A}$ formally defines unforgeability.

1. **Setup:** On input security parameter λ, $\mathcal{C}$ runs *KG* algorithm to obtain multiple signers and the verifiers' public-secret key pairs. Let $S = \{pk_{s_1}, pk_{s_2}, ..., pk_{s_m}\}$ and $V = \{pk_{v_1}, pk_{v_2}, ..., pk_{v_n}\}$ be signers and verifiers' public keys respectively. $\mathcal{A}$ is given S and V.

2. **Queries:** $\mathcal{A}$ can issue queries to the following oracles. Note that $\mathcal{A}$ can also issue a corrupt query to obtain the secret keys of signer and verifier in the lists (except the challenge public keys, i.e. pk_s^* and pk_v^*).

   - $\mathcal{O}_{sign}$: $\mathcal{A}$ can issue signing queries between signer $pk_s \in S$ and verifier pk_v.
   - $\mathcal{O}_{sim}$: $\mathcal{A}$ can request verifier pk_v to simulate signature on message m between signer $pk_s \in S$.
   - $\mathcal{O}_{ver}$: $\mathcal{A}$ can request verification queries on the pair (m, σ) on the signer $pk_s \in S$ and verifier pk_v.
   - *Restrictions:* In the multi-user setting, an additional restriction applies, namely, $pk_v \in V$ for queries to $\mathcal{O}_{sign}$,

$\mathcal{O}_{sim}$, $\mathcal{O}_{ver}$. In the multi-user$^+$ setting, pk_v can be any value chosen by $\mathcal{A}$.

3. **Forgery:** Finally, $\mathcal{A}$ outputs a forgery (m^*, σ^*) on signer and verifier from lists and wins the game if,

   - $Ver(sk_v^*, pk_v^*, pk_s^*, m^*, \sigma^*) = 1$, and
   - $\mathcal{A}$ has not issued $\mathcal{O}_{sign}$ and $\mathcal{O}_{sim}$ on input m^* on signer pk_s^* and verifier pk_v^* before.

   The probability of forging a valid signature is denoted by $\Pr[Forge]$. An SDVS scheme is unforgeable if

$$\Pr[Forge] < \epsilon(\lambda),$$

   where $\epsilon(\lambda)$ is negligible in λ.

Definition 3.2.2 (*Non-Transferability*)**.** *An SDVS scheme is non-transferable if there exists a PPT simulation algorithm Sim which takes sk_v, pk_v, pk_s and message m as input. It outputs a simulated signature that is indistinguishable from the real signature generated by the signer on the same m.*

That is, for any PPT distinguisher $\mathcal{D}$, any $(pk_s, sk_s) \leftarrow KG(1^\lambda)$, $(pk_v, sk_v) \leftarrow KG(1^\lambda)$ and any message $m \in \{0,1\}^*$, it holds that

$$\left| \Pr \left[\begin{array}{c} \sigma_0 \leftarrow \mathsf{Sign}(sk_s, pk_s, m), \\ \sigma_1 \leftarrow \mathsf{Sim}(sk_v, pk_v, m), \\ b \xleftarrow{\$} \{0,1\}, \\ b' \leftarrow D(pk_s, sk_s, pk_v, sk_v, \sigma_b) \end{array} : b' = b \right] - \frac{1}{2} \right| < \epsilon(\lambda)$$

where $\epsilon(\lambda)$ is a negligible function with security parameter λ. The random coins consumed by $\mathcal{D}$ and the probability takes over the randomness used in *KG*, *Sign* and *Sim*. If the probability is equal to $\frac{1}{2}$, we say that the SDVS scheme is *perfectly non-transferable*.

Definition 3.2.3 (*Privacy of Signer's Identity*)**.** *We call a scheme that satisfies* privacy of signer's identity *in the multi-user (resp. multi-user$^+$) setting if a third party cannot tell whether the signature generated by signer S_0 or by signer S_1 correctly without knowing signer's and verifier's secret key.*

The game below, which is played by a challenger $\mathcal{C}$ and a distinguisher $\mathcal{D}$, formally defines privacy of signer's identity in the multi-user setting. Let S and V denote the lists of signers and verifiers' public keys generated by $\mathcal{C}$, same as the unforgeability game.

1. **Setup:** $\mathcal{C}$ generates public and secret keys for signers and verifiers. The corresponding public key lists, namely, S and V, are given to distinguisher $\mathcal{D}$.

2. **Queries:** $\mathcal{D}$ can adaptively issue $\mathcal{O}_{sign}$, $\mathcal{O}_{sim}$ and $\mathcal{O}_{ver}$ queries on signer pk_{s_i} and verifier pk_{v_i}, same as in the unforgeability game. $\mathcal{D}$ can also corrupt secret keys on signer and verifier from the lists.

3. **Challenge:** $\mathcal{D}$ chooses two signers, e.g. S_0^* and S_1^*, from S and one verifier pk_v^* from V to be the challenge identities.

$\mathcal{D}$ submits a message m^* and $\mathcal{C}$ tosses a coin $b \in \{0,1\}$ and computes challenge signature $\sigma^* \leftarrow \text{Sign}(sk^*_{s_b}, pk^*_{s_b}, pk^*_v, m^*)$. $\mathcal{C}$ then returns σ^* to $\mathcal{D}$.

4. **Queries:** $\mathcal{D}$ continues to issue queries as in step 2 with the restriction that no verification queries on $(m^*, \sigma^*, pk^*_{s_i})$ for any $pk^*_{s_i} \in \{S^*_0, S^*_1\}$. Note that $\mathcal{D}$ cannot corrupt challenge identities' secret keys.

5. **Guess:** Finally, $\mathcal{D}$ outputs a guess b' of b and wins the game if $b' = b$. The probability of $\mathcal{D}$ in winning this game is defined as $\Pr[PSI]$. An SDVS scheme possesses *PSI* if

$$\left| \Pr[PSI] - \frac{1}{2} \right| < \epsilon(\lambda),$$

where $\epsilon(\lambda)$ is negligible.

Definition 3.2.4 (*SDVS*)**.** *An SDVS scheme is secure in the multi-user (resp. multi-user$^+$) setting if it possesses unforgeability, non-transferability and privacy of signer's identity.*

3.3 Our Construction

In this section, we give our generic construction towards our strengthened models. We first give a high-level overview, followed by a detailed description.

3.3.1 Overview of Our Construction

Our generic construction of SDVS relies on *KEM* and *PRF*, where *KEM* must be 2-phase as discussed in Definition 2.2.2. This is not too restrictive since most *KEM* schemes can satisfy this requirement. Our generic construction is secure in the multi-user setting (resp. multi-user$^+$ setting) if *PRF* is secure and the underlying *KEM* is IND-CPA secure (resp. IND-CCA secure). Below we give a high-level description of our generic construction.

In our construction, we use *KeyGen* in *KEM* to generate signer's keys, i.e. $(pk_s, sk_s) \leftarrow KeyGen(1^\lambda)$. We use the first phase in *KEM*'s *Encap*, $C \leftarrow Encap^1(w)$, to generate verifier's keys, i.e. $pk_v \leftarrow C$, $sk_v \leftarrow w$ (w is the randomness used in $Encap^1$). To sign a message, the signer uses his secret key to decapsulate C to obtain the session key, i.e. $K_{sv} \leftarrow Decap(C, sk_s)$. He then uses this key in *PRF* to sign message m, i.e. $\sigma \leftarrow PRF_{K_{sv}}(m)$. For verification, verifier executes the second phase in *Encap* to acquire the same session key, i.e. $K_{sv} \leftarrow Encap^2(C, pk_s, sk_v := w)$, followed by checking $\sigma \stackrel{?}{=} PRF_{K_{sv}}(m)$.

3.3.2 Details of Our Generic Construction

Given a *KEM* scheme $K = (K.KeyGen, K.Encap, K.Decap)$, which is a 2-phase encapsulation mechanism, and a *PRF* function, we can construct a secure SDVS scheme $D = (D.KG, D.Sign, D.Ver)$. The construction is as follows.

1. ***D.KG:*** The key generation algorithm takes 1^λ as input where λ is the security parameter. It invokes *KeyGen* in *KEM* to obtain signer's keys, namely, $K.KeyGen(1^\lambda) \rightarrow (D.pk_s, D.sk_s)$. It also invokes the first phase in *Encap* to obtain verifier's keys, namely, $K.Encap^1(w) \rightarrow C$, $(C, w) \rightarrow (D.pk_v, D.sk_v)$.

2. ***D.Sign:*** The signing algorithm takes the signer's keys, the verifier's public key and the message as input, namely, $(D.sk_s, D.pk_s, D.pk_v, m)$. It first runs the decapsulation algorithm in *KEM* to obtain the key, i.e. $K.Decap(D.pk_v, D.sk_s) \rightarrow K_{sv}$. It then takes key K_{sv} and message m into *PRF* algorithm with returning signature σ, $PRF_{K_{sv}}(m) \rightarrow \sigma$. The signing algorithm can be written as, $D.Sign(D.sk_s, D.pk_s, D.pk_v, m) \rightarrow \sigma$.

3. ***D.Ver:*** The verification algorithm takes the verifier's keys, public key of signer, the message and signature as input, namely, $(D.sk_v, D.pk_v, D.pk_s, m, \sigma)$. It first runs the second phase of encapsulation algorithm in *KEM* to compute key K_{sv}, namely, $K.Encap^2(D.pk_v, D.pk_s, D.sk_v) \rightarrow K_{sv}$. It will then invoke K_{sv} and message m into *PRF* to obtain its signature σ'. If $\sigma' = \sigma$, it returns 1; otherwise, it returns 0. The whole verification algorithm can be written as, $D.Ver(D.sk_v, D.pk_v, D.pk_s, m, \sigma) \rightarrow b$.

3.4 Security Analysis of Our Construction

In this section, we give security analysis of the above generic construction. We prove that our SDVS scheme is secure if the underlying *KEM* and *PRF* schemes are secure.

Theorem 3.4.1. *If the underlying KEM scheme is IND-CPA (resp. IND-CCA) secure and PRF function achieves pseudoramdomness, then we can construct a secure SDVS scheme in the multi-user setting (resp. multi-user$^+$ setting).*

The proof of Theorem 3.4.1 is divided into the proof of the following three lemmas, which stated that our generic construction possesses unforgeability, non-transferability and privacy of signer's identity.

Lemma 3.4.1. *If the underlying KEM scheme is IND-CPA (resp. IND-CCA) secure and PRF is a pseudorandom function, then our constructed scheme D achieves the property of unforgeability (in the multi-user setting) (resp. multi-user$^+$ setting). That is,* $\Pr^{SDVS}_{\mathcal{A},D}[Forge]$ *is negligible.*

Lemma 3.4.2. *If the underlying KEM is IND-CPA (resp. IND-CCA) secure and PRF achieves pseudorandomness, our constructed scheme D is perfectly non-transferable.*

Lemma 3.4.3. *If the underlying KEM scheme is IND-CPA (resp. IND-CCA) secure and PRF is a pseudorandom function, then our constructed scheme D (with multi-user setting) (resp. multi-user$^+$ setting) achieves the property of privacy of signer's identity. That is,*

$\Pr^{SDVS}_{\mathcal{A},D}[PSI]$ *is no larger than* $\frac{1}{2}$*, indicating that the adversary has no advantage compared with randomly guessing the bit.*

As we will see in the proof of Lemma 3.4.2, the scheme is perfectly non-transferable so the queries to $\mathcal{O}_{sim}$ can be perfectly handled by $\mathcal{O}_{sign}$ in the game of unforgeability and privacy of signer's identity. Hence, we only consider signing and verification queries in these two games.

Proof. (of Lemma 3.4.1) For any PPT forger $\mathcal{A}$, $\Pr[Forge]$ in the multi-user (resp. multi-user$^+$) setting is negligible assuming *KEM* scheme is IND-CPA (resp. IND-CCA) secure and *PRF* achieves pseudorandomness.

We prove this lemma by using a sequence of games played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. Let G_i denote the i-th game and X_i imply the event that $\mathcal{A}$ outputs a valid forgery in game G_i. Let S and V denote two lists for signers and verifiers, with m and n entities respectively.

G_0: This game is with multi-user setting (resp. multi-user$^+$). Challenger $\mathcal{C}$ invokes adversary with S and V lists. Adversary $\mathcal{A}$ can issue signing and verification queries on the signer and verifier from the lists (resp. no restrictions on the verifier in multi-user$^+$). We can have that,

$$\Pr^{SDVS}_{\mathcal{A},D}[Forge(multi\text{-}user)](resp.\ multi\text{-}user^+) = \Pr[X_0]. \tag{3.1}$$

G_1: In this game, the key used in *PRF* between challenge identities pk_s^* and pk_v^* is randomly chosen, i.e. $K' \xleftarrow{\$} N$. When $\mathcal{A}$ issues signing and verification queries between them, $\mathcal{C}$ uses this key K' to response. The only difference between this game and game 0 is the key used in *PRF*. If the adversary can distinguish these two games, we can construct an adversary $\mathcal{A}_1$ to break the IND-CPA (resp. IND-CCA) game in *KEM*. Therefore, we have,

$$|\Pr[X_1] - \Pr[X_0]| \leq mn \cdot \epsilon_{cpa}(resp.\ \epsilon_{cca}). \tag{3.2}$$

We construct adversary $\mathcal{A}_1$ in CPA (resp. CCA) game where $\mathcal{A}_1$ is given challenge ciphertext (C^*, K^*) and public key pk^*. $\mathcal{A}_1$ will simulate the game for the adversary in SDVS. He randomly guesses a signer-verifier pair from the two lists as challenge identities and sets, $pk_s^* = pk^*$, $pk_v^* = C^*$. The key used between these identities is K^*. Note that $\mathcal{A}_1$ will abort if he cannot guess the challenge identities correctly.

- In the multi-user$^+$ setting, $\mathcal{A}$ can issue queries on verifier pk_{v_i} beyond the V list. To response this query, $\mathcal{A}_1$ will make decapsulation queries on $C_i \leftarrow pk_{v_i}$ in CCA game (under pk_s^*) and get the corresponding key K_i. He then uses this key to response signing and verification queries.

If the SDVS's adversary successfully forges the signature, $\mathcal{A}_1$ outputs 0, indicating that K^* is the correct shared key; Otherwise $\mathcal{A}_1$ randomly outputs a bit b'. Hence his probability to

win the game is $mn \cdot \epsilon_{cpa}$ (resp. $mn \cdot \epsilon_{cca}$). Therefore, the difference between game 0 and game 1 is equal to mn times the advantage that $\mathcal{A}_1$ can distinguish them in CPA (resp. CCA) game.

G_2: In this game, we replace *PRF* with a truly random function. It means that the signature is randomly chosen from $\{0,1\}^l$ in this game. We have,

$$|\Pr[X_2] - \Pr[X_1]| \leq \epsilon_{prf}. \tag{3.3}$$

To obtain the above equation, we construct an adversary $\mathcal{A}_2$ to break the pseudorandomness of *PRF* with advantage ϵ_{prf}. Given an oracle function $F(\cdot)$ which is either a pseudorandom function chosen from **F** or a truly random function. Here, $\mathcal{A}_2$ maintains a table T, which is initially empty.

When responding to signing queries on m_i between pk_s^* and pk_v^*, $\mathcal{A}_2$ returns σ_i if (m_i, σ_i) exists in T; Otherwise, $\mathcal{A}_2$ submits message m_i to function F and returns σ_i, followed by storing it in table. When responding to verification queries on (m_i, σ_i), $\mathcal{A}_2$ will just return $\sigma_i' \stackrel{?}{=} \sigma_i$ if m_i exists in the table with σ_i'; Otherwise, $\mathcal{A}_2$ forwards message m_i to function F with obtaining a signature σ_i'. He then returns $\sigma_i' \stackrel{?}{=} \sigma_i$ to adversary $\mathcal{A}$ with storing (m_i, σ_i') in table T. The pseudorandom function is deterministic so that our simulation is perfect. Finally, $\mathcal{A}$ outputs a forgery (m^*, σ^*) on pk_s^* and pk_v^*. $\mathcal{A}_2$ submits m^* to function F with obtaining $\sigma^{*\prime}$ and outputs 1 if $\sigma^{*\prime} = \sigma^*$, indicating that

function F is chosen from $\mathbf{F}$; Otherwise, he outputs 0.

Note that if F is chosen from $\mathbf{F}$, this is actually game 1; If F is a truly random function, this is game 2. Therefore, the difference between these two games is whether function F is chosen from $\mathbf{F}$. Thus we can obtain equation (3.3). In game 2, the signature between pk_s^* and pk_v^* is a truly random string. After querying q_{sign}, q_{sim} and q_{ver} queries, the probability that $\mathcal{A}$ outputs a valid forgery is up to

$$\begin{aligned} \Pr[X_2] &\leq (2^l - q_{sign} - q_{sim} - q_{ver})^{-1} \\ &< (q_{sign} + q_{sim} + q_{ver})2^{-l}, \end{aligned} \tag{3.4}$$

which is negligible. Combing equations (3.2) to (3.4), we have,

$$\begin{aligned} &\Pr_{\mathcal{A},D}^{SDVS}[Forge(multi\text{-}user)](resp.\ multi\text{-}user^+) \\ &= \Pr[X_0] \leq \sum_{i=1}^{2} |\Pr[X_i] - \Pr[X_{i-1}]| + \Pr[X_2] \\ &< mn \cdot \epsilon_{cpa}(resp.\ \epsilon_{cca}) + \epsilon_{prf} + (q_{sign} + q_{sim} + q_{ver})2^{-l}. \end{aligned} \tag{3.5}$$

We can see from equation (3.5) that the probability of breaking the unforgeability in multi-user setting (resp. multi-user$^+$ setting) is negligible, which completes our proof.

□

Proof. (of Lemma 3.4.2) To simulate the signer's signature on message m, the designated verifier does the following, namely, $K.Encap^2(pk_v, pk_s, sk_v) \rightarrow K_{sv}$, $PRF_{K_{sv}}(m) \rightarrow \sigma$.

The verifier can simulate the signature by running the second phase in *Encap* algorithm with inputting pk_v, pk_s and sk_v. The key K_{sv} that he can obtain is the same as the key that the signer uses to generate signatures. Since both the signer and the verifier can compute the same key, they can generate the same signature on message m, i.e. $tag = PRF_{K_{sv}}(m)$. Therefore, our constructed D scheme is perfectly non-transferability.

□

Proof. (of Lemma 3.4.3) For any PPT distinguisher $\mathcal{A}$ in SDVS's PSI game, $\Pr[PSI]$ in the multi-user (resp. multi-user$^+$) setting is negligibly close to $1/2$ assuming *KEM* scheme is IND-CPA (resp. IND-CCA) secure and *PRF* achieves property of pseudo-randomness.

Let $\mathcal{A}$ be the distinguisher and $\mathcal{C}$ be the challenger against privacy of signer's identity game. Let K_0 denote the shared key between signer $pk^*_{s_0}$ and verifier pk^*_v, K_1 denote shared key between $pk^*_{s_0}$ and verifier pk^*_v. We consider the following games played between $\mathcal{A}$ and $\mathcal{C}$. Let X_i denote the event that $\mathcal{A}$ outputs the correct guess bit in game G_i.

G_0: This game is the PSI game with multi-user (resp. multi-user$^+$) setting. We can have,

$$\Pr^{SDVS}_{\mathcal{A},D}[PSI(multi\text{-}user)](resp.\ multi\text{-}user^+) = \Pr[X_0]. \tag{3.6}$$

G_1: In this game, the key shared between $pk^*_{s_0}$ and pk^*_v used

in *PRF* is randomly chosen, i.e. $K_0' \xleftarrow{\$} N$. When $\mathcal{A}$ issues signing and verification queries on these identities, $\mathcal{C}$ uses K_0' to response. The only difference between this game and game 0 is the key used in *PRF* when responding oracles between S_0 and V. Thus we can have,

$$|\Pr[X_1] - \Pr[X_0]| \leq mn \cdot \epsilon_{cpa}(resp.\ \epsilon_{cca}). \tag{3.7}$$

To obtain the above equation, we can construct an adversary $\mathcal{A}_1$ to break the IND-CPA (resp. IND-CCA)'s game and the analysis is identical to the game 1 of unforgeability and we omit the details here.

$\mathbf{G_2}$: In this game, we replace key K_1 between $pk_{s_1}^*$ and pk_v^* used in *PRF* to a random string, i.e. $K_1' \xleftarrow{\$} N$. Similar to game 1, we can have that,

$$\begin{aligned} |\Pr[X_2] - \Pr[X_1]| &\leq (m-1)n \cdot \epsilon_{cpa}(resp.\ \epsilon_{cca}) \\ &< mn \cdot \epsilon_{cpa}(resp.\ \epsilon_{cca}). \end{aligned} \tag{3.8}$$

This is the same as the transition from $\mathbf{G}_0$ to $\mathbf{G}_1$.

$\mathbf{G_3}$: In game 3, we replace PRF_{K_0} function used between $pk_{s_0}^*$ and pk_v^* to a truly random function . For every signing query on message m_i with $pk_{s_0}^*$, the signer's signature is chosen at random from $(0,1)^l$ instead of computing $PRF_{K_0}(m_i)$. We can have the following equation,

$$|\Pr[X_3] - \Pr[X_2]| \leq \epsilon_{prf}. \tag{3.9}$$

To prove equation (3.9), we can construct an adversary $\mathcal{A}_2$ to break the pseudorandomness of *PRF* with (t_2, ϵ_{prf}), where $t_2 \approx t$. The analysis is identical to the game 2 in lemma 3.4.1.

G_4: In this game, we replace function PRF_{K_1} with a truly random function used between signer $pk^*_{s_1}$ and verifier pk^*_v. Similarly, we can have that,

$$|\Pr[X_4] - \Pr[X_3]| \leq \epsilon_{prf}. \tag{3.10}$$

To obtain the above equation, we can use the same proof strategy as in the transition between **G_2** and **G_3**. Note that signature σ^* that distinguisher $\mathcal{A}$ receives in this game is generated by truly random functions, therefore, he can only randomly guess bit b with $\frac{1}{2}$ probability. Hence, we have the following equation,

$$\Pr[X_4] = \frac{1}{2}. \tag{3.11}$$

Combining equations from (3.6) to (3.11), we obtain that,

$$\begin{aligned} &\Pr_{\mathcal{A},D}^{SDVS}[PSI(multi\text{-}user)](resp.multi\text{-}user^{+}) \\ &= \Pr[X_0] \leq \sum_{i=1}^{4} |\Pr[X_i] - \Pr[X_{i-1}]| + \Pr[X_4] \\ &< 2mn \cdot \epsilon_{cpa}(resp.\ \epsilon_{cca}) + 2\epsilon_{prf} + \frac{1}{2}. \end{aligned} \tag{3.12}$$

Because ϵ_{cpa} (resp. ϵ_{cca}) and ϵ_{prf} are all negligible. It's easy to see that the probability of breaking PSI game in multi-user setting (resp. *multi-user*$^+$ setting) is negligibly close to $\frac{1}{2}$, which completes our proof. □

3.5 Instantiation and Comparison

In this section, we give two instantiations called $SDVS_1$ and $SDVS_2$, which base on DDH assumption. Besides, we give another two post-quantum safe instantiations, namely, $SDVS_3$ and $SDVS_4$, based on LWE assumption.

We employ the well-known Diffie-Hellman key exchange scheme and *PRF* function [NR04] to instantiate our first SDVS scheme ($SDVS_1$). Note that this key exchange scheme satisfies our 2-phase *KEM* requirement. Following our construction, the resulting $SDVS_1$ is the same as the first scheme proposed in [HSW09]. Based on previous analysis, the scheme in [HSW09] is actually secure in multi-user setting. However, since Diffie-Hellman key exchange is not known to be CCA-secure, this scheme is not secure in multi-user$^+$ setting. As for the instantiation in multi-user$^+$ setting, we use a CCA-secure *KEM* scheme proposed in [BSLZ09] and a *PRF* function [NR04] to construct $SDVS_2$, based on DDH assumption. Specifically, *KeyGen* outputs (g^x, x) as receiver's keys; $Encap^1$ outputs $C : g^w$ on input w; $Encap^2$ outputs pk^w given a receiver's public key; *Decap* outputs C^x using receiver's secret key x. Besides, The detailed construction of DDH-based *PRF* can be found in [NR04], we omit its details here.

Following our generic construction, the resulting SDVS is given below for clarify. The keys are generated as, $pk_s = g^{x_s}$,

$sk_s = x_s$; $pk_v = g^{x_v}$, $sk_v = x_v$. Signer first generates signing key as, namely, $K_{sv} = (pk_v)^{x_s}$ and uses this key to sign on message m: $\sigma \leftarrow PRF_{K_{sv}}(m)$. When the verifier obtains signature σ, he first generates the verification key, namely, $K_{vs} = (pk_s)^{x_v}$ and uses this key to compute $\sigma' \leftarrow PRF_{K_{vs}}(m)$. He then outputs $\sigma' \stackrel{?}{=} \sigma$ to indicate validity or invalidity of the signature. This resulting scheme is actually the same as the first scheme presented in [HSW09]. Following our security analysis, the scheme in [HSW09] is actually secure in multi-user setting. However, since Diffie-Hellman key exchange is not known to be CCA-secure, this scheme is not secure in multi-user$^+$ setting.

To instantiate our two SDVS models with multi-uer (resp. multi-user$^+$) setting, we use the scheme called SDVS_1 in [HSW09] as our initiation in the multi-user setting and we also call it as SDVS_1. Brifly speaking, the public key and secret keys in this scheme are set as: $pk = g^x$, $sk = x$ for both signer and verifier, and the signing or verification keys are set as: $K_{sv} = g^{x_s x_v}$ where the underlying hardness assumption is DDH problem. When we set the scheme like this, it is actually the scheme proposed in [HSW09] and we just skip the details here.

Note that this scheme also satisfies our 2-phase *KEM* requirement. The construction of DDH-based *PRF* can also be found in [NR04]. The resulting SDVS scheme following our generic construction is denoted by SDVS_2. It is explicitly stated as follows. We pick a group G of prime order q, with g_1 and g_2 being its generators. Choose a target-collision resistant hash function

$H : \{0,1\}^* \rightarrow Z_q^*$, a key derivation function *KDF*, a pseudo-random function *PRF* and randomly pick $(x_1, x_2, y_1, y_2) \in Z_q^4$. Then we have, $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$. Set public key and secret keys for the signer as, $pk_s = (g_1, g_2, c, d)$, $sk_s = (x_1, x_2, y_1, y_2)$. Randomly pick $r \in Z_q^*$ and compute, $u_1 = g_1^r$, $u_2 = g_2^r$, $\alpha = H(u_1, u_2)$ and $v = c^r d^{r\alpha}$. Verifier's public and secret keys are set as, $pk_v = (u_1, u_2, v)$, $sk_v = r$. When the signer generates signatures, he first generates signing key K_{sv} by running *Decap* algorithm and uses *PRF* to compute signatures on message m as follows, $\alpha = H(u_1, u_2), v' = u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha}, K_{sv} = KDF(u_1, u_1^{x_1} u_2^{x_2}), \sigma = PRF_{k_{sv}}(m)$. If $v' = v$ then returns σ; otherwise, returns $\perp$. When the verifier obtains message-signature pair (m, σ), he computes, $K_{sv} = KDF(u_1, c^r)$ and $\sigma' = PRF_{K_{sv}}(m)$, where $u_1 = pk_v$, $r = sk_v$. Verifier will then return $\sigma' \stackrel{?}{=} \sigma$ to indicate signature's validity or invalidity.

As for the lattice-based versions, we construct $SDVS_3$ scheme, based on a *KEM* [BCD+16] and a *PRF* function [BPR12]. The constructed $SDVS_4$ scheme derives from [ZYFZ19] and [BPR12], based on LWE assumption. We omit details of these constructions here due to page limitation.

We construct $SDVS_3$ based on CPA secure scheme proposed in [BCD+16] where parameter is set as FrodoKEM-976. The corresponding construction of LWE-based *PRF* function can be found in [BPR12]. Let $n, \bar{m}, \bar{n}$ denote the integer matrix dimensions with $n \equiv 0 \pmod 8$. The scheme is stated as follows. We choose a hash function H and a uniformly random seed

$seed_A \xleftarrow{\$} U\{0,1\}^s$ with using it to generate a matrix $\mathbf{A} \in Z_q^{n\times n}$ as, $\mathbf{A} \leftarrow Frodo.Gen(seed_A)$. Let χ denote a distribution over a set S, generate two $n \times \bar{n}$ matrices by sampling each of its entries independently from S according to χ: $\mathbf{S}, \mathbf{E} \leftarrow \chi(Z_q^{n\times\bar{n}})$, and generate matrix $\mathbf{B} \in Z_q^{n\times\bar{n}}$ as, $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$. Signer's keys are set as, $pk_s \leftarrow (seed_A, \mathbf{B})$, $sk_s \leftarrow \mathbf{S}$. Use the same way to generate the following four matrices, namely, $\mathbf{A} \leftarrow Frodo.Gen(seed_A)$, $\mathbf{S}', \mathbf{E}' \xleftarrow{\$} \chi(Z_q^{\bar{m}\times n})$, $\mathbf{E}'' \leftarrow \chi(Z_q^{\bar{m}\times\bar{n}})$, and compute $\mathbf{B}' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$, $\mathbf{V} \leftarrow \mathbf{S}'\mathbf{B} + \mathbf{E}''$, $\mathbf{C} \leftarrow (\mathbf{C}_1, \mathbf{C}_2) = (\mathbf{B}', \mathbf{V} + Frodo.Encode(\mu))$, where μ is randomly chosen. The verifier's keys are set as, $pk_v \leftarrow \mathbf{C}$, $sk_v \leftarrow (\mathbf{S}', \mathbf{E}', \mathbf{E}'')$. When the signer wants to generate signatures, he first computes $\mathbf{M} = \mathbf{C}_2 - \mathbf{C}_1\mathbf{S}$, $\mu \leftarrow Frodo.Decode(\mathbf{M})$, and uses $H(\mu)$ to sign on message m, i.e. $\sigma \leftarrow PRF_{H(\mu)}(m)$.

The constructed $SDVS_4$ scheme derives from [ZYFZ19], based on LWE assumption. The construction of underlying LWE-based *PRF* can be found in [BPR12]. It's straightforward to see that CCA-secure PKE scheme in [ZYFZ19] can be easily transformed into a *KEM* to construct SDVS.

In Table 3.2, we compare our four instantiations in multi-user (resp. multi-user$^+$) setting with the existing SDVS schemes. We consider pairing, hash, *PRF* and exponentiation operations, denoted by **P**, **H**, **R** and **E** respectively. Please be noted that the figures of our constructed $SDVS_2$ come from [BSLZ09] whose conclusion relies on the multi-exponential with a sliding window algorithm described in [M.05].

Table 3.2: Comparison Between Our Instantiations and Existing SDVS schemes

	$SDVS_1$	$SDVS_2$	[HSMZ08]	[TOO05]	[LV05b]		$SDVS_3$	$SDVS_4$	[NJ16]
Signing Cost	1**E**+1**R**	2.78**E**+1**R**+1**H**	1**E**+1**H**	2**E**+1**H**	2**H**+1**P**	PK Size (MB)	2.99×10^{-2}	21.82	1.34
Verification Cost	1**E**+1**R**	1**E**+1**R**	1**E**+1**H**	2**E**+1**H**	2**H**+1**P**	SK Size (MB)	1.49×10^{-2}	8.44	49.59
Hardness Assumption	DDH	DDH	GDH	CDH+DDH	GBDH	Hardness Assumption	LWE	LWE	LWE
Standard Model	$\surd$	$\surd$	$\times$	$\times$	$\times$	Standard Model	$\surd$	$\surd$	$\surd$
Multi-user$^+$	$\times$	$\surd$	$\times$	$\times$	$\times$	Multi-user$^+$	$\times$	$\surd$	$\times$

We can see from Table 3.2 that our schemes, $SDVS_1$ and $SDVS_2$, are secure in multi-user and multi-user$^+$ setting respectively in the standard model. As for $SDVS_3$ and $SDVS_4$, they are quite efficient compared with the lattice-based SDVS scheme under the same security requirement.

Chapter 4

Zero-knowledge Arguments for Paillier

Considering the widespread use of the Paillier cryptosystem in various applications, it is crucial to develop zero-knowledge proof (ZKP) systems for Paillier, thereby ensuring active security for these applications. Generally, there are two approaches for constructing ZKP systems for Paillier. The first one involves customizing proof systems for Paillier using algebraic methods, as exemplified by [Lin17, LN18, DJ01a]. However, existing works constructed in this manner exhibit linear proof size and verification time concerning the number of problem instances to be proved, rendering them less practical for real-world applications. For instance, in the voter analysis scenario introduced in Chapter 1, there are typically thousands of records in one polling station, thus the resulting proof size and verification cost are quite large. Although some existing schemes may offer efficient verification time, using existing tools to prove thousands of records could result in an inefficient proof cost.

Another potential approach entails using existing sub-linear

argument systems for generic statements (e.g., zk-SNARK). However, this leads to prohibitive proof generation costs, as it requires translating the relation to be proven into an excessively large Boolean or arithmetic circuit over a prime order field. Specifically, the source of inefficiency in using existing zk-SNARKs stems from representing statements related to Paillier encryption using an arithmetic circuit over a prime field.

In this thesis, we investigate a different approach - representing the statement being proven using an arithmetic circuit over the ring of residue classes modulo a composite number ($\mathbb{Z}_{N^2}$), which matches the ciphertext space of the Paillier cryptosystem. Modular arithmetic can then be represented using a simple gate which greatly simplifies the representation of Paillier encryption. Subsequently, we investigate how to adapt existing ZKPs for arithmetic circuits over a prime field into our setting.

The primary obstacle when working in $\mathbb{Z}_{N^2}$ instead of a prime field is that it is unclear how one can prove a message is binary. In a prime field, $b * (b - 1) = 0$ implies b is binary. Yet, in our setting, there are non-trivial roots because N^2 is not a prime number. To solve this problem, we develop an innovative approach: the prover additionally provides the sum of a random subset (of the verifier's choice) of the witness "bits". If the sum is small, the verifier is convinced that all witness "bits" are binary[1]. To offer zero-knowledge, the sum is not provided in the

[1]To sustain the claim, one needs to show that all non-trivial roots are large.

clear but is masked by some small "noise" value.

Chapter Organization. We give a summary of our contribution in Section 4.1. We then give an overview of our approach in Section 4.2. Our main protocol is given in Section 4.3 where we can use our method to prove that multiple bits "packed" in specific positions of Paillier messages, along with other requirements of these bits. We then extend this protocol into a range proof protocol in Section 4.4. The underlying security proof for our main protocol is given in Section 4.5. Besides, we give the experimental results in Section 4.6 to examine its practicability.

4.1 Our Contribution

We present an efficient zero-knowledge argument of knowledge system customized for Paillier cryptosystem. Our system enjoys sub-linear proof size, low verification cost, and acceptable proof generation effort, while also supporting batch proof generation/verification. More specifically, we propose several zero-knowledge argument of knowledge for various relations for Paillier cryptosystem, including (1) the well-formedness of multiple Paillier ciphertexts with packing of binary messages; (2) an extension that proves additionally the number of one's in each ciphertext is no larger than a certain threshold and the number of one's in each unit exceeds a certain threshold; and

(3) range proof of multiple Paillier ciphertexts. Our proof system features sub-linear proof size and efficient verification time, while maintain an acceptable proof generation time. Specifically, we made the following contributions.

- We design a constraint system defined over $\mathbb{Z}_{N^2}$ to represent correct encryption of Paillier cryptosystem with plaintext satisfying various properties. We show how to compile the constraint system into an zero-knowledge argument of knowledge (ZKAoK) which allows a prover to convince a verifier the knowledge of witnesses satisfying the constraint system without revealing extra information.

- We design new techniques to prove that a witness is binary even if the constraint system is defined over $\mathbb{Z}_{N^2}$ for an RSA modulus N. We believe that our new techniques can be used for other scenarios where using arithmetic constraints over composite order filed is desirable.

- Based on the above, we give efficient ZKAoKs useful for data analytic applications. We conduct a series of experiments to examine their practicability. For proving packed Paillier with binary messages, our proof size is 27x smaller than using a standard OR-proof in proving 51.2K bits when $|N| = 2048$. For proving 800 messages are all 256-bit numbers, our proof size is 27x smaller compare with state-of-the-art range proof. Since our system is asymptotically more efficient the gap is even larger for more ciphertexts.

Table 4.1: Summarization and Comparison of Our Protocol with the State-of-the-art Approaches in Different Scenarios with $|N| = 2048$ bits.

Our protocol can support batch proof and the inputs of ZKAoK* *are packed Paillier messages.* $\mathcal{N}_p$ *and* $\mathcal{N}_b$ *denote the number of proved plaintexts and bits in our protocol. "−" indicates that this attribute does not apply in this protocol. In a proving many bits scenario, the averaged cost measures the cost for a single bits while that in a range proof measures the cost for one message.*

Scenario	Protocol	$\mathcal{N}_p$	$\mathcal{N}_b$	Ave. Proof Cost	Ave. Proof Time	Ave. Verification Time
multiple binary records	OR-proof	1	1	>16384 bits	35.32 ms	18.04 ms
	ZKAoK*	800	51.2K	**605.92 bits**	2.53 s	**4.19 ms**
		1M	64M	**18.85 bits**	2.19 s	**2.02 ms**
		10M	640M	**7.90 bits**	2.19 s	**1.98 ms**
multiple range proofs	[Lin17, Bou00]	1	-	> 128 KB	241.43 ms	199.32 ms
	ZKAoK$'$	800	-	**4.68 KB**	160.21 s	230.70 ms
		1M	-	**932.84 bits**	139.43 s	**97.68 ms**
		10M	-	**232.12 bits**	139.42 s	**95.32 ms**

Let us revisit our voter analysis example introduced in Chapter 1. Table 4.1 compares the performance of our system and existing systems for the aforementioned voter analysis scenario, where there are $\mathcal{N}_p$ voters and $\mathcal{N}_b/\mathcal{N}_p$ seats to be filled. We also compare our system with existing systems for multiple range proofs when there are $\mathcal{N}_p$ ciphertexts (to prove that all $\mathcal{N}_p$ ciphertexts correspond to 256-bit plaintexts). One can see our system outperforms existing works in terms of proof size and verification time when the number of ciphertexts is in the order of hundreds.

4.2 Technical Overview of Our Results

We give a technical overview of our solution (main protocol) that proves correctness of packed Paillier encryption of multiple binary messages. We further show how we can prove correctness of polynomially many such ciphertexts.

Representing Arithmetic Circuit. Following the terminology of [BCC+16], the statements to be proved are represented as a list of equations known as constraints. For example, the following list of 5 constraints represents the circuit shown in Fig. 4.1. Note that all constraints are modulo N^2 unless otherwise indicated.

$$a_1 * b_1 = c_1$$
$$a_2 * b_2 = c_2$$
$$a_3 * b_3 = c_3$$
$$3 * c_1 = a_3$$
$$c_1 + c_2 = b_3$$

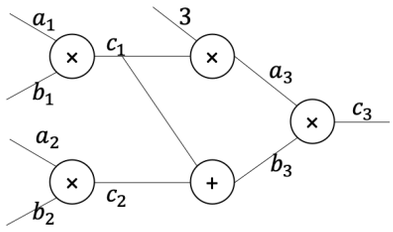

Figure 4.1: An Arithmetic Circuit

The knowledge of wire assignment to satisfy the circuit translates directly to assignment of variables satisfying the set of constraints. There are two types of constraints, namely, multiplication and linear constraints. A multiplication constraint is of the form $a_x * b_x = c_x$ while a linear constraint is of the form $\sum_x w_x^{(a)} a_x + \sum_x w_x^{(b)} b_x = \sum_x w_x^{(c)} c_x + c_0$, where constants $\{w_x^{(a)}, w_x^{(b)}, w_x^{(c)}, c_0\}$ depend solely on the circuit and the public values while $\{a_x, b_x, c_x\}$ are the assignments of the wires (depends on values known only to the prover, or say witness).

Note also that only wires of multiplication gates of intermediate values are labelled. Wires of addition gates and multiplication gates with public inputs (constants) are handled by linear constraints. To prove that the prover knows some input such that the output of above circuit is some specific number, say 0, one could add another linear constraint as $c_3 = 0$.

Jonathan et al. [BCC+16] showed how to transform a circuit into constraints, and their zero-knowledge argument system works directly over a set of constraints.

Constraints for Correctness of Paillier Encryption. Our first contribution is a (compact) set of constraints handcrafted to represent correct encryption. The prover wants to prove that he knows a pair, (m, r), satisfying $c = (1+N)^m \cdot r^N \mod N^2$, which can be computed as $c = (1+mN) \cdot r^N \mod N^2$, where N is an RSA modulus. Note that $(1+mN)$ is readily a linear constraint. We focus on our set of constraints for r^N.

Let $\alpha = \lceil \log N \rceil$ and $\{n_\alpha, \ldots, n_2, n_1\}$ be the binary decomposition of N. That is, $N = 2^{\alpha-1} \cdot n_\alpha + \cdots + 2 \cdot n_2 + n_1$. Define sequence $\tilde{R} = \{\tilde{R}_1 = r, \tilde{R}_2 = r^2, \ldots, \tilde{R}_\alpha = r^{2^{\alpha-1}}\}$. Let β be the hamming weight of N. Define index set $\tilde{D} = \{\gamma | n_\gamma = 1\}$. We have $|\tilde{D}| = \beta$. We use $d_1, \ldots, d_\beta$ to denote elements of $\tilde{D}$ with $d_i < d_j$ if $i < j$.

Define sequence $\tilde{S} = \{\tilde{S}_1, \ldots, \tilde{S}_\beta\}$ such that $\tilde{S}_1 = \tilde{R}_{d_1}$, $\tilde{S}_k = \tilde{S}_{k-1} \cdot \tilde{R}_{d_k}$ for $k \in [2, \beta]$. We have $\tilde{S}_\beta = r^N$ if $\tilde{R}_1 = r$. Note that $\tilde{R}$, $\tilde{S}$ are the intermediate values when we calculate r^N from r

using the square-and-multiply algorithm. One can uniquely compute $(\alpha, \beta, \tilde{D})$ from N.

As an example, consider N = 11, which can be represented as 1011 in binary. Then $\alpha = 4$ and $\beta = 3$. To compute r^{11}, sequence $\tilde{R}$ is set as, $\tilde{R} = \{\tilde{R}_1 = r, \tilde{R}_2 = r^2, \tilde{R}_3 = r^4, \tilde{R}_4 = r^8\}$. Sequence $\tilde{D}$ is, $\tilde{D} = \{d_1 = 1, d_2 = 2, d_3 = 4\}$. We have $\tilde{S} = \{\tilde{S}_1 = \tilde{R}_1 = r, \tilde{S}_2 = \tilde{S}_1 \cdot \tilde{R}_2 = r^3, \tilde{S}_3 = \tilde{S}_2 \cdot \tilde{R}_4 = r^{11}\}$.

Correctness of sequence $\tilde{R}$ and $\tilde{S}$ indicates r^N is computed correctly. Thus, correct encryption of Paillier ciphertext can be represented using the following constraints.

$$
\begin{aligned}
& c = \grave{T} * \tilde{S}_\beta \\
& \grave{T} = 1 + m * N \\
& \tilde{R}_1 = r \quad //\text{for clarity} \\
& \tilde{R}_{i+1} = \tilde{R}_i * \tilde{R}_i && i \in [1, \alpha - 1] \\
& \tilde{S}_1 = \tilde{R}_{d_1} \quad //\text{for clarity} \\
& \tilde{S}_k = \tilde{S}_{k-1} * \tilde{R}_{d_k} && k \in [2, \beta]
\end{aligned}
\tag{4.1}
$$

For ease of writing, we define the above set of constraints as $\mathsf{Const}_{\{c,m,r\}}$, with respect to c, m, and r. Note that $\tilde{R}_1$, r (resp. $\tilde{S}_1$, $\tilde{R}_{d_1}$) can be combined into one witness. Thus, $\mathsf{Const}_{\{c,m,r\}}$ contains $\alpha + \beta - 1$ multiplication constraints and one linear constraint.

Proof that A Message Is Binary. Very often, we need to prove that a variable in the constraint is binary. For example, we may need

to prove that c is an encryption of a message m. If the constraints are defined over a prime field, adding the following constraint is sufficient:

$$m * (m - 1) = 0.$$

However, we work in $\mathbb{Z}_{N^2}$ and the above constraint does not guarantee m is binary. According to the Chinese Reminder Theorem, there are 4 values satisfying this constraint:

$$\begin{cases} m = & 0 \\ m = & 1 \\ m = & q^2 \cdot [(q^2)^{-1} \mod p^2] \ (= X) \\ m = & p^2 \cdot [(p^2)^{-1} \mod q^2] \ (= Y), \end{cases}$$

One of the core technical contributions of this work is an innovative statistical argument to ensure m is binary. In more detail, our solution requires that the prover also commits a random "noise" value,R', chosen from a relatively small range $\mathcal{L}$[2], say, $\mathcal{L} := \{1, \ldots, 2^{256}\}$. The verifier will choose a random challenge, $\ell \in \{0, 1\}$, and the prover is required to give $L' := \ell m + R'$, along with a proof that L' is computed correctly. For simplicity, we will also use Paillier encryption for the 'commitment' of R'. And the proof that L' is correctly computed can be represented by a linear constraint.

More concretely, the prover computes and sends $c' = (1 +$

[2]Here, we also require the prover to attach a range proof for R'.

$N)^{R'} * r'^N \mod N^2$ to the verifier, who replies with a challenge bit ℓ, and the prover sends L' along with a proof that the following constraints are satisfied:

$$\begin{aligned}
&\mathsf{Const}_{\{c,m,r\}}\\
&m * (m-1) = 0\\
&\ell * m + R' = L'\\
&\mathsf{Const}_{\{c',R',r'\}}.
\end{aligned}$$

Besides checking the proof, the verifier also checks whether L' is in $\mathcal{L}$. For an honest prover, standard statistical argument ensures L' leaks negligible information about m since R' is much larger than m. A cheating prover may use X or Y as a witness. Recall that X and Y are large (of the order p^2 or q^2), the only way for a cheating prover to ensure $L' := \ell * m + R' \in \mathcal{L}$ is to guess ℓ and pick R' accordingly. If he/she guesses that $\ell = 1$, he/she should pick a large R' such that X (or Y) plus R' modulo N^2 is within $\mathcal{L}$. Likewise, if he/she guesses $\ell = 0$, he/she should pick a small R', i.e., $R' \in \mathcal{L}$. Therefore, with probability $1/2$, a cheating prover will be caught. To amplify soundness, above is repeated κ times (say $\kappa = 128$).

Proof that Polynomially-many Messages are Binary. Our method can be extended to prove that polynomially many witnesses are binary. Specifically, assume we would like to prove that there are $\mathcal{N}_p$ ciphertexts, each of which encrypts a binary message.

We use (m_i, c_i) to denote one message-ciphertext pair, where $i \in [1, \mathcal{N}_p]$. Same as before, the auxiliary information generated by the prover is $\{R'_j\}_{j\in[1,\kappa]}$, encrypted in $\{c'_j\}_{j\in[1,\kappa]}$ using randomness $\{r'_j\}_{j\in[1,\kappa]}$. Now, the random challenges from the verifier are $\{\ell_j^{(i)}\}_{i\in[1,\mathcal{N}_p],j\in[1,\kappa]}$. The corresponding constraints are:

$$\begin{aligned}
&\mathsf{Const}_{\{c_i,m_i,r_i\}},\ i \in [1, \mathcal{N}_p] \\
&m_i * (m_i - 1) = 0,\ i \in [1, \mathcal{N}_p] \\
&\sum_{i=1}^{\mathcal{N}_p} \ell_j^{(i)} * m_i + R'_j = L'_j,\ j \in [1, \kappa] \\
&\mathsf{Const}_{\{c'_j,R'_j,r'_j\}},\ j \in [1, \kappa].
\end{aligned}$$

Same as above, the verifier checks that L'_j is in $\mathcal{L}$ for $j \in [1, \kappa]$ in addition to checking the proof. We would like to remark that the amortized cost for proving one message, say m, being binary is 1 (i.e., the constraint of $m * (m - 1) = 0$). Intuitively, if any of the m_i is malformed (say, $m_i = X$ or $m_i = Y$), probability that all $L'_j \in \mathcal{L}$ is $2^{-\kappa}$, which is negligible when we set κ to 128.

The actual analysis is much more involved since we need to show no matter how a cheating prover chooses his m_i's, the probability that it can pass the verification is bounded (in fact, we show that it is at most $1/2$) if any of the m_i is X or Y (and is independent of the number of messages). The analysis is shown in Lemma 4.5.2.

Proof of Messages with Correct Structure. Recall our goal is to prove the correctness of encryption for messages with specific formats such as packing. For example, we may consider packing two binary messages into one ciphertext, where the first two slots are 32-bit. That is,

$$m = \underbrace{00\ldots0}_{32-bit}\ldots\underbrace{00\ldots b_{32}}_{32-bit}\underbrace{00\ldots b_0}_{32-bit}.$$

We can make use of the constraint $2^{32} \cdot b_2 + b_1 = m$ to shift the bits to the correct position. For instance, the following set of constraints represents all i ciphertexts are encryption of 2 bits, each occupying a 32-bit slot:

$$\begin{aligned}
&\mathsf{Const}_{\{c_i, m_i, r_i\}} \\
&2^{32} \cdot b_{32}^{(i)} + b_0^{(i)} = m_i \\
&b_0^{(i)} * (b_0^{(i)} - 1) = 0 \\
&b_{32}^{(i)} * (b_{32}^{(i)} - 1) = 0 \\
&\mathsf{Const}_{\{c'_j, R'_j, r'_j\}} \\
&\sum \left(l_{j,0}^{(i)} b_0^{(i)} + l_{j,32}^{(i)} b_{32}^{(i)} \right) + R'_j = L'_j,
\end{aligned}$$

for $i \in [1, \mathcal{N}_p]$ and $j \in [1, \kappa]$.

One may wish to directly extend the above method to support Paillier with packing for an arbitrary number of slots and plaintext, e.g.,

$$m = \underbrace{0\ldots0b_{32\cdot63}}_{32-bit}\underbrace{0\ldots0b_{32}}_{32-bit}\ldots\underbrace{0\ldots0b_0}_{32-bit},$$

where $|N| = 2048$, and we pack 64 bits into 64 slots. However, our analysis showed that this is not straightforward. The reason is that we have to ensure 'bits' $\{b^{(i)}_{32\cdot 63}, \ldots, b^{(i)}_0\}$ and auxiliary input $\{R'_j\}$ are fixed before random challenges $\{l^{(i)}_{j,32(s-1)}\}$ are chosen. However, given (1) $c_i = (1 + m_i N) r_i^N \bmod N^2$; (2) $m = \sum_s 2^{32(s-1)} b^{(i)}_{32(s-1)} \bmod N^2$; and (3) $b^{(i)}_{32(s-1)} * (b^{(i)}_{32(s-1)} - 1) = 0 \bmod N^2$, the set $\{b^{(i)}_{32(s-1)}\}$ is not unique (despite m is fixed due to the injective nature of encryption).

This counter-intuitive observation arises from the fact that at this point we cannot ensure $\{b^{(i)}_{32(s-1)}\}$ are binary and thus $m = \sum_s 2^{32(s-1)} b^{(i)}_{32(s-1)} \bmod N^2$ may have multiple solutions. There is a possibility that a malicious prover may choose different $\{b^{(i)}_{32(s-1)}\}$ after seeing challenges, and the analysis in Lemma 4.5.2 crucially relies on the fact that the prover's "bits" are fixed before seeing the verifier's challenges.

We tackle this subtlety by carefully identifying the condition under which the prover's "bits" are fixed. Specifically, we observe that if message m satisfies $|m| < \min\{|p|, |q|\}$ (where p and q are two factorizations of N), fulfilling constraints (1) $m = \sum_s 2^{32(s-1)} b_{32(s-1)}$; and (2) $b_{32(s-1)} * (b_{32(s-1)} - 1) = 0$ (for $s \in [1, 64]$), the set $\{b_{32(s-1)}\}$ is unique. The formal analysis is shown in Lemma 4.5.3.

Since under this condition the set $\{b_{32(s-1)}\}$ satisfying the constraints is unique, the 'bits' are fixed given c and above constraints. Consequently, when the message space is 2048-bit, it is only safe to use 1024 bits. In other words, we can only use 32

32-bit slots to achieve provable security. This is not ideal and we describe our final solution below.

Our Final Solution. We construct auxiliary messages to fulfill the above "length requirement". Assuming $|N| = 2048$ and we divide the message space into 64 slots (each of which is 32-bit), we need to introduce one new auxiliary message m_t^* for every 15 messages. Thus there will be $\mathcal{N}_p/15$ auxiliary messages in total. Here we give an example to see how we construct m_t^* from messages m_{15t-14} to m_{15t} ($t \in [1, \mathcal{N}_p/15]$),

$$m_t^* = \underbrace{b_{32\cdot 63}^{(15t)} \ldots b_0^{(15t)}}_{from\ m_{15t}} \ \ldots \ \underbrace{b_{32\cdot 63}^{(15t-14)} \ldots b_0^{(15t-14)}}_{from\ m_{15t-14}},$$

where $b_{32(s-1)}^{(i)}$ indicates the last bit in the s-th slot of message m_i, for $s \in [1, 64]$ and $i \in [1, \mathcal{N}_p]$. Each auxiliary message is 960-bit, and they satisfy,

$$\begin{aligned}&(2^{959} * b_{32\cdot 63}^{(15t)} + \ldots + 2^{896} * b_0^{(15t)}) + \ldots \\ &+ (2^{63} * b_{32\cdot 63}^{(15t-14)} + \ldots + b_0^{(15t-14)}) = m_t^*,\end{aligned}$$

for $t \in [1, \mathcal{N}_p/15]$. We use c_t^* to denote Paillier ciphertexts of m_t^*. As the length of m_t^* satisfies above requirement, we can use our proposed method to prove that all $\{b_{32(s-1)}^{(i)}\}$ are 0 or 1.

Constraints in our final solution are given below:

$$\mathsf{Const}_{\{c_i, m_i, r_i\}}$$
$$\sum_s 2^{32(s-1)} b^{(i)}_{32(s-1)} = m_i$$
$$b^{(i)}_{32(s-1)} * (b^{(i)}_{32(s-1)} - 1) = 0$$
$$\mathsf{Const}_{\{c'_j, R'_j, r'_j\}}$$
$$\sum_{i,s} \ell^{(i)}_{j,32(s-1)} b^{(i)}_{32(s-1)} + R'_j = L'_j$$
$$\sum_s \sum_k 2^{32(k-1)+s-1} \cdot b^{15(t-1)+k}_{32(s-1)} = m^*_t$$
$$\mathsf{Const}_{\{c^*_t, m^*_t, r^*_t\}},$$

where $j \in [1, \kappa]$, $s \in [1, 64]$, $k \in [1, 15]$, $i \in [1, \mathcal{N}_p]$ and $t \in [1, \frac{\mathcal{N}_p}{15}]$. We use $\ell^{(i)}_{j,32(s-1)}$ to indicate the random challenges. Since there are more bits now in the computation of L'_j, range $\mathcal{L}$ will be slightly enlarged, say, $\mathcal{L} := \{0, 2^{281}\}$.

4.3 Our Main Protocol ZKAoK*

In this section, we give the construction of our main protocol ZKAoK* and its extension, ZKAoK$^+$. Since the ZKP system proposed in [BCC$^+$16] requires constraints as inputs, it is sufficient for us to specify constraints for corresponding relations under modulo N^2. t is straightforward to adapt [39] to work over $\mathbb{Z}_{N^2}$ except how the cyclic group is generated with order N^2 can be generated. Here we describe one such method. Given N^2, one

first find prime Q such that $Q = fN^2 + 1$ for some small integer f. Then, choose an arbitrary element g in $\mathbb{Z}_Q^*$ such that $g^{N^2} = 1 \; mod \; Q$. We use g to generate $\mathbb{G}$.

4.3.1 Constraints for A Valid Paillier Message Ciphertext Pair

We recall a building block, $\mathsf{Const}_{\{c,m,r\}}$, that specifies constraints for proving a valid Paillier message-ciphertext pair (m, c) with randomness r. Let $\alpha = \lceil \log N \rceil$ and $S_N = \{n_\alpha, \ldots, n_2, n_1\}$ as the set containing the binary decomposition bits of N, satisfying $N = \sum_{k \in [1,\alpha]} 2^{k-1} \cdot n_k$. Define sequence $\tilde{R} := \{\tilde{R}_k : \forall k \in [1, \alpha], \tilde{R}_k = r^{2^{k-1}}\}$. Let β be the hamming weight of N. Define index sequence as,

$$\tilde{D} := \{d_\gamma | \forall \gamma \in [1, \beta], \; n_{d_\gamma} = 1 \; \wedge \; n_{d_\gamma} \in S_N \\ \wedge \forall \gamma \in [1, \beta - 1], d_\gamma < d_{\gamma+1}\}.$$

We have $|\tilde{D}| = \beta$. Define sequence,

$$\tilde{S} := \{\tilde{S}_k | \forall k \in [1, \beta], \tilde{S}_k = \tilde{S}_{k-1} \cdot \tilde{R}_{d_k} \; \wedge \; \tilde{S}_0 = 1\},$$

where $|\tilde{S}| = \beta$ and $\tilde{S}_\beta = r^N$ (with setting $\tilde{R}_1 = r$).

We denote the constraints for proving that c is a valid Paillier ciphertext of m with r as $\mathsf{Const}_{\{m,c,r\}}$,

$$\begin{cases} c = \grave{T} * \tilde{S}_\beta \\ \grave{T} = 1 + m * N \\ \tilde{R}_1 = r \quad //\text{for clarity} \\ \tilde{R}_{i+1} = \tilde{R}_i * \tilde{R}_i & i \in [1, \alpha - 1] \\ \tilde{S}_1 = \tilde{R}_{d_1} \quad //\text{for clarity} \\ \tilde{S}_k = \tilde{S}_{k-1} * \tilde{R}_{d_k} & k \in [2, \beta] \end{cases} \tag{4.2}$$

Note that $\tilde{R}_1$, r (resp. $\tilde{S}_1$, $\tilde{R}_{d_1}$) can be combined into one witness. Thus, $\mathsf{Const}_{\{c,m,r\}}$ contains $(\alpha + \beta - 1)$ multiplication constraints and one linear constraint.

4.3.2 Our Main Protocol ZKAoK*

Relation $\mathcal{R}^*$ for Main Protocol. Our goal is to prove that given $\mathcal{N}_p$ ciphertexts, $\{c_i\}_{i\in[1,\mathcal{N}_p]}$, where c_i is the encryption of m_i satisfies,

$$m_i = \underbrace{0\ldots0b^{(i)}_{32\cdot(64-1)}}_{32-bit}\underbrace{0\ldots0b^{(i)}_{32\cdot(63-1)}}_{32-bit}\ldots\underbrace{0\ldots0b^{(i)}_{0}}_{32-bit}. \tag{4.3}$$

That is to say, each message contains 64 32-*bit* slots, and all bits except the last bit in each slot are 0. (One can easily prove that each message contains 32 32-*bit* slots when $|N| = 1024$ using the same technique with $s \in [1, 32]$.) Formally, the relation we

prove can be described by $\mathcal{R}^*$ defined below:

$$\begin{aligned}
&\mathcal{R}^* = \{(\{c_i\}_{i\in[1,\mathcal{N}_p]}, N), (m_i, r_i, b^{(i)}_{32(s-1)})_{i\in[1,\mathcal{N}_p], s\in[1,64]} : \\
&\forall\, i \in [1,\mathcal{N}_p] \textit{ and } s \in [1,64], c_i = (1+N)^{m_i}\cdot r_i^N \mod N^2 \\
&\wedge\, m_i = \textstyle\sum_{s\in[1,64]} 2^{32(s-1)}\cdot b^{(i)}_{32(s-1)} \mod N \wedge\ b^{(i)}_{32(s-1)} \in \{0,1\}\}.
\end{aligned} \tag{4.4}$$

Main Protocol ZKAoK*. We define,

$$m_t^* = \underbrace{b^{(15t)}_{32\cdot 31}\ldots b^{(15t)}_0}_{\textit{from } m_{15t}} \cdots \underbrace{b^{(15t-14)}_{32\cdot 31}\ldots b^{(15t-14)}_0}_{\textit{from } m_{15t-14}}.$$

Then m_t^* satisfies,

$$\begin{aligned}
m_t^* &= 2^{959}\cdot b^{(15t)}_{64} + \cdots + b^{(15t-14)}_1 \mod N^2 \\
&= \sum_{s\in[1,64],k\in[1,15]} 2^{64(k-1)+s-1}\cdot b_s^{(15(t-1)+k)} \mod N^2,
\end{aligned}$$

where $t \in [1, \mathcal{N}_p/15]$. We also define c_t^* as their encryption,

$$c_t^* = (1+N)^{m_t^*}\cdot r^N \mod N^2.$$

Our main protocol, ZKAoK*, is shown in Fig. 4.2. We take $\mathcal{L} = [0, 2^{281}]$. In addition, the prover should also prove that R'_j is chosen from $\mathcal{L}$. This can be done by a standard range proof for each R'_j or ZKAoK$'$, which will be discussed in Section 4.4.

4.3.3 An Extended Protocol ZKAoK$^+$

Relation $\mathcal{R}^+$. We now extend our main protocol to ZKAoK$^+$, proving that structured messages satisfy additional requirements (i.e., sum of records and sum of entities). Its formal relation $\mathcal{R}^+$

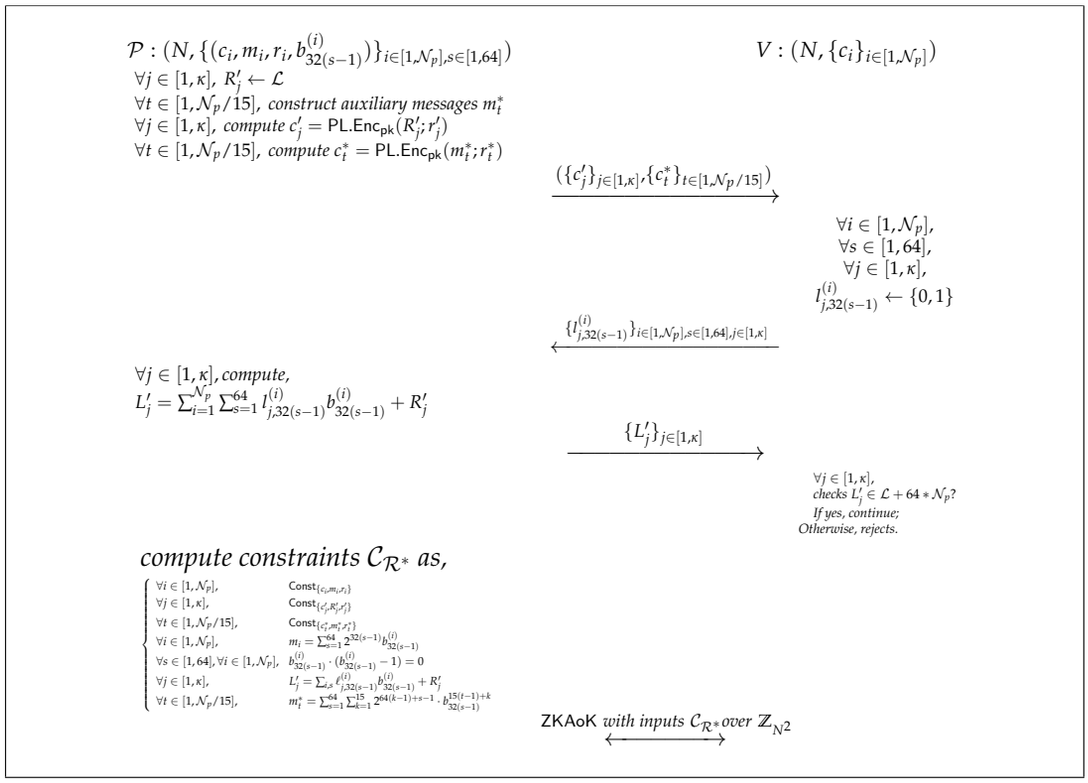

Figure 4.2: Our Main Protocol ZKAoK* for Relation $\mathcal{R}^*$

is,

$$\begin{aligned}
&\mathcal{R}^+ = \{(\{c_i\}_{i\in[1,\mathcal{N}_p]}, N, t, T), (m_i, r_i, b^{(i)}_{32(s-1)})_{i\in[1,\mathcal{N}_p],s\in[1,64]} : \\
&\forall\ i \in [1,\mathcal{N}_p]\ and\ s \in [1,64], c_i = (1+N)^{m_i} \cdot r_i^N \mod N^2 \\
&\wedge\ m_i = \textstyle\sum_{s\in[1,64]} 2^{32(s-1)} \cdot b^{(i)}_{32(s-1)} \mod N \wedge\ b^{(i)}_{32(s-1)} \in \{0,1\} \\
&\wedge\ \textstyle\sum_{s\in[1,64]} b^{(i)}_{32(s-1)} \leq t\ \wedge \sum_{i\in[1,\mathcal{N}_p]} b^{(i)}_{32(s-1)} \geq T\}.
\end{aligned} \tag{4.5}$$

Our Solution. The sum of records property requires that there are at most t 1's packed in $\{b^{(i)}_{32(s-1)}\}_{s\in[1,64]}$ for each message m_i. Suppose $t = 2^x$. we define u_i as the summation of all $\{b^{(i)}_{32(s-1)}\}$'s towards the same entity i. It is sufficient to prove that each u_i can be decomposed by x bits, $\{w^{(i)}_x, \ldots, w^{(i)}_2, w^{(i)}_1\}$, s.t., $u_i = \sum_{s\in[1,64]} b^{(i)}_{32(s-1)} = \sum_{v\in[1,x]} 2^{v-1} w^{(i)}_v$, where $w^{(i)}_v \in \{0,1\}$. We define c_{u_i} as the Paillier encryption of u_i with randomness r_{u_i}.

The sum of units property requires that there are at least T 1's among $\{b^{(i)}_{32(s-1)}\}_{i\in[1,\mathcal{N}_p]}$ towards each unit s. Assume $T = 2^y$. We use ψ_s to denote the summation of binary records towards the same s as, $\psi_s = \sum_{i\in[1,\mathcal{N}_p]} b^{(i)}_{32(s-1)}$. Suppose ψ_s can be decomposed into n binary bits, $\{\psi^{(s)}_n, \ldots, \psi^{(s)}_2, \psi^{(s)}_1\}$, s.t, $\psi_s = \sum_{\rho\in[1,n]} 2^{\rho-1} \cdot \psi^{(s)}_\rho$. Define ϕ_s as the summation of decomposition bits from $\psi^{(s)}_n$ to $\psi^{(s)}_{y+1}$ as, $\phi_s = \sum_{k\in[y+1,n]} \psi^{(s)}_k$. It is sufficient to prove that for every $s \in [1, 64]$, there exists an integer γ_s s.t., $\phi_s \cdot \gamma_s = \theta_s$, where θ_s is a randomly chosen challenge integer from the verifier.

Besides, it requires to prove that all the decomposition bits $(\{w^{(i)}_v\}$ $_{i\in[1,\mathcal{N}_p],v\in[1,x]}, \{\psi^{(s)}_k\}_{k\in[y+1,n],s\in[1,64]})$ are all bits. We use the same technique as the main protocol here. To fix $\{w^{(i)}_v\}$, one need to further encrypt its summation u_i to c_{u_i}. $\{\psi^{(s)}_k\}$ doesn't need auxiliary encryption as the verifier can compute $\prod_i c_i$ and parse the encryption of ψ_s itself. Furthermore, one will need the statistical argument to prevent a cheating prover. We re-use the statistical argument in ZKAoK*. We define constraints $\mathcal{C}'_{\mathcal{R}^*}$ by changing the original constraints, $L'_j = \sum_{i,s} \ell^{(i)}_{j,32(s-1)} b^{(i)}_{32(s-1)} + R'_j$ into,

$$L'_j = \sum_{i,s,v,k} \ell^{(i)}_{j,32(s-1)} b^{(i)}_{32(s-1)} + \ell'^{(i)}_v w^{(i)}_v + \ell'^{(s)}_k \psi^{(s)}_k + R'_j,$$

where $\{\ell^{(i)}_{j,32(s-1)}\}$, $\{\ell'^{(i)}_v\}$, and $\{\ell'^{(s)}_k\}$ are randomly chosen from $\{0, 1\}$ by the verifier and $i \in [1, \mathcal{N}_p]$, $s \in [1, 64]$, $v \in [1, x]$,

$k \in [y+1, n]$, and $j \in [1, \kappa]$. Then constraints $\mathcal{C}_{\mathcal{R}^+}$ are as follows,

$$\begin{cases} \mathcal{C}'_{\mathcal{R}^*} & \\ \mathsf{Const}_{\{c_{u_i}, u_i, r_{u_i}\}} & \forall i \in [1, \mathcal{N}_p] \\ u_i = \sum_{s=1}^{64} b^{(i)}_{32(s-1)} & \forall i \in [1, \mathcal{N}_p] \\ u_i = \sum_{v \in [1,x]} 2^{v-1} w^{(i)}_v & \forall i \in [1, \mathcal{N}_p] \\ w^{(i)}_v (w^{(i)}_v - 1) = 0 & \forall v \in [1, x], \forall i \in [1, \mathcal{N}_p] \\ \psi_s = \sum_{i \in [1, \mathcal{N}_p]} b^{(i)}_{32(s-1)} & s \in [1, 64] \\ \psi_s = \sum_{\rho \in [1,n]} 2^{n-1} \cdot \psi^{(s)}_\rho & s \in [1, 64] \\ \phi_s = \sum_{k \in [y+1, n]} \psi^{(s)}_k & s \in [1, 64] \\ \psi^{(s)}_k (\psi^{(s)}_k - 1) = 0 & \forall k \in [y+1, n], \forall s \in [1, 64] \\ \phi_s \cdot \gamma_s = \theta_s & s \in [1, 64], \end{cases} \tag{4.6}$$

ZKAoK^+ runs the same as our main protocol, based on the above constraints. The verifier chooses $\{\theta_s\}$ along with $\{\ell^{(i)}_{j,32(s-1)}\}$, $\{\ell'^{(i)}_v\}$, and $\{\ell'^{(s)}_k\}$. It is noted that we only give a naive solution towards $\mathcal{R}^+$ and additional optimizations are possible. One can easily optimize it by encrypting several $\{u_i\}$'s into one message, as each u_i contains at most 64 bits while a Paillier plaintext is 2048 bits when $|N| = 2048$.

4.4 A Range Proof Protocol

We further construct a range proof protocol ZKAoK'. Specifically, we are going to prove that polynomially many plaintexts $\{m_i\}$ are in the same range, say, $[0, 2^{256}]$, the relation $\mathcal{R}'$ is set as follows,

$$\begin{aligned}\mathcal{R}' = \{(\{c_i\}_{i\in[1,\mathcal{N}_p]}, N, 2^{256}), (m_i, r_i, b_k^{(i)})_{i\in[1,\mathcal{N}_p],k\in[0,255]} : \\ \forall\, i \in [1,\mathcal{N}_p],\ c_i = (1+N)^{m_i}\cdot r_i^N \mod N^2 \wedge m_i \leq 2^{256}\}. \quad (4.7)\end{aligned}$$

We set $|N| = 2048$ where $|p| = |q| = 1024$. We use $\{b_{256}^{(i)}, \ldots, b_2^{(i)}, b_1^{(i)}\}$ to denote the decomposition of m_i. To prove $\mathcal{R}'$ with the above setting, it is sufficient to prove that 1) the decomposition elements of m_i are all 0 or 1; 2) These 256 bits can re-construct m_i; 3) c_i is a valid ciphertext for m_i. One thing that should be addressed is that as we consider proving 256-bit messages, which already satisfies the length requirement, $|m_i| < \min\{|p|, |q|\}$, we do not have to construct auxiliary messages m_t^* here. Then the constraints $\mathcal{C}'$ are,

$$\begin{cases} \mathsf{Const}_{\{c_i,m_i,r_i\}},\ \forall i \in [1,\mathcal{N}_p] \\ \mathsf{Const}_{\{c'_j,R'_j,r'_j\}},\ \forall j \in [1,\kappa] \\ b_k^{(i)}\cdot(b_k^{(i)}-1) = 0,\ \forall i \in [1,\mathcal{N}_p], \forall k \in [1,y] \\ L'_j = \sum_{k=1}^{256}\sum_{i=1}^{\mathcal{N}_p} l_{j,k}^{(i)}\cdot b_k^{(k)} + R'_j,\ \forall j \in [1,\kappa] \\ m_i = \sum_{k=1}^{y} 2^{k-1}\cdot b_k^{(i)},\ \forall i \in [1,\mathcal{N}_p], \end{cases} \quad (4.8)$$

where $l_{j,k}^{(i)} \leftarrow \{0,1\}$, is randomly chosen from the verifier for each $b_k^{(s)}$ in constructing L_j' and $R_j' \leftarrow \mathcal{L}$. We can set $\mathcal{L} = [0, 2^{281}]$ to hide bits when having millions of messages in the above setting. Besides, same as our main protocol, the prover should send $\{c_j'\}$ before the verifier chooses challenges $\{l_{j,k}^{(i)}\}$.

4.5 Security Proof

In this section, we analyze the security of our main protocol as others are similar to analyze. It is easy to verify completeness of the protocol. Also, special honest verifier zero-knowledge property comes from the special honest verifier zero-knowledge property of the underlying zero-knowledge argument system and security of the Paillier encryption scheme. In particular, as $R_j' \leftarrow [0, 2^{281}]$ and (an honestly generated) $S_j' = \sum_{i=1}^{\mathcal{N}_p} \sum_{s=1}^{64} l_j^{(s,i)} \cdot b_{32(s-1)}^{(i)} \leq 2^{26}$ (where we take $\mathcal{N}_p \leq 2^{20}$), the two distributions R_j' and $S_j' + R_j'$ are statistically close and thus $L_j' = S_j' + R_j'$ will not leak information about S_j'.

Next, we argue the special soundness of our protocol. It is sufficient to show that $\mathcal{R}^*$ is equivalent to the constraints specified by $\mathcal{C}_{\mathcal{R}^*}$ specified in Fig. 4.2 . Here, we focus on showing that if $\mathcal{C}_{\mathcal{R}^*}$ holds, then each $b_{32(s-1)}^{(i)}$ is from $\{0,1\}$ with all but negligible probability. The remaining parts are trivial to check.

Let $\bar{q} = q^{-1} \mod p$, $\bar{p} = p^{-1} \mod q$, then $b_{32(s-1)}^{(i)} = b_{32(s-1)}^{(i)} \cdot b_{32(s-1)}^{(i)}$ implies that $b_{32(s-1)}^{(i)} \in \{0, 1, q \cdot \bar{q}, p \cdot \bar{p}\}$. Note that both $q \cdot \bar{q}$ and $p \cdot \bar{p}$ are much larger than p, q, thus, it is sufficient to

show that $b^{(i)}_{32(s-1)}$ is small. We complete this task by revealing a random subset sum of all $b^{(i)}_{32(s-1)}$. If there exists some large $b^{(i)}_{32(s-1)}$, then at least half of the subset sums will be large (we explain this latter in Lemma 4.5.2). Thus, we can bound $b^{(i)}_{32(s-1)}$ via showing that random subset sums of all $b^{(i)}_{32(s-1)}$ are always small. One subtle issue here is that a (malicious) prover may use different $b^{(i)}_{32(s-1)}$ to answer different challenges. We solve this issue by committing all $b^{(i)}_{32(s-1)}$ in the beginning. In particular, we show in Lemma 4.5.3 that the Paillier encryption can play the role of commitment in this case. Specifically, we formalize the above proof idea in Lemma 4.5.1 to Lemma 4.5.3. We give the whole proof of our main protocol in Theorem 4.5.1.

Lemma 4.5.1. *There are 4 possible values of b_i (0, 1, X and Y) satisfying constraint, $b_i(b_i - 1) = 0 \mod N^2$. There are only 2 possible values of b_i, 0 and 1, satisfying the same constraint, under modulo p or q. N is the product of these two prime numbers, s.t., $N = p \cdot q$.*

Proof. (of Lemma 4.5.1) There are 4 possible values of b_i satisfying the above constraint modulo N^2, i.e., $0, 1, X$ and Y, where X denotes $q^2 \cdot [(q^2)^{-1} \mod p^2]$ and Y denotes $p^2 \cdot [(p^2)^{-1} \mod q^2]$. The following table shows possible values of b_i under modulo p and q respectively.

From the table we can see that there are only 2 possible values

Table 4.2: Possible values of b_i

Value of b_i	0	1	X	Y
Value of $b_i \mod p$	0	1	1	0
Value of $b_i \mod q$	0	1	0	1

of b_i under modulo p or q , 0 or 1. □

Lemma 4.5.2. *Let B be a set $\{b_1, \ldots, b_k\}$, where $b_i \in \{0, 1, X, Y\}$ for $1 \leq i \leq k$. Let $L = b_1 \cdot l_1 + b_2 \cdot l_2 + \cdots + b_k \cdot l_k + R \mod N^2$, where $l_i \leftarrow \{0, 1\}$, $1 \leq i \leq k$, and $R \leftarrow \mathcal{L}$. Define the event that there exists i in B such that $b_i \notin \{0, 1\}$ as NonBits; Otherwise, define the event as Bits. Let $M = k + \max\{R\}$. Assuming that N is a correctly generated RSA modulus and $M < \min\{X, Y\} < N^2 - M$, we have,*

$$\Pr[L \leq M \mid \mathit{NonBits}] \leq \frac{1}{2}.$$

Please note that L is computed modulo N^2. Then for $1 \leq j \leq t$, define $L_j = b_1 \cdot l_1^{(j)} + b_2 \cdot l_2^{(j)} + \cdots + b_i \cdot l_i^{(j)} + \cdots + b_k \cdot l_k^{(j)} + R_j \mod N^2$, where $l_i^{(j)} \leftarrow \{0, 1\}$ and $R_j \leftarrow \mathcal{L}$ is chosen from the same range as R. We have, $\Pr[L_1 \leq M \wedge L_2 \leq M \wedge \cdots \wedge L_t \leq M \mid \mathit{NonBiits}] \leq \frac{1}{2^t}$.

Proof. (of Lemma 4.5.2) Firstly, we prove the first probability. Let $U = \{0, 1\}^k$ be the challenge space for random challenges, $l_1, l_2, \ldots, l_k$. Let $M = k + \max\{R\}$, where $R \leftarrow \mathcal{L}$. If NonBits happens, that is to say, there exists i, s.t., $b_i = \{X, Y\}$. Let $\{l_1, \ldots, l_i, \ldots, l_k\}$ be challenge bits such that,

$$L = b_1 \cdot l_1 + \cdots + b_i \cdot l_i + \cdots + b_k \cdot l_k + R \leq M \mod N^2.$$

Then for another challenge set, $\{l_1, \ldots, \bar{l}_i, \ldots, l_k\}$, we define,

$$\bar{L} = b_1 \cdot l_1 + \cdots + b_i \cdot \bar{l}_i + \cdots + b_k \cdot l_k + R \mod N^2,$$

where $\bar{l}_i$ indicates the reverse of l_i. We can have, $\bar{L} = L \pm X(resp.\ Y) > M \mod N^2$, since X (resp. Y) is a big integer. That is, there must exist at least half of the challenges in challenge space U such that it can satisfy $\bar{L} > M \mod N^2$. Thus, $\Pr[L \leq M \,|\mathsf{NonBits}] \leq \frac{1}{2}$.

Similarly, for the same set B with $b_i \in \{X, Y\}$, if it can satisfy,

$$L_j = b_1 \cdot l_1^{(j)} + b_2 \cdot l_2^{(j)} + \cdots + b_i \cdot l_i^{(j)} + \cdots + b_k \cdot l_k^{(j)} + R_j \leq M \mod N^2.$$

Then for every $j \in [1, t]$, we can construct $\bar{L}_j$ as,

$$\bar{L}_j = b_1 \cdot l_1^{(j)} + b_2 \cdot l_2^{(j)} + \cdots + b_i \cdot \bar{l}_i^{(j)} + \cdots + b_k \cdot l_k^{(j)} + R_j > M \mod N^2,$$

where $R_j \leftarrow \mathcal{L}$ is chosen from the same range as R. Thus for each $L_j \leq M \mod N^2$, we can construct $\bar{L}_j > M \mod N^2$ as before. Similarly, we can have,

$$\Pr[L_j \leq M \mid \mathsf{NonBits}] \leq \frac{1}{2},$$

for all j. As all challenge bits are independently chosen, then the probability that all L_j can satisfy $L_j \leq M \mod N^2$ is,

$$\Pr[L_1 \leq M \ \wedge \cdots \wedge \ L_t \leq M \mid \mathsf{NonBits}]$$

$$= \Pr[L_1 \leq M \mid \mathsf{NonBits}] \cdots \Pr[L_t \leq M \mid \mathsf{NonBits}] = \frac{1}{2} \cdots \frac{1}{2} \leq \frac{1}{2^t},$$

which ends the proof. □

Lemma 4.5.3. *For every message $m < N^2$, there exists at most one possible solution set $\{b_0, b_1 \ldots, b_u\}$ satisfying the following constraints,*

$$c = (1+N)^m \cdot r^N \mod N^2, \tag{4.9}$$

$$m = 2^u \cdot b_u + \cdots + 2 \cdot b_1 + b_0 \mod N^2, \tag{4.10}$$

$$0 = b_i \cdot (b_i - 1) \text{ for } i \in [0, u] \mod N^2, \tag{4.11}$$

provided that the following inequality holds,

$$u + 1 < \min\{|p|, |q|\}, \tag{4.12}$$

where $N = pq$.

Proof. (of Lemma 4.5.3) To prove this lemma, we use contradiction. Assume that there exists two different sets, $\{b_i\}_{i=0}^u$ and $\{b'_i\}_{i=0}^u$, such that a j exists where $b_j \neq b'_j$ for $0 \leq j \leq u$, satisfying constraints 4.9 to 4.11. They can construct messages m and m' respectively, with the same encryption c. We denote these two sets as, $(c, m, \{b_i\}_{i=0}^u)$ and $(c, m', \{b'_i\}_{i=0}^u)$. According

to constraint 4.9, we can get, $m = m' + kN \mod N^2$, where $k > 0$. Constraint 4.10 provides,

$$m = 2^u b_u + \cdots + 2b_1 + b_0 \mod N^2, \tag{4.13}$$

$$m' = 2^u b'_u + \cdots + 2b'_1 + b'_0 \mod N^2. \tag{4.14}$$

Equation 4.13 minus equation 4.14 yields,

$$d = kN = 2^u(b_u - b'_u) + \cdots + 2(b_1 - b'_1) + b_0 - b'_0 \mod N^2. \tag{4.15}$$

Consequently, $d = 0 \mod N$. With $N = pq$, this evolves to, $d = 0 \mod p$ $(resp.\ q)$. We use $d_i = b_i - b'_i$, to denote the gap difference between each element in $\{b_i\}_{i=0}^u$ and $\{b'_i\}_{i=0}^u$. We can get,

$$d = 2^u d_u + \cdots + 2d_1 + d_0 = 0 \mod p\ (resp.\ q). \tag{4.16}$$

We let $e_i = d_i \mod p = (b_i - b'_i) \mod p$, and $f_i = d_i \mod q$. When considering modulo p, Equation 4.16 can be rewritten as,

$$d = 2^u e_u + \cdots + 2e_1 + e_0 = 0 \mod p. \tag{4.17}$$

According to constraint 4.11, we have b_i $(resp.\ b'_i) \in \{0, 1, X, Y\}$. When it goes to modulo a prime number p, they can only be 0 or 1. Thus, we can have $e_i \in \{-1, 0, 1\}$. Given $u + 1 < \min\{|p|, |q|\}$, Equation 4.17 is still satisfied without the final

modulo operation, and we can have,

$$e = 2^u e_u + \cdots + 2e_1 + e_0 = 0. \tag{4.18}$$

To satisfy the above equation, $e_i = 0$ must hold for every $i \in [0, u]$. Similarly, we can prove that $f_i = 0$ must hold for every $i \in [0, u]$. This implies that $d_i = 0 \mod p$ and $d_i = 0 \mod q$. Additionally, since we have b_i (resp. b'_i) $\in \{0, 1, X, Y\}$, it follows that $d_i \in \{0, 1, X, Y, -1, -X, -Y, 1 - X, 1 - Y, X - 1, X - Y, Y - 1, Y - X\}$.

Now, assuming $d_i \neq 0$, we prove that either $d_i \neq 0 \mod p$ or $d_i \neq 0 \mod q$. We first assume w.l.o.g. that $X = 1 \mod p$, $X = 0 \mod q$, $Y = 0 \mod p$, $Y = 1 \mod q$. Then if $d_i \in \{1, X, 1 - Y, X - Y\}$, we will have $d_i = 1 \mod p$. If $d_i \in \{-1, -X, Y - 1, Y - X\}$, then we can derive $d_i = -1 \mod p$. If $d_i \in \{1, Y, 1 - X, Y - X\}$ or $d_i \in \{-1, -Y, X - 1, X - Y\}$, we can have that $d_i = 1 \mod q$ or $d_i = -1 \mod q$, respectively. As the above contradicts the fact that $d_i = 0 \mod p$ and $d_i = 0 \mod q$, we can derive that d_i cannot be other variables except 0, which ends our proof. □

Looking ahead, Theorem 4.5.1 requires the condition $\mathcal{N}_b + \max\{R'_j\} < \min\{X, Y\} < N^2 - \mathcal{N}_b - \max\{R'_j\}$ ($\mathcal{N}_b$ denotes the number of proved bits) in order to apply Lemma 4.5.2. Below we show that this condition is met in our application scenario.

Let $M = \mathcal{N}_b + \max\{R'_j\}$. We show that if $\min\{p^2, q^2\} > M$ holds, the condition $M < \min\{X, Y\} < N^2 - M$ is satisfied.

First, recall that $X = q^2 \cdot [(q^2)^{-1} \mod p^2]$ and $Y = p^2 \cdot [(p^2)^{-1} \mod q^2]$. We have $\max\{X\} = q^2 \cdot (p^2 - 1) = N^2 - q^2$. Likewise, $\max\{Y\} = N^2 - p^2$. Similarly, we have $\min\{X\} = q^2$ and $\min\{Y\} = p^2$.

Since $\min\{p^2, q^2\} > M$, we have $\min\{X, Y\} = \min\{p^2, q^2\} > M$, i.e., the first inequality holds. For the second inequality, we have $\min\{X, Y\} < \max\{X, Y\} = \max\{N^2 - p^2, N^2 - q^2\} = N^2 - \min\{p^2, q^2\} < N^2 - M$, i.e., the second inequality holds.

In our application scenario, let s_d and λ be the statistical and security parameters, and $\mathcal{N}_p \leq 2^{20}$. Then $\mathcal{N}_b = |N| * \mathcal{N}_p/32$, where $N = \mathsf{PL.Gen}(1^\lambda)$. We set $\max\{R'_j\} = 2^{s_d} * \mathcal{N}_b$. For $\lambda = 128$, $|N| = 2048$, we can set $s_d = 255$. Then $\mathcal{N}_b \leq 2^{26}$ and $M < 2^{400}$. Assuming p and q are of equal length, $\min\{p^2, q^2\} > M$ is readily met. To ensure this is true, we can require the private key owner to prove in zero-knowledge that N is correctly generated (from 2 equal-length primes).

Theorem 4.5.1. *If the condition $\mathcal{N}_b + \max\{R'_j\} < \min\{X, Y\} < N^2 - \mathcal{N}_b - \max\{R'_j\}$ holds, the argument presented in the main protocol (ZKAoK^*) using the protocol in [26] for relation $\mathcal{R}^*$ satisfies perfect completeness, perfect special honest verifier zero-knowledge, and statistical witness-extended emulation.*

Proof. (of Theorem 4.5.1) Define security and statistical parameters as λ and s_d. We have $\kappa = poly(\lambda)$ and $\mathcal{N}_b = |N| * \mathcal{N}_p/32$, where $N = \mathsf{PL.Gen}(1^\lambda)$. We set $\max\{R'_j\} = 2^{s_d} * \mathcal{N}_b$.

Perfect completeness is derived from the completeness property of the underlying protocol and straightforward inspection, as all valid witnesses satisfy the constraints specified in $\mathcal{C}_{\mathcal{R}^*}$. Special honest verifier zero-knowledge comes from the security of Paillier encryption and the underlying zero-knowledge argument system. Specifically, (an honestly generated) S'_j always satisfies, $S'_j = \sum_{i=1}^{\mathcal{N}_p} \sum_{s=1}^{|N|/32} l^{(i)}_{j,32(s-1)} b^{(i)}_{32(s-1)} \leq \mathcal{N}_b$ (where we fix the slot size as 32-bit). Consequently, the statistical distance between the distributions of R'_j and $L'_j = S'_j + R'_j$ is bounded by $\frac{1}{2^{s_d}}$. With choosing an appropriate value of s_d, the two distributions will be statistically indistinguishable ensuring that L'_j will not leak information about S'_j.

For witness extended emulation, it is sufficient to prove that $\mathcal{R}^*$ is equivalent to $\mathcal{C}_{\mathcal{R}^*}$, since the underlying protocol already satisfies this property. It is easy to check that $\mathcal{R}^*$ implies $\mathcal{C}_{\mathcal{R}^*}$. To prove that $\mathcal{C}_{\mathcal{R}^*}$ implies $\mathcal{R}^*$, we show that if $\mathcal{C}_{\mathcal{R}^*}$ holds, then each $b^{(i)}_{32(s-1)}$ is from $\{0,1\}$ with all but negligible probability. Using Lemma 4.5.3, we show that the Paillier encryption, with input message satisfying $|m| < \min\{|p|, |q|\}$, can be treated as a commitment scheme with the binding property. Consequently, all $\{b^{(i)}_{32(s-1)}\}$'s (which are decomposition bits of $\{m^*_t\}$'s) are fixed, as m^*_t satisfies this length requirement. By applying Lemma 4.5.1, we can have that $b^{(i)}_{32(s-1)} \in \{0,1,X,Y\}$. Therefore, by deploying Lemma 4.5.3, we can prove that if every L'_j can satisfy $L'_j \leq M$, where $M = \mathcal{N}_b + \max\{R'_j\}$ and $j \in [1,\kappa]$, then all $\{b^{(i)}_{32(s-1)}\}$'s, where $i \in [1,\mathcal{N}_p]$ and $s = |N|/32$, are from 0

or 1 except with probability $\frac{1}{2^\kappa} = \frac{1}{2^{poly(\lambda)}}$ (i.e., soundness error), which is negligible in λ. We can, therefore, prove that our main protocol satisfies witness extended emulation by calling the extractor in the underlying protocol to extract a valid witness.

In particular, when we set $\lambda = 128$, $|N| = 2048$, $s_d = 255$, and $\mathcal{N}_p \leq 2^{20}$ in our application scenario, we have $\mathcal{N}_b \leq 2^{26}$ and $\max\{R'_j\} = 2^{281}$. Therefore, the statistical distance between the two distributions of R'_j and L'_j is bounded by $\frac{1}{2^{255}}$, which is negligible. As a result, the zero-knowledge property can be satisfied. The soundness error is bounded by $\frac{1}{2^\kappa} = \frac{1}{2^{poly(128)}}$, which is also negligible. This implies that our protocol satisfies perfect completeness, perfect special honest verifier zero-knowledge, and statistical witness-extended emulation in our considered scenario, which ends our proof. □

4.6 Performance Evaluation

We implement our main protocol as a proof-of-concept to verify its practicality. Our implementation is written in C++ and we utilize NTL [Sho96], for big integer operations. We experiment on a PC with Linux version 6.0.7-arch1-1 (linux@archlinux) 12th Gen Intel(R) Core(TM) i7-12700F and 64 GB of RAM. Our implementation is single-threaded. Our result is given in the non-interactive form of proof.

To better facilitate our explanation, we recall notations and

their definitions in Table 4.3. We then illustrate our main protocol, ZKAoK*, in two aspects, namely, communication and computation. We give comparisons in 4.6.4, considering all our schemes. Two sets of parameters used in this section are specified in Table 4.4.

Table 4.3: Notations and Definitions.

Notation	Definition
κ	security parameter
U, $\mathcal{S}_b$	slot size, batch size
$\mathcal{N}_p$, $\mathcal{N}_b$	number of plaintext, number of binary messages to be proved
$\mathcal{S}_{N^2}$, $\mathcal{S}_Q$	element size in $\mathbb{Z}_{N^2}$, element size in $\mathbb{Z}_Q$
$\mathcal{N}_l$, $\mathcal{N}_m$	linear constraints number, multiplication constraints number
$\mathcal{N}_{om}$	number of multiplication operations
$\mathcal{N}_{oe}$	number of exponentiation operations
$\mathcal{T}_m$	time consumed in a single multiplication operation
$\mathcal{T}_e$	time consumed in a single exponentiation operation

Table 4.4: Parameter Sets Used in Our Experiment.

$\mathcal{N}_b/msg$ means the number of bits to be proved in a single plaintext.

Set No.	$\lvert N\rvert$	U	$\mathcal{N}_b/msg$	$\mathcal{S}_b$	κ
$s1$	1024	32 (bits)	32	15	80
$s2$	2048	32 (bits)	64	15	128

4.6.1 Communication Efficiency

Micro-benchmarks

We perform a series of micro-benchmarks to quantify the cost of each basic operation involved in our protocol, including time cost for one exponentiation or multiplication operation and the size of an element in $\mathbb{Z}_{N^2}$ and $\mathbb{Z}_Q$ respectively. We report these data in Table 4.5.

Table 4.5: Micro-benchmarks.

The following data is the median by running 1000 *times of corresponding opperations.*

$\|N\|$	$\mathcal{T}_e$ (ms)	$\mathcal{T}_m$ (ns)	$\mathcal{S}_{N^2}$ (bits)	$\mathcal{S}_Q$ (bits)
1024	2.09	1279.15	2048	2056
2048	13.96	3786.83	4096	4112

Total and Average Proof Size

To demonstrate communication complexity, we test and plot total and average proof size in Fig. 4.3. Total proof size includes all the communication cost from the prover to the verifier. The average proof size is computed by dividing total proof size by the number of proved binary messages (i.e., records).

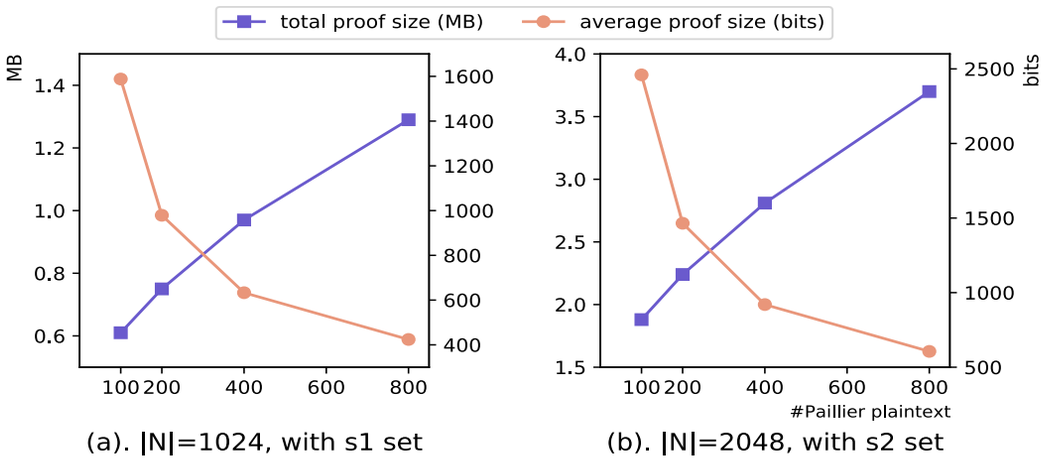

Figure 4.3: Total and Average Proof Size.

As the total proof size is not linear to the number of proved messages (bits), the average cost will decrease when the number of messages increases. For example, the total proof cost for 6.4K binary messages is 1.88 MB and this cost will increase to 3.7 MB when proving 51.2K binary messages, with setting $|N| = 2048$. On the contrary, the average proof cost per binary

message is reduced from 2460 bits to 606 bits in the same setting.

Estimate Communication Cost

Here, we give estimations on communication cost. We first estimate the number of multiplication and linear constraints and denote them as $\mathcal{N}_m$ and $\mathcal{N}_l$ respectively. $\mathcal{N}_m$ can be computed as,

$$\begin{aligned}\mathcal{N}_m &= \frac{|N|}{U} \cdot \mathcal{N}_p + (\mathcal{N}_p + \kappa + \lceil \frac{|N|}{\mathcal{S}_b} \rceil) \cdot (\lceil \log N \rceil + h(N) - 1) \\ &\approx \frac{|N|}{U} \cdot \mathcal{N}_p + (\mathcal{N}_p + \kappa + \lceil \frac{\mathcal{N}_p}{\mathcal{S}_b} \rceil) \cdot (|N| + |N|/2).\end{aligned} \tag{4.19}$$

The equation holds as we use the average hamming weight of N, which is $|N|/2$. Additionally, the number of linear constraints can be calculated as, $\mathcal{N}_l = 2\mathcal{N}_m$. The proof sent from the prover to verifier contains,

- Auxiliary ciphertexts $\{c'_j\}$ and $\{c^*_t\}$, and summation of bits $\{L'_j\}$, specified in $\mathcal{C}_{\mathcal{R}^*}$.
- Commitment used in [BCC$^+$16]

We estimate the total communication cost as,

$$\mathcal{S}_{N^2} \cdot O(\mathcal{N}_p/U + \sqrt{\mathcal{N}_p} + \kappa) + \mathcal{S}_Q \cdot O(\sqrt{\mathcal{N}_m} + \sqrt{\sqrt{\mathcal{N}_m}}). \tag{4.20}$$

We compare the estimated data with the real cost in Table 4.6, for both total and average case. The result shows our estimation matches the real data quite well.

Table 4.6: Comparison of Real and Estimated Proof Cost.

The average cost is computed by dividing the whole cost by the number of proved bits.

Set No.	$\mathcal{N}_p$	$\mathcal{N}_b$	Average Cost		Total Cost	
			Real (bits)	Est. (bits)	Real (MB)	Est. (MB)
s1	100	3.2K	1587.99	1570.28	0.61	0.60
	200	6.4K	978.73	969.35	0.75	0.74
	400	12.8K	632.90	626.68	0.97	0.96
	800	25.6K	423.86	419.96	1.29	1.28
s2	100	6.4K	2460.35	2460.36	1.88	1.88
	200	12.8K	1465.09	1464.77	2.24	2.24
	400	25.6K	920.45	920.93	2.81	2.81
	800	51.2K	605.92	604.96	3.70	3.69

Compare with A Baseline: Honest-but-curious Model

We further compare our main protocol with a honest-but-curious model. Here we examine the upgrading cost for providing active security in large scenarios (i.e., proving thousands of binary messages (i.e., records) once), utilizing the above estimation methods.

In the honest-but-curious model, thanks to the use of packing, the cost of encrypting a bit is $32 * 2 = 64$ *bits*, with $U = 32$ *bits*. When $|N| = 1024$, if one aims to prove ten thousand messages (2^{14}), the cost for proving the correctness of encryption and knowledge of inserted bits will be about 355.3 bytes per message. The amortized cost is 89 *bits*. That is, it takes an additional 89 bits per bit to upgrade the protocol to offer adaptive security. This is readily acceptable since adaptive secure

protocols are considered a much stronger requirement; If a scenario consists of millions (2^{20}) of messages, the upgrading cost will be about 15 *bits* per bit. Furthermore, by packing more bits into one message, this average cost will be further reduced. [3]

We compare and show more comparisons in Table 4.7. In most "large-scale" (with hundred thousands of binary messages (i.e., records)) scenarios, the cost of upgrading is less than the cost of encrypting. Then upgrading the original protocol to obtain active secure is certainly admissible.

Table 4.7: Comparison Between the Cost for Encrypting and Proving One Bit Using Our Protocol in Large Scenarios.

Set No.	$\mathcal{N}_p$	Encryption Cost/ Bit (bits)	Proof Cost/ Bit (bits)
s1	10 thousand	64	88.82
	100 thousand		**33.74**
	1 million		**14.62**
s2	10 thousand	64	123.26
	100 thousand		**45.72**
	1 million		**18.83**

* We use 2^{14}, 2^{17} and 2^{20} to denote 10 thousand, 100 thousand and 1 million.

Compare with the Second Baseline: Exponential ElGamal Encryption

As both exponential ElGamal and Paillier are selected as ISO standard [Hom19] for offering homomorphism, we choose exponential ElGamal as the second baseline. In exponential Elgamal, at 80-bit security level (i.e., Paillier public key N is 1024 bits), the cost of encrypting each bit will be 320 bits. For proof of correctness, it will also cost 320 bits for one bit. Thus the proof cost for each bit is 640 bits. In comparison, the total cost

[3] It should be noted that choose of slot size should be careful, considering the total number of plaintexts and the range of weights.

for one bit in Paillier is 89 bits, 34 bits, and 15 bits when we have 10 thousand, 100 thousand, and 1 million messages separately while the average encryption cost is 64 bits. That is, using our protocol is 4x - 8x more compact for the total cost of one bit.

4.6.2 Computation Efficiency

Time Consumed by Each Side

To analyze the computation efficiency of each side in our protocol, we first test the running time of prover and verifier separately. The result is shown in Fig. 4.4 (a). Considering proving 25.6K bits in $|N| = 1024$ setting, we have to pack them into 800 Paillier plaintexts. The total time required for the prover is 2689 s while the cost for the verifier is only 30 s. The time spent by the prover is about 89 times more than the verifier. If we consider proving the same number of plaintexts when $|N| = 2048$, this time gap can be enlarged to 189x. As it is fairly inefficient on the prover side, we group the prover's whole procedure into 4 phases and test their consumed time in Fig. 4.4 (b). Our data is obtained when proving 800 plaintexts in $|N| = 1024$ and $|N| = 2048$.

(a) Encryption phase. The prover does Paillier encryption for all messages $\{m_i\}$, $\{R_j\}$ and $\{c_t^*\}$, specified in $\mathcal{C}_{\mathcal{R}^*}$ of Fig. 4.2.

(b) Circuit creation phase. It generates all multiplication and linear constraints specified in $\mathcal{C}_{\mathcal{R}^*}$.

(c) Value assign phase. This is an inner phase in [BCC+16], where the prover assigns values to the circuit wires according to the constraint $\mathcal{C}_{\mathcal{R}^*}$.

(d) Commitment phase. This is an inner phase proposed in the protocol of [BCC+16]. The prover mainly does Pedersen commitment to circuit wires in this phase.

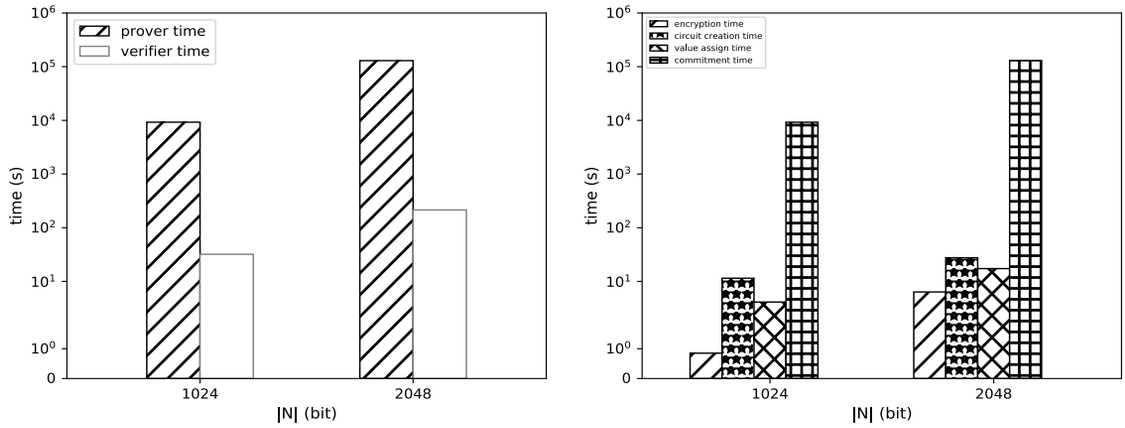

Figure 4.4: Total Time Consumed by the Prover and Verifier.
The evaluation result is given by proving 800 plaintexts in s1 and s2 settings.

We can see that the commitment phase is most expensive on prover side. In $|N| = 1024$ with proving 800 plaintexts, time consumed in this phase takes up 99.83% of the total proof time. When it goes to $|N| = 2048$, 99.96% of the total proof time is consumed in this phase. The reason that the time consumed in this phase is so prominent is that the prover should conduct a series of exponentiation and multiplication operations. We provide several optimization approaches in Subsection 4.7.

Estimate Poof and Verification Time

We use commitment time, the most dominant part, to denote total proof time. This is an internal phase used in [BCC+16]. The prover mainly does commitments in this phase, resulting in a series of exponentiation and multiplication operations, where their required number is associated with $\mathcal{N}_m$ and $\mathcal{N}_l$. $\mathcal{N}_m$ can be estimated by Equ. 4.19. The number of exponentiation and multiplication operations required in this phase can be computed by,

$$\begin{aligned} \mathcal{N}_{oe} &= (|N|/4+1) * (3\sqrt{\mathcal{N}_m}+1)/2 \\ \mathcal{N}_{om} &= |N| * (3\sqrt{\mathcal{N}_m}+1) * (\sqrt{\mathcal{N}_m}+512)/4. \end{aligned} \tag{4.21}$$

We use $T_{prf}^{est.} = \mathcal{N}_{oe} * \mathcal{T}_e + \mathcal{N}_{om} * \mathcal{T}_m$ to estimate the total proof time. For verification time, its cost is about,

$$T_{vrf}^{est.} = O(\mathcal{N}_l + \mathcal{N}_m) * \mathcal{T}_m + O(2\sqrt{\mathcal{N}_m}+1) * \mathcal{T}_e. \tag{4.22}$$

We compare our estimated proof and verify time with real consumed time in Table 4.8. It shows that our estimation approach matches well with the real data.

For proving bits inserted into millions (about 2^{20}) of 2048-bit Paillier messages, it will cost the prover and verifier about 0.31 s and 2.01 ms for one bit respectively. We provide more optimization approaches in Subsection 4.7 to speed up the time on the prover side.

Table 4.8: Comparison between Real and Estimated Time Cost for One Bit in Both Prover and Verifier Sides.

The following data is the average consumed time for each bit.

Set No.	$\mathcal{N}_p$	$\mathcal{N}_b$	Prover Side (Ave.)		Verifier Side (Ave.)	
			real (s)	est. (s)	real (ms)	est. (ms)
s1	100	3.2K	0.26	0.30	2.57	2.71
	200	6.4K	0.17	0.20	1.80	1.88
	400	12.8K	0.13	0.15	1.38	1.42
	800	25.6K	0.10	0.12	1.17	1.15
s2	100	6.4K	2.14	2.46	11.42	12.69
	200	12.8K	1.33	1.57	7.04	8.20
	400	25.6K	0.95	1.08	5.40	5.78
	800	52.6k	0.72	0.81	3.79	4.40

4.6.3 End-to-end Performance

We develop an end-to-end prototype for two-party aggregation, which leverages the voter analysis scenario discussed in Section 1 as context. Its performance is reported in Table 4.9. The results are given with $\mathcal{N}_p = 800$ under both $s1$ and $s2$ parameter sets. For ease of clarity, we retain the notations introduced in Section 1. The whole protocol comprises 3 phases: 1) $\mathcal{P}_1$ structures and encrypts messages for generating a proof π. 2) $\mathcal{P}_2$ verifies π and computes $\bar{C} = \sum_{i=1}^{\mathcal{N}_p} c_i^{W_i}$. 3) $\mathcal{P}_1$ decrypts $\bar{C}$ and parses the result to obtain $\{S_j\}$.

Table 4.9: End-to-end Performance Evaluation with $\mathcal{N}_p = 800$.

Parameter	Phase 1 (s)	Phase 2 (s)	Phase 3 (ns)	Proof (MB)
***s*1**	2689.36	30.00	5955	1.29
***s*2**	36785.80	194.07	9896	3.69

The first three columns denote the time cost of each stage, while the last column represents the proof size. When aggregating 25.6K bits with $|N| = 1024$, the protocol completes in 0.76 hours, requiring a total communication cost of 1.49 MB

(1.29 MB for proof + 0.20 MB for ciphertexts $\{c_i\}$). For aggregating 51.2K bits under $|N| = 2048$, the required time is 10.27 hours with a total communication cost of 4.08 MB.

4.6.4 Comparison

Here, we give a comparison between all our protocols and the potential solutions utilizing state-of-the-art tools.

Compare ZKAoK* with An OR-Proof

To evaluate the utility of packing used in our approach, we compare our main protocol with a standard OR-proof [CDS94], where the prover uses more (unpacked) Paillier ciphertexts and proves that each plaintext encrypts a Boolean value. Furthermore, as our approach supports batch proving and verification, we report more data on varying numbers of plaintexts ($\mathcal{N}_p$), ranging from small (single or dozens) to large (hundreds), to facilitate batch size selection in practice.

Table 4.10: Compare Ours with A Standard Or-proof.

Data under $s1$ / $s2$ parameters represent amortized values for ZKAoK and averages from 1000 repetitions for OR-proof. A "−" indicates non-applicability of the attribute.*

Schemes	Para.	$\mathcal{N}_p$	$\mathcal{N}_b$	Proof Size (bits)	Prove Time (s)	Verify Time (ms)	Enc. Size (bits)
OR-proof [CDS94]	$s1$ / $s2$	–	1	>8192 / >16384	4.08*10^{-3} / 0.35	2.00 / 18.04	2048 / 4096
ZKAoK*	$s1$ / $s2$	1	32 / 64	107.69K / 187.05K	14.65 / 143.38	147.94 / 779.91	**64 / 64**
		10	320 / 640	11.27K / 19.26K	1.55 / 14.80	15.90 / 82.42	
		20	640 / 1.28K	**5931.65** / **9978.29**	0.83 / 7.94	8.56 / 43.43	
		40	1.28K / 2.56K	**3228.11** / **5298.84**	0.47 / 4.44	4.81 / 23.40	
		80	2.56K / 5.12K	**1851.42** / **2937.45**	0.29 / 2.53	2.97 / 13.84	
		100	3.2K / 6.4K	**1563.53** / **2453.29**	0.25 / 2.14	2.57 / **11.42**	
		200	6.4K / 12.8K	**964.24** / **1461.24**	0.17 / 1.33	**1.80** / **7.04**	
		400	12.8K / 25.6K	**623.25** / **918.20**	0.13 / 0.95	**1.38** / **5.40**	
		800	25.6K / 51.2K	**422.57** / **605.08**	0.10 / 0.70	**1.17** / **3.79**	

The comparison result is reported in Table 4.10. In the OR-proof, the cost for each bit is constant, while in our cases, the amortized cost decreases when proving more plaintexts (bits) simultaneously. In both the $|N| = 1024$ and $|N| = 2048$ settings, our method yields a smaller amortized proof size than the OR-proof when more than 20 plaintexts are being proven. In the $|N| = 2048$ setting, when over 100 plaintexts are being proven, both our proof size and verification time outperform the OR-proof.

Our method becomes significantly more efficient as the number of plaintexts increases. When proving between 25.6K and 51.2K bits under $|N| = 2048$ setting, our amortized proof size is around 17.8x - 27x smaller. This gap will further expand to 133x when proving 10 thousand messages. As a result, our amortized bandwidth cost (proof+encryption) is only 982-670 bits, which is 20x - 30x lower than that of a standard OR-proof.

Indeed, when proving a single or a small number of messages (i.e., 10-20), our cost can remain relatively high. This is due to the need to construct auxiliary ciphertexts, c'_j, to ensure soundness. Therefore, ZKAoK* is better suited when the number of plaintexts is 'larger', typically on the scale of hundreds. Conversely, for proving a single bit or several bits, the naive solution is more appropriate. Nonetheless, we underscore that this requirement is compatible with our motivating scenarios presented in Section 1, where each election poll gathers at least

hundreds to thousands of messages daily. As such, our approach is ideally suited for these large-scale scenarios.

Compare ZKAoK′ with Paillier Range Proofs [Lin17, Bou00]

There are state-of-the-art methods in [Lin17, Bou00] for range proving Paillier messages. Their implementation can be found in [ran18]. We compare the result of range proving 256-bit plaintexts under $|N| = 2048$ and $\kappa = 128$ setting and plot the result in Table 4.11.

Table 4.11: Compare Ours with [Lin17, Bou00] for Range Proving 256-*bit* Paillier Plaintexts in $|N| = 2048$ Setting.

Scheme	$\mathcal{N}_p$	Proof Size/Message	Total Proof Size	Proof time/meessage	Verify time/Message
[ran18]	10 thousand	>128.00 KB	>2.00 GB	241.43 ms	199.32 ms
Ours		0.93 KB	14.87 MB	140.45 s	120.49 ms
[ran18]	1 million	>128.00 KB	>128 GB	241.43 ms	199.32 ms
Ours		931.84 bit	116.48 MB	139.43 s	97.68 ms
[ran18]	10 million	> 128.00 KB	>2048.00 GB	241.43 ms	199.32 ms
Ours		232.12 bit	0.45 GB	139.42 s	95.31 ms

* We use 2^{14}, 2^{20} and 2^{24} to estimate 10 thousand, 1 million and 10 million.

The communication and computation costs for each plaintext in method [Lin17, Bou00] are constant. For proving one plaintext, without considering other costs (e.g., $\{z_i\}_i$ used in [Lin17]), it requires $4096 * 2 * 128$ bits $= 128$ KB, which is 256 larger than encryption cost (4096 bits). Besides, it requests almost the same time in verification and proof.

Ours is competitive in proof size and verification time while can provide exact range proof. Specifically, we only require 11.4 KB per message when proving 200 messages at a time. When proving 10 million messages, their approach requires $>$ 2048 GB size in total while ours only requires about 0.45

GB. The verification time for each message in our protocol is about half of theirs, though they optimized their code by parallelization while ours only uses single thread. (Without parallelization, theirs requires 1.67 s and 1.25 s for proof and verification respectively.) In proving 10 million messages, they require $>$ 2048 GB of communication while ours only requires 0.35 GB, which is around 5000x smaller. Besides, they can can only guarantee that $x \in [0, q)$ with input $x \in [0, \lfloor q/3 \rfloor)$ while ours can provide precise proof.

Compare ZKAoK^+ with Existing Work

As there are no existing work designed for relations $\mathcal{R}^+$, we give potential approaches other than ours.

For proving sum of records property, one should encrypt all bits $\{b^{(i)}_{32(s-1)}\}_{i,s}$ as $c^{(i)}_{32(s-1)}$ and prove that there are at most t 1's among all bits of $\{b^{(i)}_{32(s-1)}\}_s$. We use $C^{(i)}$ to denote the encryption of summation of $\{b^{(i)}_{32(s-1)}\}_s$, s.t., $C^{(i)} = \mathsf{Enc}(\sum_s b^{(i)}_{32(s-1)}) = \prod_s c^{(i)}_{32(s-1)} = \mathsf{Enc}(M^{(i)})$, where $M^{(i)} = \sum_s b^{(i)}_{32(s-1)}$. Then after proving that each $\{b^{(i)}_{32(s-1)}\}$ is 0 or 1 using an OR-proof, one should further prove that $M^{(i)}$ is no larger than t. Suppose $M^{(i)}$ can be decomposed by a set of binary bits $\{B^{(i)}_{n-1}, \ldots, B^{(i)}_0\}$. Then one should further encrypt $\{B^{(i)}_x\}_{x\in[0,n-1]}$ and prove that each $B^{(i)}_x$ is 0 or 1 using an OR-proof, together with proving that, $M^{(i)} = 2^{|t|-1} \cdot B^{(i)}_{|t|-1} + \cdots + 2^1 \cdot B^{(i)}_1 + 2^0 \cdot B^{(i)}_0$. This relation can be translated by a sequence of multiplication relations, $\mathcal{R}_{\mathsf{mul}}$, which can be handled by [DJ01a].

However, it is not trivial by utilizing existing tools for proving the sum of entities property. As the existing range proof protocol is to prove that an integer, x, is in a range $[0, t]$, with input $x \in [0, t/3]$. To provide the entity property, one possible approach is to first encrypt each bit $\{b^{(i)}_{32(s-1)}\}$ and prove that each bit is 0 or 1 using a standard OR-proof for each bit. One then does integer commitment (which is usually homomorphic) to each $\{b^{(i)}_{32(s-1)}\}$ as $\{c^{(i)}_{32(s-1)}\}$ and the threshold T as c_T. One proves that the plaintext in, $\prod_i \{c^{(i)}_{32(s-1)}\}/c_T$, is larger than 0.

The above potential approaches using existing tools are much more involved than ZKAoK^+ while requiring multiple auxiliary commitments and encryption, which may result in large proof size. On the contrary, ours can be easily extended by adding multiplication and linear constraints into the main protocol and batch prove all these constraints at a time.

4.7 Potential Optimizations and Discussions

4.7.1 Potential Optimizations.

Since many workloads in the commitment phase can be precomputed, we can divide our protocol into online-offline phases. Here we have a closer look at the required multiplication constraints. Most of them have the Paillier encryption form as, $c = (1+N)^m \cdot r^N$ (while others have the form, $b * (b-1) = 0$). We need to construct 1 linear constraint and $(\lceil \log N \rceil + h(N) - 1)$

multiplication constraints to represent $1 + m \cdot N$ and r^N respectively. As these multiplication constraints are only related to randomness, not witness, we can pre-commit them before running our protocol. The number of multiplication constraints that can be pre-computed is,

$$(\mathcal{N}_p + \kappa + \lceil \frac{\mathcal{N}_p}{\mathcal{S}_b} \rceil) \cdot (\lceil \log N \rceil + h(N) - 1).$$

That is, over 98% commitments can be pre-computed.

Besides, as our implementation uses single-thread, we can further optimize it by applying parallelization.

4.7.2 Discussions.

In this subsection, we provide more discussions on the potential application scenarios that can utilize our technique, as well as how to prove other relations among Paillier plaintexts.

Discussion of More Application Scenarios. We further explore potential scenarios that could benefit from our approaches. Practical applications, including transaction evaluation [Ord03, ZH08], advertisement targeting [TNB+10, LVKF16], and social relationship analysis [LWC12, AHKL12], may find our approaches advantageous. In the context of a transaction evaluation, for instance, our method enables companies to identify target consumers' income groups, in collaboration with banks, where W_i represents the consumer's salary. Additionally, entities like Google [adw, ads], which provide advertising campaigns for their clients

based on online ad targeting [WRF+, YLW+], can use our approach to assess campaign effectiveness. Typically, these entities utilize binary vectors to represent various marketing strategies. The effectiveness can be privately evaluated through our method by calculating the average conversion value, where W_i denotes the cost paid by each consumer.

Beyond these specific applications, our methodology gives potential advantages in scenarios requiring the use of correctly-structured binary, a common occurrence in fields such as machine learning [LFY+17, HYK10], cryptographic schemes [SW21, HK21], image and audio processing [CYL+20, ZWDY21], graph theorems, and so on. For instance, utilizing binary (or one-hot) vectors to encode a label is a common practice in classification [GHH+21, LLL+20] for regression tasks [LZF+20, DM19], which require that each vector can only contain a '1' and others should be '0'. Similarly, in computing a special case of inner product [SW21], a "selection vector" is required, where only one coordinate is set to 1. We would like to emphasize that any constructions involving (structured) binary vectors can gain adaptive security through our approach.

Discussion of Proving Other Plaintext Relations Using Our Approach. Here we discuss how to additionally prove other Paillier plaintext relations using our approach. For example, to prove the relation $\mathcal{R}_{\mathsf{mul}}$ in [DJ01a], one can use our method by adding one constraint, $m_3 = m_1 * m_2$, into the system. One thing need

to be argued is that our constraint is under modulo N^2 where the original relation is under modulo N. We now give the following observation.

If we have $m_3 = m_1 \cdot m_2 \mod N^2$, then we can rewrite it as $m_3 = m_1 \cdot m_2 + kN^2$ for some $k \in \mathbb{Z}$. Then we can have, $m_3 = m_1 \cdot m_2 + kN \cdot N = m_1 \cdot m_2 \mod N$. We use c_1, c_2, c_3 to denote their corresponding ciphertexts. Even if we put $m_1 > N$, say, $m_1 = m_1' + k_1 N$ (resp. $m_2 = m_2' + k_2 N$ and $m_3 = m_3' + k_3 N$) into the encryption, the actual value being encrypted is still m_1' (resp. m_2' and m_3'). So if $m_3 = m_1 \cdot m_2 \mod N^2$ holds with having $\mathsf{PL.Enc}(m_1; r_1)$, $\mathsf{PL.Enc}(m_2; r_2)$ and $\mathsf{PL.Enc}(m_3; r_3)$, we can only get m_1', m_2' and m_3' after decryption. Therefore we still have $m_3' = m_1' \cdot m_2' \mod N$.

Discussion of Using a Generic Two-party Computation Protocol. While generic two-party computation protocols can potentially provide similar functionality, current efficient advances [BHKR, ZRE] can only achieve semi-honest security. To achieve malicious security, one can additionally use a generic ZKP. However, this will bring in large overhead and break its efficiency advantage. Although one might consider using adaptive-secure two-party computation protocols, existing works such as [RR16, WRK] fall short in terms of round and communication efficiency. Customized maliciously secure OT/OLE schemes [DGN$^+$] could be an alternative but these require greater computation and communication resources. We emphasize our non-interactive

approach wherein $\mathcal{P}_1$ can reuse proofs across collaborations. In contrast, a two-party computation protocol necessitates $\mathcal{P}_1$ to rerun the scheme with each collaborator, increasing the complexity.

Discussion of the choice of slot size and the number of slots. For ease of concreteness, we have fixed these parameters, although they can be adjusted according to the problem size and its requirements. For instance, the slot size U can be decreased or increased, provided that it meets the condition, $\mathcal{N}_p * \max\{W_i\} < 2^U$. Similarly, the number of slots ($\mathcal{N}_b/msg$) in each plaintext can also be adjusted but should satisfy $\mathcal{S}_b * \mathcal{N}_b/msg < |N|/2$. This condition is derived from Lemma 4.5.3 and is necessary for constructing auxiliary message m_t^*.

Chapter 5

Blockchain-based Threshold Electronic Voting Systems

Voting, as a fundamental social function, has been considered moving to the online mode, with providing more flexibility and convenience to voters. However, it is not that straightforward to move this service onlice as privacy, security and trust are main concerns in this whole process.

To be more specific, for instance, the security of the preference of each voter must be protected, as all processes are moved online. It is not acceptable that the voters' ballots could be revealed (or even leaked) before publishing the final results. Additionally, trust is also a another big issue that must be considered, for example, how to tally and publish the final result. If a single entity is responsible for this procedure, we have to trust this single node and it is vulnerable to be corrupted. Besides, the choice of the public bulletin board is also significant as we should guarantee that the bulletin board can always publish the correct final result without malicious modifications. In practice, the bulletin board is always played by a trusted third

party. However, as the third party also could also be corrupted and this assumption is not that desirable, one fundamental remaining question is that how we can get rid of this assumption.

Based on above obstacles, we propose a truly decentralized e-voting system based on Blockchain, without relying on a single trusted third party. We employ threshold signature and encryption schemes to distribute the authorization and tallying rights to multiple players. As there are at no more than the threshold number of malicious participants, the authority cannot be abused. In this way, we reduce the underlying trust assumption of our system. Furthermore, we also deploy blind signature to hide voters' identities. in order to preserve their privacy.

Chapter Organization. We summarize our main contributions in Section 5.1, followed by giving a high-level overview in Section 5.2. We present the syntax of an e-voting system and its security requirements in Section 5.3. Our detailed construction and its security analysis are given in Section 5.4. We report the performance evaluation in Section 5.5.

5.1 Our Contribution

In this thesis, we propose an e-voting system using distributed blind signature, encryption and blockchain. Specifically, we system offers the following features.

1. *Our system does not rely on a single trusted third party.* Using a threshold blind signature scheme, the role of registration authority is played by n organizers in our system. Likewise, n^* tellers play the role of tallying authority. Through combining these two techniques, our system does not rely on a single trusted third party.

2. *Our system is distributed.* We distribute the capability of registration and tallying in a round efficient manner. This matches the inherent decentralized nature of blockchain perfectly. Thus, our system can is truly distributed.

3. *Our system is anonymous.* We use blind signature to protect the voters' identity. Specifically, even if the set of registration authority colludes, they will not be able to link a ballot to a registered voter.

4. *Our system is efficient.* We implement our system to evaluate its efficiency. From the experimental results, the performance of our system is efficient enough to be adopted in practice. Notably, the time and complexity at the user side is constant.

5.2 Overview of Our Solution

In this section, we give a high-level overview of how we construct a fully distributed e-voting system based on blockchain. In general, we construct the system by combining a (t, n)-threshold

blind signature $\mathcal{TB}$, a (t^*, n^*)-threshold ElGamal decryption scheme $\mathcal{TE}$ with blockchain technology. The conceptual construction is outlined as follows.

For registration, t organizers co-operate to issue a threshold blind signature, using $\mathcal{TB}$, for the eligible voter. When an eligible voter casts his/her ballot, he/she encrypts his/her ballot with threshold ElGamal scheme $\mathcal{TE}$, whose encryption key is publicly available. The encrypted ballot, together with the signature obtained from the registration phase, is submitted to the blockchain. In the vote counting phase, co-operations of at least t^* tellers is needed to decrypt the encrypted ballots. The results are also published on the blockchain. As long as there are less than t (resp. t^*) malicious organizers (resp. tellers), the votes must be casted by legitimate users. Also, thanks to the blindness of $\mathcal{TB}$, no one will be able to link the ballot to a voter. In addition, since each user only receives one signature-random value pair, double ballots can therefore be detected and discarded. This ensure fairness. Looking ahead, the threshold techniques employed in $\mathcal{TB}$, $\mathcal{TE}$ are both one-round, meaning that the communication process can be simply conducted through the public blockchain efficiently.

In the following section, we clarify the design architecture of our system and then present the work process.

5.2.1 Architecture of Our System

We firstly identify the involved identities in our voting system, followed by proposing the architecture of our system.

Entities. There are three classes of entities involved in an e-voting system, namely, **Voters**, **Organizers** and **Tellers**.

- **Voters**. The voters have their right to vote. For simplicity, we assume that each eligible voter can only votes one ballot during voting.
- **Organizers**. Organizers are responsible for authorizing eligible voters. They can give voters voting rights through registration.
- **Tellers**. Tellers will count the ballots after voting and publish the final tallying result.

Architecture. Since our system is based on the blockchain, all participates communicate through the blockchain, as shown in the Figure 5.1. Besides, the role of organizers and tellers are played by some voters in the system. For example, the voter with identifier 1 is also a teller and the voter with identifier 10 is also responsible for organization. The roles can be selected randomly or based on some pre-determined discipline through the participants before running our system. Therefore, the voter that is also a teller should also run $\mathcal{TE}$ scheme

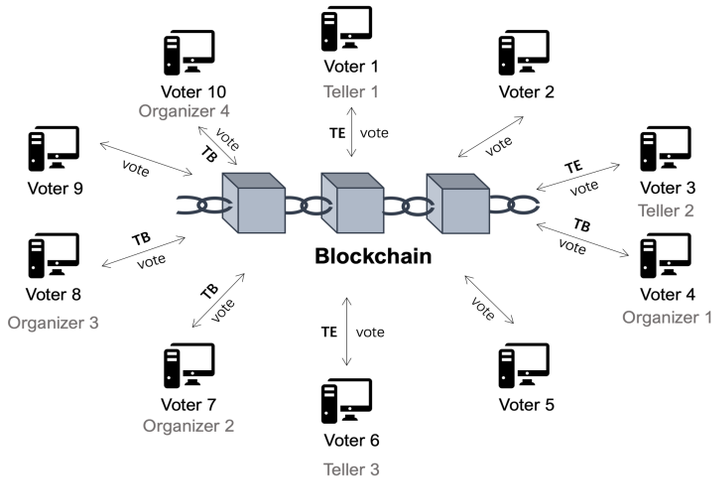

Figure 5.1: Architecture of our blockchain-based voting system

while the one with organization responsibility should run $\mathcal{TB}$ scheme, besides casting their ballots.

5.2.2 Work Process of Our System

Here we elaborate the process that how our system works. Before generating keys, we suppose that the system has selected n voters as organizations and n^* voters as tellers already. Then the voters will generate their private and public key pairs where their public keys are published through the blockchain. Besides, the organizers and tellers collaboratively generate the public and private key pairs separately, followed by publishing their public keys.

To get their voting rights, voters will register with organizations using $\mathcal{TB}$ scheme and getting a signature for their public keys. To cast their ballots securely, they will encrypt their ballots using the $\mathcal{TE}$ scheme. Besides, to guarantee that only eligible voters can vote, they should also attach their signature

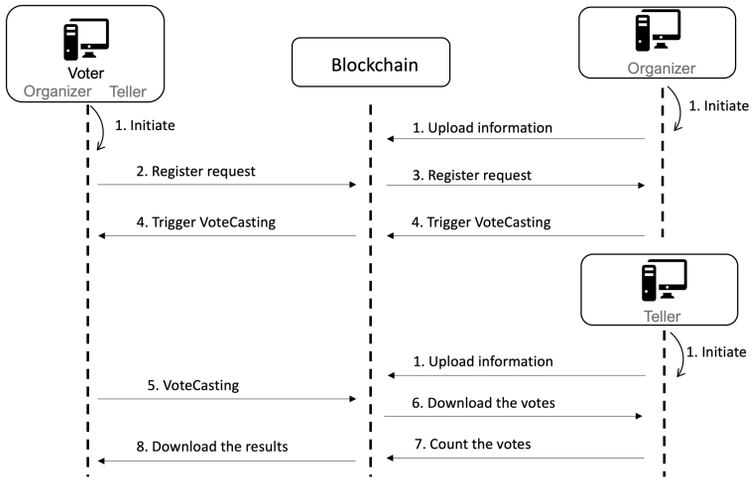

Figure 5.2: Voting Process Diagram of our blockchain-based voting system

on the encrypted ballots, together with the public-key and signature pair obtained from $\mathcal{TB}$ to publish on-chain.

To count the voting result, tellers should firstly verify the validity of signatures for encrypted ballots and for their public keys (checking their eligibility for voting), followed by collaboratively decrypting the ballots with l^* honest tellers and publishing the final result on-chain. Each voter can downloads the decrypted ballots from the blockchain and verify that whether his/her ballot has been counted or not. Besides, no body can link the voting ballot to the voter because of the use of blind signature and the anonymity guaranteed by the blockchain. We give the detailed illustration of this process in Figure 5.2.

5.3 Syntax and Security Requirements

In this section, we identify the syntax of our system and its security requirements. Besides, we give a simplified syntax of

our building block, Π^{sig}, a signature scheme in general.

Syntax

An e-voting system consists of the following four phases, namely, **Keygen**, **Register**, **VoteCasting** and **VoteCounting**.

1. **Keygen**. The voters, organizers and tellers generate their own public and secret key pairs respectively.
2. **Register**. In this phase, eligible voters get voting rights from organizers through registration.
3. **VoteCasting**. Each voter casts their ballot in the system.
4. **VoteCounting**. Tellers count the ballots and publish the final results.

Security Requirements

The security properties of an e-voting system are summarized as following four points.

- **Verifiability**. It means that individual voter can verify whether his/her ballot has been counted correctly or not.
- **Eligibility**. This property requires that only eligible voters are allowed to cast votes only once. Besides, only valid ballots will be counted.
- **Fairness**. This property requires that no early results can be obtained before the end of voting. It guarantees that

the choice of voters cannot be influenced by current voting results.

- **Anonymity**. The anonymity property requires that no one can reveal the owner of a ballot. In other words, voters can cast their ballots anonymously.

A Building Block Π^{sig}

A signature scheme consists of three algorithms, $(\mathsf{KeyG}, \mathsf{Sign}, \mathsf{Verify})$, as follows,

- **KeyG**: With input a security parameter λ, it returns the public and private key pair, (pk, sk). Besides, we denote its specified message and signature space as $\mathcal{M}$ and $\mathcal{S}$ respectively.
- **Sign**: To sign on a message $m \in \mathcal{M}$ with the secret key sk, it returns the corresponding signature σ from $\mathcal{S}$.
- **Verify**: For verification of a message-signature pair (m, σ), one should use the public key as, $\mathsf{Verify}(m, \sigma, pk) \rightarrow$ true/false.

5.4 Our Construction and Security Analysis

We adopt blockchain technology, a (t, n) threshold blind signature $\mathcal{TB}$ - (TBU, TBK, TBS, TBV), a (t^*, n^*) threshold ElGamal encryption scheme $\mathcal{TE}$- (TEU, TEK, TEC, TED) (which are defined in Chapter 2) and a signature scheme $\Pi^{sig} - \{\mathsf{KeyG}, \mathsf{Sign}, \mathsf{Verify}\}$ to construct our blockchain-aided e-voting system.

Entities and Their Notations

Three entities involved in our e-voting system is described as follows.

- **Voters.** Assume that there are l eligible voters, i.e., $V_i \in \{V_1, V_2, .., V_\ell\}$.
- **Organizers.** Organizers are played by n eligible voters, i.e., $L_i \in \{L_1, L_2, .., L_n\}$.
- **Tellers.** Tellers are played by n^* eligible voters, i.e., $T_i \in \{T_1, T_2, .., T_{n^*}\}$.

Detailed Construction

There are five phases in our voting system, namely, **Setup**, **KeyGen**, **Register**, **VoteCasting** and **VoteCounting**.

1. **Setup**: On input security parameter 1^λ, it outputs $\mathsf{param} = (\mathbf{G_1}, q, P, H)$ and broadcasts it. $\mathbf{G_1}$ is a GDH group with order q, P is its generator. $H : \{0,1\}^* \rightarrow \mathbf{G_1}$ is a one-way function. param is an implicit input to the following algorithms.
2. **KeyGen**: Each voter V_i first generates public key and secret key pair in Π^{sig}, i.e., $(pk_i, sk_i) \leftarrow \mathsf{KeyG}$. n players in $\mathcal{TB}$ will be randomly selected from all eligible voters, i.e., $\mathcal{L} = \{L_1, L_2, .., L_n\}$. n^* players in $\mathcal{TE}$ are randomly picked from all voters and we denote them as $\mathcal{T} = \{T_1, T_2, .., T_{n^*}\}$.

Finally, run TBK with players in $\mathcal{L}$ and TEK with players in $\mathcal{T}$ to get,

- Public and secret shares of L_i are set as, $Q_i = P^{s_i}$ and s_i. Q_i will be broadcasted on blockchain.
- The public key in $\mathcal{TB}$ is set as $Q = P^s$, with secret key s.
- Public and secret shares of T_i are set as, $Q_i^* = P^{s_i^*}$ and s_i^*. Q_i^* will be broadcasted on blockchain.
- The public key in $\mathcal{TE}$ is set as $Q^* = P^{s^*}$, with secret key s^*.

3. **Register**: Every voter V_i can get a blind signature on its public key pk_i, by interacting with t players L_i on TBS protocol, i.e., (σ_i, pk_i).

4. **VoteCasting**: Each voter V_i encrypts its ballot b by running TEC algortihm under the public key Q^*,

$$\mathsf{TEC}_{Q^*}(b) \rightarrow C_i.$$

V_i then uses its registered secret key sk_i to sign on the ciphertext, e.g., $\mathsf{Sign}_{sk_i}(C_i) \rightarrow \sigma_i'$, and puts the quadruple $(pk_i, \sigma_i, C_i, \sigma_i')$ on blockchain.

5. **VoteCounting**: When counting the ballots, tellers T_i first run TBV protocol to verify the signature σ_i on the voter's

public key pk_i, followed by validating signature σ_i' on ciphertext C_i. That is,

$$In\ \mathcal{TB}.\mathsf{TBV}:\ \hat{e}(\sigma_i, P) \stackrel{?}{=} \hat{e}(H(pk_i), Q);$$
$$In\ \Pi^{sig}.\mathsf{Verify}:\ \mathsf{Verify}(C_i, \sigma_i') \stackrel{?}{=} 1.$$

If all the verifications are passed, t^* tellers decrypt C_i together to get the ballot b. Ballot b will be added to the quadruple, i.e.,$(pk_i, \sigma_i, C_i, \sigma_i', b)$. The quadruple will then be broadcasted on blockchain.

5.4.1 Security Analysis

Our e-voting system can achieve four properties, namely, verifiability, eligibility, fairness and anonymity. We analyze them as follows.

- **Verifiability.** This property can be easily verified since the result will be put behind the quadruple as (pk_i, σ_i, C_i, σ_i', b). As the public key will be published on-chain, each user can verify that whether his or her ballot has been counted correctly.

- **Eligibility.** This property is guaranteed by the unforgeability property of the blind signature scheme and can be proved by contradiction. Specifically, if a malicious illegal adversary successfully forges a valid ballot in the form of (pk_i, σ_i, C_i, σ_i'), then one can forge a valid signature pair,

(pk_i, σ_i), in the underlying blind signature scheme. Besides, the other way of destroying the voting system's eligibility is to vote multiple times by using the same identity. This cannot be achieved because the quadruple (pk_i, σ_i, C_i, σ_i') reveals pk_i of the voter. Tellers can recognize it immediately if same voter casts ballots for multiple times.

- **Fairness.** This property can be guaranteed by the IND-CPA security in threshold ElGamal decryption scheme. As the ballots cast by voters are encrypted by using the ElGamal Decryption scheme, anyone cannot obtain the final result before the vote counting phase (i.e., decryption).

- **Anonymity.** This property is guaranteed by the blindness property provide in the threshold blind signature and anonymity of the blockchain. Specifically, the blindness property guarantees that the real identity of every voter remains unknown even to the organizers, since each of them gets signatures on their public keys blindly. Besides, the real characters of voters are hidden since we utilize blockchain to broadcast all our information. Therefore our system achieves highly anonymous.

Table 5.1: Total Time Consumed in Each Phase with $(7, 10)$ Threshold

Time Consumed (min) \ Number of Voters	1000	2000	3000	4000	5000
Register	0.3	0.6	0.9	1.22	1.49
VoteCasting	0.1	0.19	0.29	0.39	0.48
VoteCounting	0.18	0.37	0.55	0.74	0.91

Table 5.2: Average Time Consumed in Each Phase with $(7, 10)$ Threshold

Time Consumed (ms) \ Number of Voters	1000	2000	3000	4000	5000
Register	18.007	17.987	18.001	18.255	17.919
VoteCasting	5.829	5.835	5.838	5.874	5.792
VoteCounting	11.058	11.011	11.060	11.084	10.973

5.5 Implementation and Performance

5.5.1 Implementation Setup

We implement our voting system on a MacBook Pro with 3.1 GHz Intel Core i5 processor and 16 GB memory. We use PBC library [PBCa] and Crypto++ library [Cry] to implement our system. We choose the parameters suggested in Type A internals [PBCb]. Our implementation results are given by parallel computing to simulate the real situation.

5.5.2 Performance Evaluation

As *Setup* and *KeyGen* can be seen as the preparation process of voting, we evaluate **Register**, **VoteCasting** and **VoteCounting** phases in our system with holding $(7, 10)$-threshold. The results of our experiments are as follows.

The total time cost in each phase is linear to the number of voters in the system, as shown in Table 5.1. From Table 5.2, we can see that the time consumed in each phase for each voter is a

roughly constant value, with maintaining parameters in threshold. Besides, we also test our system by changing parameters (t, n) in threshold blind signature scheme and threshold ElGamal decryption scheme. The average time consumed in each stage has slight linear relation with parameter t, while the additional operation only costs little time while increasing t. The number in n (i.e., the total number of voters) does not have impact on the average consumed time in each phase.

In particular, when using a $(7, 10)$ threshold, it takes roughly 11-ms to count one vote. For 1 million votes, it takes about 3.06 hours to complete vote counting on a laptop. We stress that our final results are based on experiments over a laptop. When the system is deployed on a real server, the efficiency can be improved. Furthermore, the efficiency can be further optimized with moving the process of validation on signature ahead, in vote casting phase. Therefore, our whole system is efficient and practical enough to be adopted in the real world.

Chapter 6

Conclusions and Future Research Directions

Public-key cryptography has been extensively employed not only in our traditional daily lives but also in the recently introduced Web3 domain. With the potential of more complex interactions and security requirements in recent application scenarios, this thesis seeks to address some newly emerging challenges raised in the area of public-key cryptography. Specifically, contributions of this thesis can be summarized as follows.

- In Chapter 3, we concentrate on addressing the strong designated verifier signature schemes in more complex situations, by taking real-world interaction scenarios into account. Specifically, we propose two strengthened models in SDVS, namely multi-user and multi-user$^+$. We then provide a generic construction for these two security models, relying on Key Encapsulation Mechanism (KEM) and Pseudorandom Function (PRF) schemes. We formally prove the security of our constructions within these two models.

Furthermore, we offer instantiations based on different security assumptions in these models, respectively.

- We propose an efficient zero-knowledge argument of knowledge system customized for Paillier cryptosystem in Chapter 4. The foundation of our system is a constraint system defined over the ring of residue classes modulo a composite integer. Our constraint system is generic and can be utilized to express various typical relations in the Paillier system. Our proof system can support range proof, correctness proof, relations between bits of plaintexts, and more. Our argument system facilitates batch proof generation and verification. Additionally, we provide experimental results demonstrating that its amortized cost outperforms state-of-the-art proof systems specialized for Paillier when the number of Paillier ciphertexts reaches the order of hundreds.

- In Chapter 5, we construct a fully distributed e-voting scheme based on blockchain technology. We distribute registration and tallying authorities among multiple parties to prevent single node failure. Moreover, our system enjoys the feature of fast tallying with employing homomorphic encryption. We also conduct a series of experiments to examine its practicability.

Future Work. In addition to the problems addressed in this thesis, many intriguing questions remain to be explored. For instance, in our first work, we only consider the strengthened situation in which an adversary may obtain signatures for any verifier of its choice. We could further investigate the security model wherein the adversary may issue queries from any signer of its choice and determine how to construct a provably secure scheme towards this model.

Regarding the construction of ZKP systems for Paillier, the bottleneck lies in its proof generation cost. The primary challenge is that we must store a massive matrix to express variables in the constraint system and commit to them using Pedersen commitments. Although multi-exponentiation can accelerate this process, generating commitments can still be time-consuming when the matrix is large. A subsequent objective is to reduce time and memory usage in the proof phase to better accommodate large-scale application scenarios with limited computational power.

Furthermore, numerous interesting questions also arise in the field of blockchain. For example, as blockchain remains relatively isolated from the real world for security reasons, one cannot directly transfer real-world data to the blockchain. A promising approach involves using so-called oracles. However, the latency of existing oracles is on the order of seconds or

even to minutes. While this latency may be acceptable for relatively stable data, it is not satisfactory when data changes frequently. The method for exporting off-chain data to the blockchain with low latency without compromising its integrity and other security guarantees remains unknown. Furthermore, the cost of existing approaches for exporting data on-chain is quite high, typically requiring at least two transactions. How to reduce this cost without sacrificing security is another compelling goal for our future work.

Another promising direction of future work lies in how to accelerate the two-party data aggregation/ computation process by utilizing trusted a execution environment to meet the efficiency requirement in real-life deployment. On one hand, the security guarantee cannot be sacrificed by introducing trusted hardware; On the other hand, the system must be efficient enough to be deployed in reality.

References

[ABDV03] Riza Aditya, Colin Boyd, Ed Dawson, and Kapali Viswanathan. Secure e-voting for preferential elections. In Roland Traunmüller, editor, *Electronic Government*, pages 246–249, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[Abe98] Masayuki Abe. Universally verifiable mix-net with verification work independent of the number of mix-servers. *Lecture notes in computer science*, 1403:437–447, 1998.

[ABMR20] Shashank Agrawal, Saikrishna Badrinarayanan, Pratyay Mukherjee, and Peter Rindal. Game-set-match: Using mobile devices for seamless external-facing biometric matching. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS'20, pages 1351–1370, 2020.

[ACST06] Man Ho Au, Sherman SM Chow, Willy Susilo, and Patrick P Tsang. Short linkable ring signatures revisited. In *Public Key Infrastructure: Third European PKI Workshop: Theory and Practice, EuroPKI 2006,*

Turin, Italy, June 19-20, 2006. Proceedings 3, pages 101–115. Springer, 2006.

[Adi08] Ben Adida. Helios: Web-based open-audit voting. In *USENIX security symposium*, volume 17 of *USENIX'08*, pages 335–348, 2008.

[ads] Google adsense. `https://adsense.google.com`.

[adw] Google adwords. `https://ads.google.com/home/`.

[AGW20] Michel Abdalla, Junqing Gong, and Hoeteck Wee. Functional encryption for attribute-weighted sums from k-lin. In *Annual International Cryptology Conference*, Crypto' 20, pages 685–716. Springer, 2020.

[AHKL12] Ashton Anderson, Daniel Huttenlocher, Jon Kleinberg, and Jure Leskovec. Effects of user similarity in social media. In *Proceedings of the fifth ACM international conference on Web search and data mining*, 2012.

[ALSY06] Man Ho Au, Joseph K Liu, Willy Susilo, and Tsz Hon Yuen. Constant-size id-based linkable and revocable-iff-linked ring signature. In *Progress in Cryptology-INDOCRYPT 2006: 7th International Conference on Cryptology in India, Kolkata, India, December 11-13, 2006. Proceedings 7*, pages 364–378. Springer, 2006.

[Ara] Pradeep Aradhya. Distributed ledger visible to all? ready for blockchain? `https://www.huffpost.com/entry/are-we-ready-for-a-global_b_9591580`.

[AVS13] Maryam Rajabzadeh Asaar, Ali Vardasbi, and Mahmoud Salmasizadeh. Non-delegatable strong designated verifier signature using a trusted third party without pairings. In *Proceedings of Information Security 2013 (AISC 2013)*, pages 13–25. Australian Computer Society, 2013.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*, SP' 18, pages 315–334. IEEE, 2018.

[BCC+16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, pages 327–357, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[BCD+16] Joppe Bos, Craig Costello, Leo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko,

Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from lwe. In *SIGSAC*, pages 1006–1018. ACM, 2016.

[BCG+17] Jonathan Bootle, Andrea Cerulli, Essam Ghadafi, Jens Groth, Mohammad Hajiabadi, and Sune K Jakobsen. Linear-time zero-knowledge proofs for arithmetic circuit satisfiability. In *International Conference on the Theory and Application of Cryptology and Information Security*, AsiaCrypt' 17, pages 336–365. Springer, 2017.

[Ben87] Josh Daniel Cohen Benaloh. *Verifiable Secret-ballot Elections*. PhD thesis, New Haven, CT, USA, 1987. AAI8809191.

[BFL91] Joan Boyar, Katalin Friedl, and Carsten Lund. Practical zero-knowledge proofs: Giving hints and using deficiencies. *Journal of cryptology*, 4(3):185–206, 1991.

[BFP+01] Olivier Baudron, Pierre-Alain Fouque, David Pointcheval, Jacques Stern, and Guillaume Poupard. Practical multi-candidate election system. In *Proceedings of the twentieth annual ACM symposium on Principles of distributed computing (PODC)*, pages 274–283, 2001.

[BG02] Dan Boneh and Philippe Golle. Almost entirely correct mixing with applications to voting. In *Proceedings of the 9th ACM conference on Computer and communications security*, CCS'02, pages 68–77, 2002.

[BHKR] Mihir Bellare, Viet Tung Hoang, Sriram Keelveedhi, and Phillip Rogaway. Efficient garbling from a fixed-key blockcipher. In *2013 IEEE Symposium on Security and Privacy*.

[BIK+17] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 1175–1191, New York, NY, USA, 2017. Association for Computing Machinery.

[Bou00] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In Bart Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, Eurocrypt '00, pages 431–444, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen.

Pseudorandom functions and lattices. In *EUROCRYPT 2012*, pages 719–737. Springer, 2012.

[BS15] Raef Bassily and Adam Smith. Local, private, efficient protocols for succinct histograms. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, STOC'15, pages 127–135, 2015.

[BSCG+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. Snarks for c: Verifying program executions succinctly and in zero knowledge. In *Annual cryptology conference*, pages 90–108. Springer, 2013.

[BSLZ09] Joonsang Baek, Willy Susilo, Joseph K. Liu, and Jianying Zhou. A new variant of the cramer-shoup kem secure against chosen ciphertext attack. In *Applied Cryptography and Network Security*, pages 143–155. Springer, 2009.

[BSNS05] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Universal designated verifier signature proof (or how to efficiently prove knowledge of a signature). In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 644–661. Springer, 2005.

[BT94] Josh Benaloh and Dwight Tuinstra. Receipt-free

secret-ballot elections. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, STOC'94, pages 544–553, 1994.

[BY86] Josh C Benaloh and Moti Yung. Distributing the power of a government to enhance the privacy of voters. In *Proceedings of the fifth annual ACM symposium on Principles of distributed computing*, pages 52–62, 1986.

[CDN01] Ronald Cramer, Ivan Damgård, and Jesper B. Nielsen. Multiparty computation from threshold homomorphic encryption. In Birgit Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, Eurocrypt '01, pages 280–300, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo G. Desmedt, editor, *Advances in Cryptology — CRYPTO '94*, pages 174–187, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[CF85] Josh D Cohen and Michael J Fischer. *A robust and verifiable cryptographically secure election scheme*. FOCS'85. Yale University. Department of Computer Science, 1985.

[CFSY96] Ronald Cramer, Matthew Franklin, Berry Schoenmakers, and Moti Yung. Multi-authority secret-ballot elections with linear work. In *Eurocrypt*, volume 96 of *Eurocrypt'96*, pages 72–83. Springer, 1996.

[CGB17] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. In *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation*, NSDI'17, page 259–282, USA, 2017. USENIX Association.

[CGGI13] Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachene. Distributed elgamal á la pedersen: application to helios. In *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, pages 131–142, 2013.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. 8(5):481–490, 1997.

[Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, February 1981.

[Cha83] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology*, pages 199–203, Boston, MA, 1983. Springer US.

[CJZ+19] Jie Cai, Han Jiang, Pingyuan Zhang, Zhihua Zheng, Guangshi Lyu, and Qiuliang Xu. An efficient strong designated verifier signature based on r-sis assumption. *IEEE Access*, 7:3938–3947, 2019.

[CLW08] Sherman SM Chow, Joseph K Liu, and Duncan S Wong. Robust receipt-free election system with ballot secrecy and verifiability. In *NDSS*, volume 8 of *NDSS'08*, pages 81–94, 2008.

[CM99] Jan Camenisch and Markus Michels. Proving in zero-knowledge that a number is the product of two safe primes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 107–122. Springer, 1999.

[Cry] Crypto++ library. `https://www.cryptopp.com/`.

[CSY06] Sherman SM Chow, Willy Susilo, and Tsz Hon Yuen. Escrowed linkability of ring signatures and its applications. In *Progress in Cryptology-VIETCRYPT 2006: First International Conference on Cryptology in Vietnam, Hanoi, Vietnam, September*

25-28, 2006. Revised Selected Papers, pages 175–192. Springer, 2006.

[CVA90] David Chaum and Hans Van Antwerpen. Undeniable signatures. In Gilles Brassard, editor, *CRYPTO' 89 Proceedings*, pages 212–216. Springer New York, 1990.

[CYL+20] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. Explicit filterbank learning for neural image style transfer and image processing. *IEEE transactions on pattern analysis and machine intelligence*, 43(7):2373–2387, 2020.

[DF90] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 307–315, New York, NY, 1990. Springer New York.

[DGB87] Yvo Desmedt, Claude Goutier, and Samy Bengio. Special uses and abuses of the fiat-shamir passport protocol (extended abstract). In *CRYPTO '87*, pages 21–39. Springer, 1987.

[DGN+] Nico Dottling, Satrajit Ghosh, Jesper Buus Nielsen, Tobias Nilges, and Roberto Trifiletti. Tinyole: Efficient actively secure two-party computation from oblivious linear function evaluation. CCS'17.

[Dif76] Whitfield Diffie. New direction in cryptography. *IEEE Trans. Inform. Theory*, 22:472–492, 1976.

[DJ01a] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In Kwangjo Kim, editor, *Public Key Cryptography*, pages 119–136, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[DJ01b] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *Public Key Cryptography: 4th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2001 Cheju Island, Korea, February 13–15, 2001 Proceedings 4*, PKC'01, pages 119–136. Springer, 2001.

[DKNS04] Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous identification in ad hoc groups. In Christian Cachin and Jan L. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, pages 609–626, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[DM19] Raul Diaz and Amit Marathe. Soft labels for ordinal regression. In *Proceedings of the IEEE/CVF*

conference on computer vision and pattern recognition, CVPR'19, pages 4738–4747, 2019.

[DY91] Yvo Desmedt and Moti Yung. Weaknesses of undeniable signature schemes. In *EUROCRYPT '91*, pages 205–220. Springer, 1991.

[EDG14] Tariq Elahi, George Danezis, and Ian Goldberg. Privex: Private collection of traffic statistics for anonymous communication networks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS' 14, pages 1068–1079, 2014.

[elea] 2021 election committee subsector ordinary elections. `https://www.elections.gov.hk/ecss2021/eng/brief.html`.

[eleb] Election for swiss council of states 2019. `https://www.electionguide.org/elections/id/3448/`.

[elec] Election news of federative republic of brazil. `https://www.electionguide.org/countries/id/31/`.

[ET12] Zekeriya Erkin and Gene Tsudik. Private computation of spatial and temporal power consumption

with smart meters. In Feng Bao, Pierangela Samarati, and Jianying Zhou, editors, *Applied Cryptography and Network Security*, pages 561–577, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[FIPR05] Michael J Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In *Theory of Cryptography Conference*, pages 303–324. Springer, 2005.

[FMM+03] Jun Furukawa, Hiroshi Miyauchi, Kengo Mori, Satoshi Obana, and Kazue Sako. An implementation of a universally verifiable electronic voting scheme based on shuffling. In Matt Blaze, editor, *Financial Cryptography*, pages 16–30, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[FMS10] Jun Furukawa, Kengo Mori, and Kazue Sako. *An Implementation of a Mix-Net Based Network Voting Scheme and Its Use in a Private Organization*, pages 141–154. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[FO97] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Annual International Cryptology Conference*, Crypto' 97, pages 16–30. Springer, 1997.

[FOO93] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In Jennifer Seberry and Yuliang Zheng, editors, *Advances in Cryptology — AUSCRYPT '92*, pages 244–251, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[FPS01] Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern. Sharing decryption in the context of voting or lotteries. In *Financial Cryptography: 4th International Conference, FC 2000 Anguilla, British West Indies, February 20–24, 2000 Proceedings 4*, FC'01, pages 90–104. Springer, 2001.

[FS01] Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In *Advances in Cryptology—CRYPTO 2001: 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19–23, 2001 Proceedings 21*, Crypto'01, pages 368–387. Springer, 2001.

[GH19] Craig Gentry and Shai Halevi. Compressible fhe with applications to pir. In *Theory of Cryptography Conference*, pages 438–464. Springer, 2019.

[GHH+21] Biyang Guo, Songqiao Han, Xiao Han, Hailiang Huang, and Ting Lu. Label confusion learning to

enhance text classification models. In *Proceedings of the AAAI conference on artificial intelligence*, 2021.

[GI08] Jens Groth and Yuval Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 379–396. Springer, 2008.

[GMO16] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. {ZKBoo}: Faster {Zero-Knowledge} for boolean circuits. In *25th USENIX Security Symposium (USENIX Security 16)*, USENIX' 16, pages 1069–1083, 2016.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems. In *STOC '85*, pages 291–304, 1985.

[GP87] Jeroen van de Graaf and Renė Peralta. A simple and secure way to show the validity of your public key. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 128–134. Springer, 1987.

[Gro09] Jens Groth. Linear algebra with sub-linear zero-knowledge arguments. In *Annual International Cryptology Conference*, Crypto' 09, pages 192–208. Springer, 2009.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 305–326. Springer, 2016.

[GSM+20] Rajesh Gupta, Arpit Shukla, Parimal Mehta, Pronaya Bhattacharya, Sudeep Tanwar, Sudhanshu Tyagi, and Neeraj Kumar. VAHAK: A blockchain-based outdoor delivery scheme using UAV for healthcare 4.0 services. In *39th IEEE Conference on Computer Communications, INFOCOM Workshops 2020, Toronto, ON, Canada, July 6-9, 2020*, INFOCOM Workshop'20, pages 255–260. IEEE, 2020.

[GZ07] Tingjian Ge and Stanley B. Zdonik. Answering aggregation queries in a secure system model. In Christoph Koch, Johannes Gehrke, Minos N. Garofalakis, Divesh Srivastava, Karl Aberer, Anand Deshpande, Daniela Florescu, Chee Yong Chan, Venkatesh Ganti, Carl-Christian Kanne, Wolfgang Klas, and Erich J. Neuhold, editors, *Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007*, pages 519–530. ACM, 2007.

[GZB+02] Philippe Golle, Sheng Zhong, Dan Boneh, Markus Jakobsson, and Ari Juels. Optimistic mixing for exit-polls. In *Advances in Cryptology—ASIACRYPT*

2002: 8th International Conference on the Theory and Application of Cryptology and Information Security Queenstown, New Zealand, December 1–5, 2002 Proceedings 8, Asiacrypt'02, pages 451–465. Springer, 2002.

[Her] Alyssa Hertig. The first bitcoin voting machine is on its way. `http://motherboard.vice.com/read/the-first-bitcoin-voting-machine-ison-its-way`.

[HHL+15] Shuquan Hou, Xinyi Huang, Joseph K. Liu, Jin Li, and Li Xu. Universal designated verifier transitive signatures for graph-based big data. *Information Sciences*, 318(10):144–156, 2015.

[HK21] David Heath and Vladimir Kolesnikov. One hot garbling. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 574–593, 2021.

[HKLD17] Rolf Haenni, Reto E Koenig, Philipp Locher, and Eric Dubuis. Chvote system specification. *IACR Cryptol. ePrint Arch.*, 2017:325, 2017.

[HKR19] Max Hoffmann, Michael Klooß, and Andy Rupp. Efficient zero-knowledge arguments in the discrete log setting, revisited. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2093–2110, 2019.

[HL09] Carmit Hazay and Yehuda Lindell. Efficient oblivious polynomial evaluation with simulation-based security. *Cryptology ePrint Archive*, 2009.

[HMR+19] Carmit Hazay, Gert LæssØe Mikkelsen, Tal Rabin, Tomas Toft, and Angelo Agatino Nicolosi. Efficient rsa key generation and threshold paillier in the two-party setting. *Journal of Cryptology*, 32(2):265–323, apr 2019.

[Hom19] Homomorphic standard. iso/textbakslash 18033-6:2019, 2019.

[HS00] Martin Hirt and Kazue Sako. Efficient receipt-free voting based on homomorphic encryption. In *Advances in Cryptology—EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques Bruges, Belgium, May 14–18, 2000 Proceedings*, Eurocrypt'00, pages 539–556. Springer, 2000.

[HSMW06] Xinyi Huang, Willy Susilo, Yi Mu, and Wei Wu. Universal designated verifier signature without delegatability. In *International Conference on Information and Communications Security*, pages 479–498. Springer, 2006.

[HSMZ06] Xinyi Huang, Willy Susilo, Yi Mu, and Futai Zhang. Short (identity-based) strong designated

verifier signature schemes. In *ISPEC 2006*, pages 214–225. Springer, 2006.

[HSMZ08] Xinyi Huang, Willy Susilo, Yi Mu, and Futai Zhang. Short designated verifier signature scheme and its identity-based variant. *International Journal of Network Security*, 6(1):82–93, 2008.

[HSW09] Qiong Huang, Willy Susilo, and Duncan S. Wong. Non-delegatable identity-based designated verifier signature. Cryptology ePrint Archive, Report 2009/367, 2009.

[HYK10] Hannaneh Hajishirzi, Wen-tau Yih, and Aleksander Kolcz. Adaptive near-duplicate detection via similarity learning. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 419–426, 2010.

[HYWS11] Qiong Huang, Guomin Yang, Duncan S. Wong, and Willy Susilo. Efficient strong designated verifier signature schemes without random oracle or with non-delegatability. *International Journal of Information Security*, 10(6):373, Aug 2011.

[JJR02] Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *Proceedings of the*

11th USENIX Security Symposium, pages 339–353, Berkeley, CA, USA, 2002. USENIX Association.

[JK12] Marek Jawurek and Florian Kerschbaum. Fault-tolerant privacy-preserving statistics. In Simone Fischer-Hübner and Matthew Wright, editors, *Privacy Enhancing Technologies*, PETS '12, pages 221–238, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[JKO13] Marek Jawurek, Florian Kerschbaum, and Claudio Orlandi. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 955–966, 2013.

[JL13] Marc Joye and Benoît Libert. A scalable scheme for privacy-preserving aggregation of time-series data. In *International Conference on Financial Cryptography and Data Security*, FC' 13, pages 111–125. Springer, 2013.

[JLL02] Wen-Sheng Juang, Chin-Laung Lei, and Horng-Twu Liaw. A verifiable multi-authority secret election allowing abstention from voting. In *The Computer Journal 2002*, 45, pages 672 – 682. IEEE, 2002.

[JSI96a] Markus Jakobsson, Kazue Sako, and Russell Impagiazzo. Designated verifier proofs and their applications. In *EUROCRYPT 1996*, pages 143–154. Springer, 1996.

[JSI96b] Markus Jakobsson, Kazue Sako, and Russell Impagiazzo. Designated verifier proofs and their applications. In *EUROCRYPT 1996*, pages 143–154. Springer, 1996.

[KV16] Oksana Kulyk and Melanie Volkamer. Efficiency comparison of various approaches in e-voting protocols. In *Financial Cryptography and Data Security: FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers 20*, FC' 16, pages 209–223. Springer, 2016.

[KY02] Aggelos Kiayias and Moti Yung. Self-tallying elections and perfect ballot secrecy. In David Naccache and Pascal Paillier, editors, *Public Key Cryptography*, pages 141–158, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[KY04] Aggelos Kiayias and Moti Yung. The vector-ballot e-voting approach. In *International Conference on Financial Cryptography*, FC '04, pages 72–89. Springer, 2004.

[LFY+17] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Diversified texture synthesis with feed-forward networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[Lin03] Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. *Journal of Cryptology*, 16(3), 2003.

[Lin08] Andrew Y Lindell. Efficient fully-simulatable oblivious transfer. In *Cryptographers' Track at the RSA Conference*, pages 52–70. Springer, 2008.

[Lin17] Yehuda Lindell. Fast secure two-party ecdsa signing. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, Crypto '17, pages 613–644, Cham, 2017. Springer International Publishing.

[LK03] Byoungcheon Lee and Kwangjo Kim. Receipt-free electronic voting scheme with a tamper-resistant randomizer. In Pil Joong Lee and Chae Hoon Lim, editors, *Information Security and Cryptology — ICISC 2002*, pages 389–406, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[LLL+20] Jin Li, Xuguang Lan, Yang Long, Yang Liu, Xingyu

Chen, Ling Shao, and Nanning Zheng. A joint label space for generalized zero-shot classification. *IEEE Transactions on Image Processing*, 29:5817–5831, 2020.

[LN18] Yehuda Lindell and Ariel Nof. Fast secure multiparty ecdsa with practical distributed key generation and applications to cryptocurrency custody. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS' 18, pages 1837–1854, 2018.

[LTK+22] Naivedya Lath, Kaustubh Thapliyal, Kartik Kandpal, Mohammad Wazid, Ashok Kumar Das, and D. P. Singh. Bdesf-its: Blockchain-based secure data exchange and storage framework for intelligent transportation system. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, INFOCOM Workshop'22, pages 1–6, 2022.

[LV04] Fabien Laguillaumie and Damien Vergnaud. Designated verifier signatures: Anonymity and efficient construction from any bilinear map. In *Security in Communication Networks*, pages 105–119. Springer, 2004.

[LV05a] Fabien Laguillaumie and Damien Vergnaud. Designated verifier signatures: Anonymity and efficient construction from any bilinear map. In *International Conference on Security in Communication Networks*, pages 105–119. Springer, 2005.

[LV05b] Fabien Laguillaumie and Damien Vergnaud. Designated verifier signatures: Anonymity and efficient construction from any bilinear map. In *Security in Communication Networks*, pages 105–119. Springer, 2005.

[LVKF16] Sheng Li, Nikos Vlassis, Jaya Kawale, and Yun Fu. Matching via dimensionality reduction for estimation of treatment effects in digital marketing campaigns. In *IJCAI*, pages 3768–3774, 2016.

[LWC12] Rui Li, Shengjie Wang, and Kevin Chen-Chuan Chang. Multiple location profiling for users and relationships from social network and content. *arXiv preprint arXiv:1208.0288*, 2012.

[LWW04a] Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups. In Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan, editors, *Information Security and Privacy*, pages 325–335, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[LWW04b] Joseph K Liu, Victor K Wei, and Duncan S Wong. Linkable spontaneous anonymous group signature for ad hoc groups. In *ACISP*, volume 4, pages 325–335. Springer, 2004.

[LWY12] Han-Yu Lin, Tzong-Sun Wu, and Yi-Shiung Yeh. A dl based short strong designated verifier signature scheme with low computation. *Journal of Information Science and Engineering*, 27(2):451–463, 2012.

[LXZ21] Tianyi Liu, Xiang Xie, and Yupeng Zhang. Zkcnn: Zero knowledge proofs for convolutional neural network predictions and accuracy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS' 21, pages 2968–2985, 2021.

[LZF+20] Bingchen Liu, Yizhe Zhu, Zuohui Fu, Gerard De Melo, and Ahmed Elgammal. Oogan: Disentangling gan with one-hot sampling and orthogonal regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[M.05] Avanzi Roberto M. The complexity of certain multi-exponentiationtechniques in cryptography. *Journal of Cryptology*, 18(4):357–373, Sep 2005.

[Mas99] ABE Masayuki. Mix-networks on permutation networks. In *ASIACRYPT'99*, page 258. Springer, 1999.

[MBKM19] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge snarks from linear-size universal and updatable structured reference strings. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2111–2128, 2019.

[MDDC15] Luca Melis, George Danezis, and Emiliano De Cristofaro. Efficient private statistics with succinct sketches. *arXiv preprint arXiv:1508.06110*, 2015.

[MO18] Daniel Minoli and Benedict Occhiogrosso. Blockchain mechanisms for iot security. *Internet of Things*, 1:1–13, 2018.

[MR+89] S Micali, Charles Rackoff, et al. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[MSH17] Patrick McCorry, Siamak F. Shahandashti, and Feng Hao. A smart contract for boardroom voting with maximum voter privacy. In *Financial Cryptography*, 2017.

[MSV13] V. Mateu, F. Sebé, and M. Valls. Blind certificates for secure electronic voting. In *2013 10th International Conference on Information Technology: New Generations*, pages 20–26, April 2013.

[MZO+99] Masahiro Mambo, Yuliang Zheng, Miyako Ohkubo, Fumiaki Miura, Masayuki Abe, Atsushi Fujioka, and Tatsuaki Okamoto. An improvement on a practical secret voting scheme. In *Information Security: Second InternationalWorkshop, ISW'99 Kuala Lumpur, Malaysia, November 6-7, 1999 Proceedings 2*, pages 225–234. Springer, 1999.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[Nef01] C Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, CCS'01, pages 116–125, 2001.

[NJ16] Geotae Noh and Ik Rae Jeong. Strong designated verifier signature scheme from lattices in the standard model. *Security and Communication Networks*, 9(18):6202–6214, 2016.

[nom] Alberta senate election act. `https://www.alberta.ca/alberta-senate-election-act.aspx`.

[NR04] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–262, March 2004.

[NWI+13] Valeria Nikolaenko, Udi Weinsberg, Stratis Ioannidis, Marc Joye, Dan Boneh, and Nina Taft. Privacy-preserving ridge regression on hundreds of millions of records. In *2013 IEEE symposium on security and privacy*, pages 334–348. IEEE, 2013.

[Oka98] Tatsuaki Okamoto. Receipt-free electronic voting schemes for large scale elections. In Bruce Christianson, Bruno Crispo, Mark Lomas, and Michael Roe, editors, *Security Protocols*, pages 25–35, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[OKST97] Wakaha Ogata, Kaoru Kurosawa, Kazue Sako, and Kazunori Takatani. Fault tolerant anonymous channel. In *Information and Communications Security: First International Conference, ICIS'97 Beijing, China, November 11–14, 1997 Proceedings 1*, ICICS'97, pages 440–444. Springer, 1997.

[Ord03] Carlos Ordonez. Clustering binary data streams with k-means. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 12–19, 2003.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, pages 223–238, Berlin, Heidelberg,

1999. Springer Berlin Heidelberg.

[PBBL11] Raluca Ada Popa, Andrew J. Blumberg, Hari Balakrishnan, and Frank H. Li. Privacy and accountability for location-based aggregate statistics. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, page 653–666, New York, NY, USA, 2011. Association for Computing Machinery.

[PBCa] Pbc library. `https://crypto.stanford.edu/pbc/manual/`.

[PBCb] Pbc library - type a internals. `https://crypto.stanford.edu/pbc/manual/ch08s03.html`.

[Ped91] Torben Pryds Pedersen. A threshold cryptosystem without a trusted party. In *Proceedings of the 10th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'91, pages 522–526, Berlin, Heidelberg, 1991. Springer-Verlag.

[Pfi95] Birgit Pfitzmann. Breaking an efficient anonymous channel. In *Advances in Cryptology—EUROCRYPT'94: Workshop on the Theory and Application of Cryptographic Techniques Perugia, Italy, May 9–12, 1994 Proceedings 13*, Eurocrypt'94, pages 332–340. Springer, 1995.

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, SP' 13, pages 238–252. IEEE, 2013.

[PIK93] Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *Advances in Cryptology—EUROCRYPT'93: Workshop on the Theory and Application of Cryptographic Techniques Lofthus, Norway, May 23–27, 1993 Proceedings 12*, EUROCRYPT' 93, pages 248–259. Springer, 1993.

[PP90] Birgit Pfitzmann and Andreas Pfitzmann. How to break the direct rsa-implementation of mixes. In *Advances in Cryptology—EUROCRYPT'89: Workshop on the Theory and Application of Cryptographic Techniques Houthalen, Belgium, April 10–13, 1989 Proceedings 8*, Eurocrypt'89, pages 373–381. Springer, 1990.

[RAD78] R L Rivest, L Adleman, and M L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.

[ran18] Range proof for paillier. `https://github.com/ZenGo-X/zk-paillier/blob/master/src/`

`zkproofs/range_proof_ni.rs`, 2018.

[RN10] Vibhor Rastogi and Suman Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, page 735–746, New York, NY, USA, 2010. Association for Computing Machinery.

[RR16] Peter Rindal and Mike Rosulek. Faster malicious 2-party secure computation with online/offline dual execution. In *25th USENIX Security Symposium*, 2016.

[RST01] Ronald L Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *Advances in Cryptology—ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings 7*, Asiacrypt'01, pages 552–565. Springer, 2001.

[Rya08] P. Y. A. Ryan. Prêt à voter with paillier encryption. *Math. Comput. Model.*, 48(9-10):1646–1662, November 2008.

[SBWP03] Ron Steinfeld, Laurence Bull, Huaxiong Wang, and

Josef Pieprzyk. Universal designated-verifier signatures. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 523–542. Springer, 2003.

[SCR+11] Elaine Shi, T.-H. Hubert Chan, Eleanor Gilbert Rieffel, Richard Chow, and Dawn Xiaodong Song. Privacy-preserving aggregation of time-series data. In *NDSS*, 2011.

[Sho96] Victor Shoup. Ntl: a library for doing number theory. 1996.

[SJZ+19] Asad Ali Siyal, Aisha Zahid Junejo, Muhammad Zawish, Kainat Ahmed, Aiman Khalil, and Georgia Soursou. Applications of blockchain technology in medicine and healthcare: Challenges and future perspectives. *Cryptography*, 3(1):3, 2019.

[SK95] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *Advances in Cryptology — EUROCRYPT '95*, pages 393–403, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[SKM03a] Shahrokh Saeednia, Steve Kremer, and Olivier Markowitch. An efficient strong designated verifier signature scheme. In *ICISC 2003*, pages 40–54. Springer, 2003.

[SKM03b] Shahrokh Saeednia, Steve Kremer, and Olivier Markowitch. An efficient strong designated verifier signature scheme. In *ICISC 2003*, pages 40–54. Springer, 2003.

[SLL+22] Yang Shi, Junqing Liang, Mianhong Li, Tianchen Ma, Guodong Ye, Jiangfeng Li, and Qinpei Zhao. Threshold eddsa signature for blockchain-based decentralized finance applications. In *Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses*, pages 129–142, 2022.

[SMPP10] Francesc Sebé, Josep M. Miret, Jordi Pujolàs, and Jordi Puiggalí. Simple and efficient hash-based verifiable mixing for remote electronic voting. *Computer Communications*, 33:667–675, 04 2010.

[SSN08] Siamak F Shahandashti and Reihaneh Safavi-Naini. Construction of universal designated-verifier signatures and identity-based signatures from standard signatures. In *International Workshop on Public Key Cryptography*, pages 121–140. Springer, 2008.

[SW21] Elaine Shi and Ke Wu. Non-interactive anonymous router. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 489–520. Springer, 2021.

[SWP04] Ron Steinfeld, Huaxiong Wang, and Josef Pieprzyk. Efficient extension of standard schnorr/rsa signatures into universal designated-verifier signatures. In *International Workshop on Public Key Cryptography*, pages 86–100. Springer, 2004.

[SZM04] Willy Susilo, Fangguo Zhang, and Yi Mu. Identy-based strong designated verifier signature schemes. In *ACISP 2004*, pages 313–324. Springer, 2004.

[Tak20] Yuta Takanashi. Future of finance. In Matthew Bernhard, Andrea Bracciali, L. Jean Camp, Shin'ichiro Matsuo, Alana Maurushat, Peter B. Rønne, and Massimiliano Sala, editors, *Financial Cryptography and Data Security*, pages 242–253, Cham, 2020. Springer International Publishing.

[TCZ+12] Haibo Tian, Xiaofeng Chen, Fangguo Zhang, Baodian Wei, Zhengtao Jiang, and Yi Liu. An efficient identity-based strong designated verifier signature without delegatability. In *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 81–88, 2012.

[TL14] Haibo Tian and Jin Li. A short non-delegatable strong designated verifier signature. *Frontiers of*

Computer Science, 8(3):490–502, 2014.

[TNB+10] Vincent Toubiana, Arvind Narayanan, Dan Boneh, Helen Nissenbaum, and Solon Barocas. Adnostic: Privacy preserving targeted advertising. In *Proceedings Network and Distributed System Symposium*, 2010.

[TOO05] Raylin Tso, Takeshi Okamoto, and Eiji Okamoto. Practical strong designated verifier signature schemes based on double discrete logarithms. In *Information Security and Cryptology*, pages 113–127. Springer, 2005.

[TW05] Patrick P. Tsang and Victor K. Wei. Short linkable ring signatures for e-voting, e-cash and attestation. In Robert H. Deng, Feng Bao, HweeHwa Pang, and Jianying Zhou, editors, *Information Security Practice and Experience*, pages 48–60, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[TWC+05] Patrick P Tsang, Victor K Wei, Tony K Chan, Man Ho Au, Joseph K Liu, and Duncan S Wong. Separable linkable threshold ring signatures. In *Progress in Cryptology-INDOCRYPT 2004: 5th International Conference on Cryptology in India, Chennai, India, December 20-22, 2004. Proceedings 5*, pages 384–398. Springer, 2005.

[VZK02] Duc Vo, Fangguo Zhang, and Kwangjo Kim. A new threshold blind signature scheme from pairings. 2002.

[WRF+] Chi Wang, Rajat Raina, David Fong, Ding Zhou, Jiawei Han, and Greg Badros. Learning relevance from heterogeneous social network and its application in online targeting. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*.

[WRK] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Authenticated garbling and efficient maliciously secure two-party computation. CCS'17.

[WZZ+22] Chen Wang, Daosen Zhai, Ruonan Zhang, Huan Li, Haotong Cao, and Anish Jindal. Joint uavs position optimization and offloading decision for blockchain-enabled intelligent transportation. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops, INFOCOM 2022 - Workshops, New York, NY, USA, May 2-5, 2022*, INFOCOM Workshop'22, pages 1–6. IEEE, 2022.

[XSHT08] Zhe Xia, Steve A. Schneider, James Heather, and Jacques Traoré. Analysis, improvement and simplification of prêt à voter with paillier encryption. In *Proceedings of the Conference on Electronic Voting*

Technology, EVT'08, pages 13:1–13:15, Berkeley, CA, USA, 2008. USENIX Association.

[YLS+18] Bin Yu, Joseph K. Liu, Amin Sakzad, Surya Nepal, Ron Steinfeld, Paul Rimba, and Man Ho Au. Platform-independent secure blockchain-based voting system. In Liqun Chen, Mark Manulis, and Steve Schneider, editors, *Information Security*, pages 369–386, Cham, 2018. Springer International Publishing.

[YLW+] Jun Yan, Ning Liu, Gang Wang, Wen Zhang, Yun Jiang, and Zheng Chen. How much can behavioral targeting help online advertising? WWW.

[Zca] Zcash. `https://z.cash/`.

[ZFI05] Rui Zhang, Jun Furukawa, and Hideki Imai. Short signature and universal designated verifier signature without random oracles. In *International Conference on Applied Cryptography and Network Security*, pages 483–498. Springer, 2005.

[ZH08] Mi Zhang and Neil Hurley. Avoiding monotony: improving the diversity of recommendation lists. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 123–130, 2008.

[ZLX+20] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. {BatchCrypt}: Efficient

homomorphic encryption for {Cross-Silo} federated learning. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, pages 493–506, 2020.

[ZRE] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole: Reducing data transfer in garbled circuits using half gates. In *Advances in Cryptology-EUROCRYPT 2015*.

[ZSMC05] Fangguo Zhang, Willy Susilo, Yi Mu, and Xiaofeng Chen. Identity-based universal designated verifier signatures. In *International Conference on Embedded and Ubiquitous Computing*, pages 825–834. Springer, 2005.

[ZWC+22] M. Zhou, T. Wang, T. Chan, G. Fanti, and E. Shi. Locally differentially private sparse vector aggregation. In *2022 2022 IEEE Symposium on Security and Privacy (SP) (SP)*, pages 1565–1565. IEEE Computer Society, 2022.

[ZWDY21] Pingyue Zhang, Mengyue Wu, Heinrich Dinkel, and Kai Yu. Depa: Self-supervised audio embedding for depression detection. In *Proceedings of the 29th ACM international conference on multimedia*, pages 135–143, 2021.

[ZXZS20] Jiaheng Zhang, Tiancheng Xie, Yupeng Zhang, and Dawn Song. Transparent polynomial delegation

and its applications to zero knowledge proof. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 859–876. IEEE, 2020.

[ZYFZ19] Jiang Zhang, Yu Yu, Shuqin Fan, and Zhenfeng Zhang. Improved lattice-based cca2-secure pke in the standard model. Cryptology ePrint Archive, Report 2019/149, 2019. `https://eprint.iacr.org/2019/149`.