博士論文

Privacy-Preserving Blockchain with Homomorphic Encryption

Tatsuo MITANI
三谷 辰雄

情報セキュリティ大学院大学
情報セキュリティ研究科
情報セキュリティ専攻

2020年9月

Privacy-Preserving Blockchain with Homomorphic Encryption

Tatsuo Mitani

Submitted in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy in Informatics
in the Graduate School of Information Security

INSTITUTE OF INFORMATION SECURITY, JAPAN
September 2020

Publication list

This thesis is based on the below publications.

Peer Reviewed Papers

[**MO20c**] T. Mitani and A. Otsuka, "Traceability in Permissioned Blockchain," in IEEE Access, vol. 8, pp. 21573-21588, 2020. doi: 10.1109/ACCESS.2020.2969454

Peer Reviewed Proceedings of International Conferences

[**MO19**] T. Mitani and A. Otsuka, "Traceability in Permissioned Blockchain," in 2019 IEEE International Conference on Blockchain, Atlanta, GA, USA, 2019, pp. 286-293. doi: 10.1109/Blockchain.2019.00045

[**MO20b**] T. Mitani and A. Otsuka, "Confidential and Auditable Payments," in Financial Cryptography and Data Security. FC 2020. Lecture Notes in Computer Science, vol 12063, pp 466-480. Springer, Cham. doi: 10.1007/978-3-030-54455-3_33

Preprint

[**MO20a**] T. Mitani and A. Otsuka, "Anonymous probabilistic payment in payment hub," Cryptology ePrint Archive, Report 2020/748, 2020. `https://eprint.iacr.org/2020/748`

Abstract

Our modern society is highly digitized. Blockchain is an underlying technology that constitutes society. Privacy protection is a significant theme in society. For privacy protection, homomorphic encryption is a promising technology capable of computing the state of encrypted data. In this thesis, we study privacy-preserving blockchain with homomorphic encryption. There are two challenges in achieving privacy-preserving blockchain. One is confidentiality in the transaction content. The other is unlinkability in the user identity of the transaction. We have achieved both in the permissioned blockchain. Furthermore, we realized confidentiality and unlinkability in the permissionless blockchain.

Traceability in permissioned blockchain. We have achieved privacy protection and high transparency in a permissioned blockchain. Suppose that there is a sidechain that connects the permissionless blockchain and the permissioned blockchain. The behavior in the permissioned blockchain is almost a black box from the perspective of the permissionless blockchain. We present a novel concept of traceability consisting of three properties. The properties are as follows. First, trade privacy says that who trades with whom and at what asset amount. Second, preservation says that the total amount inside the permissioned blockchain is immutable. We take inflow and outflow the permissionless blockchain into consideration. Finally, noninvolvement says that some members in the permissioned blockchain are not involved in some trades. One can prove that specified members performed the transaction. Our approach is as follows. We model traceability with the hidden Markov model. The proof of traceability requires the calculation of more than quadratic degrees. Therefore we encrypt this model by using homomorphic encryption. The establishment of the original model is verifiable by the zero-knowledge proof.

Confidential and auditable payments. We construct the confidential and auditable payments scheme. This proposal eliminates concerns about money laundering caused by excessively confidential transactions and contributes to blockchain's sound use. Our approach is as follows. For confidentiality, we keep the transaction confidential by writing ciphertexts of transactions in a ledger. The soundness of the zero-knowledge proof realizes the soundness of the scheme. For auditability, a court or an authority controls a unique secret key of the ledger's ciphertexts. They can enforce confidential transactions open with the secret key according to the proper procedure.

Anonymous probabilistic payment in payment hub. Privacy protection and scalability are significant problems with blockchain. We propose an anonymous probabilistic payment under the general functionality for solving the problems. Suppose that a payer pays a payee through a tumbler. We realize the anonymity, which says that the link, which payer pays which payee via the tumbler within an epoch, is broken. The epoch is the period when one completes transactions. Our proposal includes a probabilistic payment. In the probabilistic payment, one pays an ordinary mount m with a certain probability p, and one pays a small amount mp as an expected value. It contributes scalability since one can reduce the $1/p$ times transactions. We also introduce a novel fractional oblivious transfer for the realization of the probabilistic payment. The functionality required for our proposal is the hashed time lock contract that various cryptocurrencies use. This request is general, not restricted to any particular cryptocurrency.

Acknowledgments

Firstly, I much appreciate my supervisor, Professor Akira Otsuka. Fortunately, when I started studying blockchain two years ago, he was willing to accept that he supervises me. He led me to the frontier in the blockchain. I could enjoy days full of intellectual excitement here. He has mentored me for these two years. I could not complete this work without his outstanding mentorship. I appreciate him for everything he has done for me.

I am incredibly grateful to Professor Seiko Arita, Professor Hiroshi Doi, and Professor Kazuyuki Shudo for their constructive suggestions and valuable advice. I was able to take the first step in the lattice-based cryptography from the introductory paper in the bulletin written by Prof. Arita. After that, he discussed the lattice-based cryptography and gave much guidance to me. Prof. Doi discussed the zero-knowledge proof and taught me various things. Prof. Arita and Prof. Doi gave me much advice and notices while in school. Prof. Shudo encouraged me in Atlanta, where I visited as my first international conference. I am delighted that he is willing to take on the review.

I am thankful to the members of Otsuka Laboratory. They have given me many cooperations, notices, and supports through my laboratory life. In particular, they taught us various aspects of blockchain.

I want to show my appreciation to my managers and colleagues for their understanding and cooperation. I appreciate that Mitsubishi Chemical Corporation and Mitsubishi Chemical Systems, Inc have supported this work. They gave me the precious opportunity of my studying blockchain in the Institute of Information Security. I could not fulfill this work without their excellent cooperation.

Finally, I kindly thank my family. They always encourage and help me. I sincerely appreciate them for their affectionateness while studying and staying at home.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

In this chapter, we describe the introduction consisting of the backgrounds, contributions, and organization.

1.1 Backgrounds

This section introduces three breakthroughs, blockchain, homomorphic encryption, and zero-knowledge proof that are deeply involved in this thesis as the backgrounds.

1.1.1 Blockchain

The first breakthrough is the blockchain. Blockchain is the underlying technology of Bitcoin. Nakamoto [Nak08] published a paper on Bitcoin in 2008. Bitcoin is a payment system that does not require a trusted third party, such as a bank. Participants called nodes individually manage their ledger that records transactions. Each node agrees on the new transactions and updates its ledger according to a consensus algorithm. Bitcoin realizes a decentralized payment system in this way.

Wüst *et al.* [WG18] classified blockchain into the following two types from the viewpoint of participation. One is a permissioned blockchain in which only authorized members can participate. Hyperledger Fabric [ABB+18] is an implementation of permissioned blockchains. The other is a permissionless blockchain that does not require permission to participate. Bitcoin, Ethereum [Woo14], and many cryptocurrencies are implementations of permissionless blockchains.

1.1.2 Homomorphic encryption

The second breakthrough is the fully homomorphic encryption. Fully homomorphic encryption is capable of both addition and multiplication in ciphertext state. It had been an open problem for about 30 years since the concept appeared. Gentry [Gen09] firstly announced fully homomorphic encryption in 2009.

After this breakthrough, various studies have sprouted. One direction of the studies is regarding an efficient scheme. The first realization is infeasible because a ciphertext size is too large. Regev [Reg09] defined the Learning With Errors (LWE) problem. He proved that the LWE problem results from a worst-case lattice problem, such as the shortest vector problem. Lyubashevsky *et al.* [LPR13] introduced the Ring Learning With Errors (RLWE) problem, which is an algebraic variant of the LWE problem. It is more efficient than the LWE problem, and the same level of safety is also guaranteed. Several schemes [BV11, FV12, BGV14] based on the RLWE problem were proposed. Also, several implementations [HS14, RR18, CLP17] based on these schemes are publicly available. López-Alt *et al.* [LATV12] proposed NTRU-like fully homomorphic encryption. Acar *et al.* [AAUC18] evaluated that the scheme is efficient but needs the new Decisional Small Polynomial Ratio (DSPR) assumption. We mainly consider homomorphic encryption based on the RLWE problem in this thesis.

Another direction is the lattice-based protocols. The RLWE scheme is the lattice-based cryptography. It has one aspect as post-quantum cryptography. For this reason, lattice-based protocols are one of the most important studies. Key exchange protocols [Pei14, BCNS15, ADPS16] and signature protocol [Lyu12] are proposed. Several zero-knowledge proofs that can indicate the plaintext knowledge to a verifier have been proposed [BCK$^+$14, BDLN16, PR17].

1.1.3 Zero-knowledge proof

The third breakthrough is zero-knowledge proof. A prover having secret convinces a verifier that the proposition on the secret is correct without any knowledge other than being correct. Goldwasser *et al.* [GMR89] firstly formulated this method. The zero-knowledge proof satisfies the following three conditions:

Completeness. A prover can convince a verifier if the prover has a true secret.

Soundness. If a cheating prover has no true secret, then the verifier can recognize that it is false.

Zero-knowledge. A cheating verifier attempting to steal the secret from a prover cannot obtain any knowledge other than "the proposition is true."

1.2 Contributions

Concerning blockchain, Bitcoin started operation in 2009. Technology has spread in advance, and society is following behind. There are issues to be solved before we can widely use blockchain in the real world. Throughput and privacy protection are significant issues.

Let us describe privacy protection. There are many studies on the privacy-preserving blockchain. Zhang *et al.* [ZXL19] classify seven characteristics as the characteristics of privacy: consistency, integrity, availability, prevention of double-spending, anonymity, confidentiality, and unlinkability. Bitcoin has already realized the first five characteristics. The other two are challenging. The first is the confidentiality of the transaction. That is the privacy of the transaction content. In the distributed ledger, each participant needs to verify the contents. Considering naive, it becomes necessary to disclose the contents of the transaction. However, this violates privacy protection. The other is the unlinkability of transactions. That is identity-related privacy. The transaction links the payer and payee typically. One can know the balance and trading frequency of the trader by analyzing the links of transactions. Breaking the transaction link is necessary to protect privacy. In the real world, blockchain not only keeps the contents secret but also is transparent. That is two sides of the same coin. For blockchain widely used in the real world, we believe blockchain needs to maintain the contents verifiable while keeping them private. We propose these blockchains in Chapters 3 and 4.

Let us state throughput. For example, payment processing is limited to once every 10 minute with Bitcoin. This throughput is much lower than the performance of credit card payments. If an open blockchain is unnecessary, one can adapt permissioned blockchain to get high throughput. Permissioned blockchain deals with more transactions than permissionless blockchain. We realize privacy-preserving permissioned blockchain in Chapter 3. For high throughput in permissionless blockchain, probabilistic payment is promis-

ing. Probabilistic payment reduces transaction volume and realizes high throughput. We achieve unlinkable and probabilistic payment in Chapter 5.

We show the overview of the contributions in Table 1.1. Chapter 3 achieves everything. However, the achievement is limited to a permisssioned blockchain. Chapter 4 realizes private and auditable permissionless blockchain. We focus on the first two in Chapter 4. Moreover, we focus on the next two in Chapter 5. In principle, a combination of these is a proposal that satisfies all. For blockchain widely used in the real world, we append to factors such as public verifiability and improvement of throughput. In this thesis, we study such privacy-preserving blockchain. We describe our contributions together with our motivations and the related works in detail as follows.

Table 1.1: Overview of the contributions

	public verifiability	confidentiality	unlinkability	high throughput
Bitcoin	yes	no	no	no
Chapter 3	yes	yes	yes	yes
Chapter 4	yes	yes	no	no
Chapter 5	no	no	yes	yes

1.2.1 Traceability in permissioned blockchain

Motivation. Processing transactions in the blockchain, we face low throughput and the scalability problem. Since we solve the problems, there are off-chain technologies that we can process transactions outside the blockchain (e.g., Lightning network [PD16]). Sidechain is an off-chain technology that connects a permissioned blockchain and a permissionless blockchain at the same layer level. Dilly *et al.* [DPW+16] have proposed this technique connecting between Bitcoin and their permissioned blockchain Liquid. Using this technology, we can transfer assets through a permissionless blockchain while enjoying high throughput processing in a permissioned blockchain.

A permissioned blockchain is a black box to outsiders. Corporations often use a permissioned blockchain together with authorized participants. They also want to protect trade privacy by keeping their transaction information secret in commercial activities. However, they need to disclose various types of information to fulfill their social responsibility. Transparency is vital for corporate activities. They must balance both trade privacy and

transparency. They need to share the contents of the permissioned blockchain with not only participants but also outsiders.

Related work. Let us compare the contribution of our work with the concurrent works [DPW+16, PBF+19, BAZB20] in Table 1.2. We compare from the perspective of the following three properties:

Trade privacy. Who trades with whom and at what asset amount.

Preservation. The total amount inside the permissioned blockchain, including deposits and withdrawals to the permissionless blockchain, is stable.

Noninvolvement. Some members in the permissioned blockchain are not involved in some trades. One can prove that specified members performed the transaction.

The pool account to a permissionless blockchain is public in Dilly *et al.* [DPW+16]. There are no unauthorized deposits or withdrawals in this account. Thus, preservation holds, although Dilly *et al.* do not specify this matter. Poelstra *et al.* [PBF+19] mention that there is no unauthorized increase or decrease in coin history, and the number of coins traded is concealed. However, the sender and receiver must be public. Zether [BAZB20] is the work on remittance with anonymous accounts at Ethereum. Zether executes deposit and withdrawal while preserving the whole balance. The balance between the sender and the receiver and the transfer amount can be kept confidential. There is no defined way to indicate noninvolvement later. Dummy participants must be involved in the transaction from the beginning. It is difficult for Zether to achieve noninvolvement and trade privacy simultaneously.

Table 1.2: Properties' comparison

	Preservation	Noninvolvement	Trade privacy
Dilly *et al.* [DPW+16]	yes	no	no
Poelstra *et al.* [PBF+19]	yes	yes	no
Zether [BAZB20]	yes	yes/no	no/yes
Our work	yes	yes	yes

Contribution. We have achieved privacy protection and high transparency in a permissioned blockchain. To improve transparency under privacy protection, we present the traceability in the permissioned blockchain consisting of the three properties: trade

privacy, preservation, and noninvolvement. Our approach is as follows. We model the traceability based on the hidden Markov model. Since the calculation of more than quadratic degrees is necessary, we encrypt this model by homomorphic encryption. We can construct an encrypted model by employing somewhat homomorphic encryption. The zero-knowledge proof indicates the establishment of the original model. The proof is a non-interactive zero-knowledge proof of the knowledge that the plaintext is equal to zero. This is an adaptation of Benhamouda *et al.* [BCK+14].

1.2.2 Confidential and auditable payments

Motivation. Since Bitcoin makes transaction information public, Bitcoin is transparent. However, a bank typically keeps the user's transaction information confidential. One hopes that blockchain also keeps transaction information concealed. There are several works realizing anonymity and confidentiality but do not have a forcibly auditable functionality. It is a problem that excessive confidentiality of transaction information may cause money laundering.

Related work. There are several works realizing anonymity and confidentiality. Zerocoin [MGGR13] and Zerocash [SCG+14] are extensions based on Bitcoin. They realized strong anonymity and confidentiality by designing anonymous coins that skillfully combined commitments. Zether [BAZB20] is an extension based on Ethereum. However, neither Zerocoin, Zerocash, nor Zether is forcibly auditable because of their strong anonymity and confidentiality.

Contribution. We construct the confidential and auditable payments scheme. The scheme allows a court or an authority to audit transactions while keeping the transaction information confidential. The scheme eliminates concerns about money laundering and contributes to the sound use of blockchain. Its characteristics are as follows.

- All the transactions are confidential by writing their ciphertexts in a ledger.
- The ownership and the transfer of it are provable by zero-knowledge proof. The soundness of the zero-knowledge proof can realize the soundness of the scheme.
- A court or authority controls a unique secret key of the ciphertexts in the ledger.

They can enforce confidential transactions open with the secret key according to the appropriate procedure.

1.2.3 Anonymous probabilistic payment in payment hub

Motivation. Privacy protection and scalability are significant problems with blockchain. However, there are a few proposals to solve these problems at the same time.

Let us describe privacy protection. Bitcoin publicly records each person's transaction history on its distributed ledgers. Since all transactions are open, miners can verify if the deals on the blockchain are correct. However, privacy is not protected. Many proposals keep transactions confidential while maintaining the verifiability.

Let us explain scalability. It is costly to write all the small transactions into the blockchain. The payment of a small amount of money is called micropayment. For micropayment, Wheeler [Whe97] and Rivest [Riv97] proposed a probabilistic payment before blockchain appears. The probabilistic payment reduces costs for micropayment. One pays an ordinary mount m with a certain probability p. One pays a small amount mp as an expected value. By the probability p (e.g. $p = 0.1 \sim 0.001$), one can reduce $1/p$ times transactions. Micropay [Ps15], the DAM scheme [CGL$^+$17] and Microcash [ABC20] have proposed a new micropayment on the blockchain. Micropayment is attracting attention as one of the leading solutions for scalability in the blockchain.

Related work. Let us confirm formerly anonymous cryptocurrencies. Zerocash [SCG$^+$14] is a famous anonymous cryptocurrency and is implemented as ZCash. Regarding ZCash, their group has proposed continuous researches such as BOLT [GM17] and DAM scheme [CGL$^+$17]. Monero is also a famous anonymous cryptocurrency and is provided with incredible works [Noe15, SALY17, YSL$^+$20, MSRL$^+$19]. Next, let us confirm studies to realize anonymity for existing cryptocurrencies by using off-chain technology. TumbleBit [HAB$^+$17] is compatible with Bitcoin. Zether [BAZB20] is compatible with Ethereum.

Wheeler [Whe97] and Rivest [Riv97] proposed a probabilistic payment. Micropay [Ps15] is compatible with Bitcoin. The DAM scheme, which is the extension of anonymous ZCash, also realizes a probabilistic payment.

Bellare and Micali [BM90] and Bellare and Rivest [BR99] proposed the fractional oblivious transfer based on the computational Diffie-Hellman assumption as the early

works. The DAM scheme also proposed a novel fractional oblivious transfer based on the decisional Diffie-Hellman assumption as fractional message transfer [CGL+17, CGL+16]. ([CGL+16] is the full version of [CGL+17].)

Brakerski and Döttling proposed an oblivious transfer based on the LWE problem [BD18]. This work is the first oblivious transfer in the post-quantum cryptography. Liu and Hu first proposed an efficient 1-out-of-2 oblivious transfer on the RLWE problem and extends 1-out-of-n oblivious transfer [LH19]. To the best of our knowledge, we first propose the fractional oblivious transfer over the ring.

Contribution. We propose an anonymous probabilistic payment. It aims to solve both scalability and privacy protection. We realize the anonymity, which is "k-anonymity in an epoch." TumbleBit [HAB+17] and their earlier work [HBG16] mentioned this definition. Anonymity says that the link, which payer pays which payee via a tumbler within an epoch, is broken. Even a tumbler never knows this link. The "k" is the number of participants trading via the tumbler. The epoch is the period during which transactions are completed. Our proposal includes a probabilistic payment. One can reduce $1/p$ times transactions, since one pays with a certain probability p (e.g. $p = 0.1 \sim 0.001$). We introduce a novel fractional oblivious transfer for a probabilistic payment. We call it the ring fractional oblivious transfer since this is based on the RLWE encryption. The functionality required for our proposal is the hashed time lock contract. This request is general, not restricted to any particular cryptocurrency.

1.3 Organization

Let us present the organization in this thesis. In Chapter 2, we describe the building blocks. We focus on RLWE encryption and zero-knowledge proof. In Chapter 3, we describe the first work, titled "Traceability in permissioned blockchain." It realizes both confidentiality and transparency in the permissioned blockchain. We call this meaningful traceability. In Chapter 4, we state the second work, titled "Confidential and auditable payments." It realizes confidential and auditable payments in the permissionless blockchain. In Chapter 5, we mention the third work, titled "Anonymous probabilistic payment in payment hub." It realizes an anonymous probabilistic payment, introducing a novel fractional oblivious transfer. In Chapter 6, we conclude this thesis.

Chapter 2

Building blocks

This chapter focuses on the ring learning with errors (RLWE) encryption and zero-knowledge proof.

2.1 Notation

Let $\mathbb{N}, \mathbb{Z}, \mathbb{Q}$ and $\mathbb{R}$ be the set of natural numbers, the set of integers, the set of rational numbers and the set of real numbers, respectively. Let the finite field $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z} = \{0, 1, \ldots, q-1\}$, where q is a prime number. Let $\mathbb{Z}[X]$ be the ring of polynomials over the integers. Let $\Phi_l \in \mathbb{Z}[X]$ be the l-th cyclotomic polynomial. We set l to a power of 2. We have the cyclotomic polynomial $\Phi_l = X^d + 1$, where $d = \phi(l)$ and ϕ is the Euler function. Let the ring of integers $\mathbf{R} = \mathbb{Z}[X]/\langle X^d + 1\rangle$, and let $\mathbf{R}_q = \mathbb{Z}_q[X]/\langle X^d + 1\rangle = \mathbb{Z}[X]/\langle X^d + 1, q\rangle$. $\mathbf{R}_q$ is the ring of integers $\mathbf{R}$ modulo q. We identify a vector $(a_0, \ldots, a_{d-1}) \in \mathbb{Z}^d$ with a polynomial $a_0 + a_1 X + \cdots + a_{d-1}X^{d-1} \in \mathbf{R}$. We denote the norms of $a = a_0 + a_1 X + \cdots + a_{d-1}X^{d-1} \in \mathbf{R}$ as follows. $|a_i|$ is the absolute value of a_i.

- l_2-norm $|a| = \sqrt{a_0^2 + a_1^2 + \cdots + a_{d-1}^2}$
- l_∞-norm $|a|_\infty = \max(|a_0|, |a_1|, \ldots, |a_{d-1}|)$

If $\forall c \in \mathbb{N}$, $\exists n_c$ such that $\forall n > n_c$ and $f(n) < 1/n^c$, then we denote the function f as a negligible function $\mathrm{negl}(n)$. If an element a is sampled randomly from a distribution A or a uniform distribution over the set A, then we denote $a \xleftarrow{\$} A$ with dollar sign. We denote the Big O notation $\tilde{\mathcal{O}}(\cdot)$, which is called Bachmann-Landau notation or asymptotic notation.

We denote the ω notation $\omega(f(n))$, which is an arbitrary function growing asymptotically faster than $f(n)$. We denote an oracle as $\mathcal{O}$.

2.2 Ring learning with errors encryption

In this section, we introduce the definitions regarding the ring learning with errors (RLWE) encryption.

2.2.1 Discrete Gaussian distribution

We describe the discrete Gaussian distribution.

- $\rho^d_{v,\sigma}(x) = (\frac{1}{\sqrt{2\pi}\sigma})^d e^{\frac{-|x-v|^2}{2\sigma^2}}$ is the continuous normal distribution over $\mathbb{R}^d$ centered at v with standard deviation σ. If $v = 0$, then we write $\rho^d_{v,\sigma}$ as ρ^d_σ.

- $D^d_{v,\sigma}(x) = \rho^d_{v,\sigma}(x)/\rho^d_\sigma(\mathbb{Z}^d)$ is the discrete normal distribution over $\mathbb{Z}^d$ centered at $v \in \mathbb{Z}^d$ with standard deviation σ. The quantity $\rho^d_\sigma(\mathbb{Z}^d) = \Sigma_{z\in\mathbb{Z}^d}\rho^d_\sigma(z)$ is just a normalized quantity. It is necessary to express the function as a probability distribution. We also mention that $\forall v \in \mathbb{Z}^d$, $\rho^d_{v,\sigma}(\mathbb{Z}^d) = \rho^d_\sigma(\mathbb{Z}^d)$. The scaling factor is the same for all v. If the dimension d is clear from the context, then we omit d and write D^d_σ as D_σ. We also denote D_σ as χ.

We introduce the below lemma regarding the discrete Gaussian distribution.

Lemma 2.1 (Lemma 4.4 in [MR07])**.** *Let $d \in \mathbb{N}$. For any number $\sigma > \omega(\sqrt{\log d})$, we have*

$$\Pr_{x\xleftarrow{\$}D_\sigma}[|x|_\infty > \sigma\sqrt{d}] \leq 2^{-d+1}.$$

Remark 2.1. *According to Lemma 2.1, $|x| \leq B$ with overwhelming probability if $x \xleftarrow{\$} D_\sigma$. B is a constant value.*

2.2.2 Assumption

Let us present the problems over the ring as follows.

Definition 2.1 (Ring learning with errors ($\mathsf{RLWE}_{\phi,q,\chi}$) problem)**.** *Let $\phi(X) \in \mathbb{Z}[X]$ be a polynomial of degree d, let $q \in \mathbb{Z}$ be a prime integer, let χ denote a distribution over*

the ring $\mathbf{R} = \mathbb{Z}[X]/\langle\phi(X)\rangle$*, and let* $\mathbf{R}_q = \mathbf{R}/q\mathbf{R}$*. The ring learning with errors problem* $\mathsf{RLWE}_{\phi,q,\chi}$ *is to distinguish between the following two distributions:* $(a, y) \in \mathbf{R}_q^2$ *such that* $a \xleftarrow{\$} \mathbf{R}_q,\ s, e \xleftarrow{\$} \chi$ *and* $y = as + e$*, and* $(a, y) \xleftarrow{\$} \mathbf{R}_q^2$*.*

The RLWE assumption indicates that the RLWE problem is hard for any probabilistic polynomial time (PPT) algorithm. The RLWE assumption still holds even if we choose the secret s according to the error distribution D_σ rather than uniformly [LPR13].

Definition 2.2 (Decisional small polynomial ratio ($\mathsf{DSPR}_{\phi,q,\chi}$) problem (Definition 3.4 in [LATV12]))**.** *Let* $\phi(X) \in \mathbb{Z}[X]$ *be a polynomial of degree* d*, let* $q \in \mathbb{Z}$ *be a prime integer, and let* χ *denote a distribution over the ring* $\mathbf{R} = \mathbb{Z}[X]/\langle\phi(X)\rangle$*. The decisional small polynomial ratio problem* $\mathsf{DSPR}_{\phi,q,\chi}$ *is to distinguish between the following two distributions: a polynomial* $h = g/f$*, where* f *and* g *are sampled from the distribution* χ *(conditioned on* f *being invertible over* $\mathbf{R}_q = \mathbf{R}/q\mathbf{R}$*), and a polynomial* $h \xleftarrow{\$} \mathbf{R}_q$*.*

According to [LATV12], let us explain a standard deviation of the discrete Gaussian distribution D_σ and DSPR assumption. It is known that the DSPR problem is hard if the standard deviation is significant. For the calculation using homomorphic encryption, we want to take a small one. In this case, it is assumed that the DSPR problem is still hard. This assumption is called the DSPR assumption.

2.2.3 RLWE encryption and its syntax

Now we introduce Brakerski-Vaikuntanathan (BV) scheme [BV11]. We describe plaintext space, key generation, encryption and decryption as follows.

- The plaintext space $\mathbf{M} = \mathbf{R}_p = \mathbb{Z}[X]/\langle X^d + 1, p\rangle$.
- To generate key, $a, s \xleftarrow{\$} \mathbf{R}_q,\ e_s \xleftarrow{\$} D_\sigma,\ b = as + e_s$, where (a, b) is a public key and s is a secret key.
- To encrypt a plaintext $m \in \mathbf{M}$, choose a set of randomness $v, e, f \xleftarrow{\$} D_\sigma$, and compute the ciphertext $c = (c_1, c_2) = (bv + pe + m, av + pf) \in \mathbf{C}$.
- To decrypt $c = (c_1, c_2)$ with secret key s and obtain a plaintext m, compute $m = (c_1 - s \cdot c_2 \mod q) \mod p$.

We also introduce the syntax of the RLWE scheme as follows.

Definition 2.3 (Syntax of the RLWE scheme (Definition 6 in [BKS19]))**.** *We describe the RLWE scheme represented by a tuple of probabilistic polynomial time algorithms*

$$\mathsf{RLWE} := (\mathsf{RLWE.Gen}, \mathsf{RLWE.Enc}, \mathsf{RLWE.Dec})$$

with the following syntax. We denote a message space as $\mathbf{M}$ *and a ciphertext space as* $\mathbf{C}$*.*

- $\mathsf{RLWE.Gen}(1^\lambda)$ *returns a key pair* $(\mathsf{pk}, \mathsf{sk})$ *from an input* 1^λ*.*
- $\mathsf{RLWE.Enc}(\mathsf{pk}, m, r)$ *returns a ciphertext* $c \in \mathbf{C}$ *from an input of the public key* pk*, a message* $m \in \mathbf{M}$ *and a randomness* r*.*
- $\mathsf{RLWE.Dec}(\mathsf{sk}, c)$ *returns a message* $m \in \mathbf{M}$ *or* $\perp$ *from an input of the secret key* sk *and a ciphertext* $c \in \mathbf{C}$*.*

Definition 2.4 (Pseudorandomness of ciphertexts (Definition 7 in [BKS19]))**.** *We say a RLWE scheme* $\mathsf{RLWE} := (\mathsf{RLWE.Gen}, \mathsf{RLWE.Enc}, \mathsf{RLWE.Dec})$ *satisfies pseudorandomness of ciphertexts or simply* RLWE *is secure, if for every PPT adversary* $\mathcal{A}$ *the advantage*

$$\mathrm{Adv}^{\mathsf{pr}}_{\mathsf{RLWE},\mathcal{A}}(\lambda) := \mid \Pr[\mathrm{Exp}^{\mathsf{pr}}_{\mathsf{RLWE},\mathcal{A}}(\lambda) = 1] - 1/2 \mid$$

is negligible in λ*, where* $\mathrm{Exp}^{\mathsf{pr}}_{\mathsf{RLWE},\mathcal{A}}(\lambda)$ *is as defined in Fig. 2.1.*

$\underline{\mathrm{Exp}^{\mathsf{pr}}_{\mathsf{RLWE},\mathcal{A}}(\lambda)}$:
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{RLWE.Gen}(1^\lambda)$
$\beta \leftarrow \{0, 1\}$
$\beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{RLWE.Enc}}(\cdot)}(1^\lambda, \mathsf{pk})$
if $\beta = \beta'$ return 1
else return 0

$\underline{\mathcal{O}_{\mathsf{RLWE.Enc}}(m)}$:
if $\beta = 0$
  $c \leftarrow \mathsf{RLWE.Enc}(\mathsf{pk}, m)$
else
  $c \xleftarrow{\$} \mathbf{R}_q^2$
return c

Figure 2.1: Security challenge experiment for pseudorandomness of ciphertexts.

Remark 2.2. *The randomness (that is, "noise") in the ciphertext increases with the addition and multiplication. For noise growth, multiplication is more remarkable than addition. To reduce the increasing noise, one can apply the processing called bootstrapping. However, this is costly. Homomorphic encryption without the bootstrapping is called somewhat homomorphic encryption.*

2.3 Zero-knowledge proof

In this section, we present the zero-knowledge proofs of knowledge for RLWE encryption.

2.3.1 Pedersen commitments

Let us introduce Pedersen commitments [Ped92]. Given a family of prime order groups $\{\mathbb{G}(\lambda)\}_{\lambda\in\mathbb{N}}$ such that the discrete logarithm problem is hard in $\mathbb{G}(\lambda)$ with security parameter λ, let $\tilde{q} = \tilde{q}(\lambda)$ be the order of $\mathbb{G} = \mathbb{G}(\lambda)$. We denote all elements with order $\tilde{q}$ with a tilde in the following. We write the group $\mathbb{G}(\lambda)$ additively.

- CSetup: This algorithm chooses $\tilde{g}, \tilde{h} \xleftarrow{\$} \mathbb{G}$ and outputs $cpars = (\tilde{g}, \tilde{h})$.
- Commit: To commit to a message $m \in \mathbb{Z}_{\tilde{q}}$, it first chooses $r \xleftarrow{\$} \mathbb{Z}_{\tilde{q}}$. It then outputs a pair $(\widetilde{cmt}, o) = (m\tilde{g} + r\tilde{h}, r)$.
- COpen: Given a commitment $\widetilde{cmt}$, an opening o, a public key $cpars$ and a message m, it outputs accept if and only if $(\widetilde{cmt}, o) \stackrel{?}{=} (m\tilde{g} + r\tilde{h}, r)$.

Lemma 2.2 (Theorem 2.1. in [BCK+14])**.** *Under the discrete logarithm assumption for $\mathbb{G}$, the given commitment scheme is perfectly hiding and is computationally binding.*

In this chapter's protocols, we make use of the above scheme as an auxiliary commitment scheme. We denote it as (aCSetup, aCCommit, aCOpen).

2.3.2 Rejection sampling

To realize zero-knowledge proof, the technique of rejection sampling is useful [Lyu09, Lyu12]. The technique can conceal the information of witness. Therefore, we apply the technique when a prover sends the response to a verifier.

Lemma 2.3 (Theorem 4.6 in [Lyu12])**.** *Let V be a subset of $\mathbb{Z}^m$ in which all elements have norms less than T, σ be some element in $\mathbb{R}$ such that $\sigma = \omega(T\sqrt{\log m})$, and $h : V \to \mathbb{R}$ be a probability distribution. Then, there exists a constant $M = \tilde{\mathcal{O}}(1)$ such that the distribution of the following algorithm $\mathcal{A}$:*

1. $v \xleftarrow{\$} h$
2. $z \xleftarrow{\$} D^m_{v,\sigma}$

3. *output* (z, v) *with probability* $\min(\frac{D_\sigma^m(z)}{MD_{v,\sigma}^m(z)}, 1)$

is within statistical distance $\frac{2^{-\omega(\log m)}}{M}$ *of the distribution of the following algorithm* $\mathcal{F}$*:*

1. $v \xleftarrow{\$} h$

2. $z \xleftarrow{\$} D_\sigma^m$

3. *output* (z, v) *with probability* $1/M$

Moreover, the probability that $\mathcal{A}$ *outputs something is at least* $\frac{1-2^{-\omega(\log m)}}{M}$*. More concretely, if* $\sigma = \alpha T$ *for any positive* α*, then* $M = e^{12/\alpha+1/(2\alpha^2)}$*, the output of algorithm* $\mathcal{A}$ *is within statistical distance* $\frac{2^{-100}}{M}$ *of the output of* $\mathcal{F}$*, and the probability that* $\mathcal{A}$ *outputs something is at least* $\frac{1-2^{-100}}{M}$*.*

2.3.3 Definition

We describe the formal definition of the Σ'-protocol, the protocol and its theorem proving this relationship.

Definition 2.5 (Definition 2.5. in [BCK+14])**.** *Let* (P, V) *be a two-party protocol, where* V *is a probabilistic polynomial time algorithim, and let* $L, L' \subseteq \{0,1\}^*$ *be languages with witness relations* R, R' *such that* $R \subseteq R'$*. Then,* (P, V) *is called a* Σ'*-protocol for* L, L' *with completeness error* α*, a challenge set* $\mathbb{C}$*, a public input* x *and a private input* w*, if and only if it satisfies the following conditions:*

- *Three-move form: The prover* P*, on input* (x, w)*, computes a commitment* t *and sends it to* V*. The verifier* V*, on input* x*, then draws a challenge* $c \xleftarrow{\$} \mathbb{C}$ *and sends it to* P*. The prover sends a response* s *to the verifier. Depending on the protocol transcript* (t, c, s)*, the verifier finally accepts or rejects the proof. The protocol transcript* (t, c, s) *is called accepting, if the verifier accepts the protocol run.*

- *Completeness: Whenever* $(x, w) \in R$*, the verifier* V *accepts with probability at least* $1 - \alpha$*.* α *is the completeness error.*

- *Special soundness: There exists a PPT algorithm* E *(the knowledge extractor) that takes two accepting transcripts* $(t, c', s'), (t, c'', s'')$ *satisfying* $c' \neq c''$ *as inputs, and outputs* w' *such that* $(x, w') \in R'$*. The knowledge error denotes the probability that the verifier accepts the proof even if the prover does not know a witness.*

- *Special honest verifier zero-knowledge (HVZK): There exists a PPT algorithm S (the simulator) taking $x \in L$ and $c \in \mathbb{C}$ as inputs. Moreover, the simulator outputs (t, s) so that the triple (t, c, s) is indistinguishable from an accepting protocol transcript generated by a real protocol run.*

Moreover, let us present the following useful property.

- High-entropy commitments: For all $(y, w) \in R$ and for all t, the probability that an honestly generated commitment by P takes on the value t is negligible.

2.3.4 Relation for the key generation

We want to show that we know a public key's private key without revealing the private key itself. We introduce the zero-knowledge proof for the purpose in this subsection. Let us confirm Theorem 3.3. in [BCK+14].

Lemma 2.4 (Theorem 3.3. in [BCK+14])**.** *The protocol in Fig. 2.2 is an HVZK Σ'-protocol for the following relations:*

$$\mathcal{R} = \{((a, y), (s, e)) : y = as + e \wedge |s|, |e| \leq \tilde{\mathcal{O}}(\sqrt{d}\alpha)\}$$
$$\mathcal{R}' = \{((a, y), (s, e)) : 2y = 2as + 2e \wedge |2s|, |2e| \leq \tilde{\mathcal{O}}(d^2\alpha)\}$$

where $2s$ and $2e$ are reduced modulo q. The protocol has a knowledge error of $1/(2d)$, a completeness error of $1 - 1/M$, and high-entropy commitments.

We can use Lemma 2.4 for the below relation $\mathcal{R}_K$ by replacing $y \to y/p, a \to a/p$.

$$\mathcal{R}_K = \{((a, y), (s, e)) \mid y = as + 2e \wedge |s|, |e| \leq \tilde{\mathcal{O}}(\sqrt{d}\alpha)\}$$

Let us describe the non-interactive zero-knowledge proof for the relation $\mathcal{R}_K$. The protocol in Fig. 2.2 realizes the zero-knowledge proof for the relation $\mathcal{R}_K$. Moreover, we can obtain the non-interactive zero-knowledge proof by using Fiat-Shamir transform [FS87] and running the protocol in parallel. Let us confirm the simulator for the relation $\mathcal{R}_K$. The technique of rejection sampling is used for zero-knowledge in the protocol. According to Lemma 2.3, the algorithm $\mathcal{F}$ includes no witness and is statistically indistinguishable from the algorithm $\mathcal{A}$. We can implement the simulator for $\mathcal{R}_K$ by replacing the algorithm $\mathcal{A}$ with the algorithm $\mathcal{F}$.

Common input: the open value (a, y)
Relation: $R = \{((a, y), (s, e)) : y = as + e \wedge |s|, |e| \leq \tilde{\mathcal{O}}(\sqrt{d}\alpha)\}$

Prover		Verifier
$r_s, r_e \xleftarrow{\$} D_{\tilde{\mathcal{O}}(\sqrt{d}\alpha)}$		
$t = ar_s + r_e$		
$(c_{\text{aux}}, d_{\text{aux}}) = \mathsf{aCommit}(t)$		
	$\xrightarrow{c_{\text{aux}}}$	
		$c \xleftarrow{\$} \mathbb{C} = \{0, \ldots, 2d-1\}$
	$\xleftarrow{c}$	
$s_s = r_s + X^c s$		
$s_e = r_e + X^c e$		
accept with probability $\frac{D_{\tilde{\mathcal{O}}(\sqrt{2d}\alpha)}((s_e, s_s))}{MD_{(X^c e, X^c s), \tilde{\mathcal{O}}(\sqrt{2d}\alpha)}((s_e, s_s))}$		
	$\xrightarrow{t, d_{\text{aux}}, (s_s, s_e)}$	
		$X^c y + t \stackrel{?}{=} as_s + s_e$
		$\mathsf{aCOpen}(t, c_{\text{aux}}, d_{\text{aux}}) \stackrel{?}{=} \mathsf{accept}$
		$\lvert s_s\rvert, \lvert s_e\rvert \leq \tilde{\mathcal{O}}(d\alpha)$

Figure 2.2: Zero-knowledge proof of knowledge of RLWE secrets s, e such that $y = as + e$ (Protocol 3.2. in [BCK+14])

2.3.5 Relation for the ciphertext of zero

We want to show that we know that a given ciphertext's plaintext is zero without decryption or revealing randomness of the ciphertext. We introduce the zero-knowledge proof for the purpose in this subsection. We introduce the following technical lemmas.

Lemma 2.5 (Lemma 3.1 in [BCK+14]). *Let d be a power of 2 and let $0 < i, j < 2d - 1$. Then, $2(X^i - X^j)^{-1} \mod (X^d + 1)$ only has coefficients in $\{-1, 0, 1\}$.*

Lemma 2.6 (Lemma 2 in [ZZD+15]). *For any $s, t \in \mathbf{R}$, we have $|s \cdot t| \leq \sqrt{d} \cdot |s| \cdot |t|$ and $|s \cdot t|_\infty \leq d \cdot |s|_\infty \cdot |t|_\infty$.*

We show a non-interactive zero-knowledge proof for ciphertext of zero in Fig. 2.3. h is a cryptographic hash function. We make the interactive proof non-interactive by using the Fiat-Shamir heuristic [FS87]. This protocol satisfies Lemma 2.7. The parallel protocol satisfies Theorem 2.1.

Lemma 2.7 (Lemma 5 in [MO20c]). *The protocol in Fig. 2.3 is an HVZK Σ'-protocol*

Common input: the public key (a, b), the ciphertext (c_1, c_2)
Relation: $R_0 = \{((c_1, c_2), (v, e, f)) : (c_1, c_2) = (bv + pe, av + pf)$
$\wedge |v|, |e|, |f| \leq \tilde{\mathcal{O}}(\sqrt{d}\alpha)\}$

Prover		Verifier
$r_v, r_e, r_f \xleftarrow{\$} D_{\tilde{\mathcal{O}}(\sqrt{d}\alpha)}$		
$t_1 = br_v + r_e$		
$t_2 = ar_v + r_f$		
$(c^{(1)}_{\text{aux}}, d^{(1)}_{\text{aux}}) = \mathsf{aCommit}(t_1)$		
$(c^{(2)}_{\text{aux}}, d^{(2)}_{\text{aux}}) = \mathsf{aCommit}(t_2)$		
	$\xrightarrow{c^{(1)}_{\text{aux}}, c^{(2)}_{\text{aux}}}$	
$c = h(t_1, t_2, c^{(1)}_{\text{aux}}, d^{(1)}_{\text{aux}}, c^{(2)}_{\text{aux}}, d^{(2)}_{\text{aux}})$		
$s_v = r_v + X^c v$		
$s_e = r_e + X^c pe$		
$s_f = r_f + X^c pf$		
accept with probability $\frac{D_{\tilde{\mathcal{O}}(\sqrt{3d}\alpha)}((s_v, s_e, s_f))}{MD_{(X^c v, X^c e, X^c f), \tilde{\mathcal{O}}(\sqrt{3d}\alpha)}((s_v, s_e, s_f))}$		
	$\xrightarrow{t_1, t_2, d^{(1)}_{\text{aux}}, d^{(2)}_{\text{aux}}, (s_v, s_e, s_f)}$	
		$c = h(t_1, t_2, c^{(1)}_{\text{aux}}, d^{(1)}_{\text{aux}}, c^{(2)}_{\text{aux}}, d^{(2)}_{\text{aux}})$
		$X^c c_1 + t_1 \stackrel{?}{=} bs_v + s_e$
		$X^c c_2 + t_2 \stackrel{?}{=} as_v + s_f$
		$\mathsf{aCOpen}(t_1, c^{(1)}_{\text{aux}}, d^{(1)}_{\text{aux}}) \stackrel{?}{=} \mathsf{accept}$
		$\mathsf{aCOpen}(t_2, c^{(2)}_{\text{aux}}, d^{(2)}_{\text{aux}}) \stackrel{?}{=} \mathsf{accept}$
		$\lvert s_v\rvert, \lvert s_e\rvert, \lvert s_f\rvert \leq \tilde{\mathcal{O}}(d\alpha)$

Figure 2.3: Non-interactive zero-knowledge proof of a ciphertext of zero regarding RLWE encryption (Figure 3 in [MO20c])

for the following relations:

$$\mathcal{R}_0 = \{((c_1, c_2), (v, e, f)) : (c_1, c_2) = (bv + pe, av + pf) \wedge |v|, |e|, |f| \leq \tilde{\mathcal{O}}(\sqrt{d}\alpha)\}$$
$$\mathcal{R}'_0 = \{((c_1, c_2), (v, e, f)) : (2c_1, 2c_2) = (2bv + 2pe, 2av + 2pf) \wedge |2v|, |2e|, |2f| \leq \tilde{\mathcal{O}}(d^2\alpha)\}$$

where $2v, 2e$ *and* $2f$ *are reduced modulo* q*. The protocol has a knowledge error of* $1/(2d)$*, a completeness error of* $1 - 1/M$*, and high-entropy commitments.*

Proof. We discuss the proof from the following points: completeness, honest verifier zero-knowledge, special soundness, and high-entropy commitments.

Completeness. From Lemma 2.3 of the rejection sampling, a prover responds with a probability $1/M$. If the prover does not abort, then

$$\begin{aligned} bs_v + s_e &= b(r_v + X^c v) + (r_e + X^c pe) \\ &= X^c(bv + pe) + (br_v + r_e) \\ &= X^c c_1 + t_1 \end{aligned}$$

and

$$\begin{aligned} as_v + s_f &= a(r_v + X^c v) + (r_f + X^c pf) \\ &= X^c(av + pf) + (ar_v + r_f) \\ &= X^c c_2 + t_2 \end{aligned}$$

Regarding the norms, we obtain

$$\begin{aligned} |s_v| &\leq |r_v| + |v| \leq \tilde{\mathcal{O}}(d\alpha) \\ |s_e| &\leq |r_e| + |e| \leq \tilde{\mathcal{O}}(d\alpha) \\ |s_f| &\leq |r_f| + |f| \leq \tilde{\mathcal{O}}(d\alpha) \end{aligned}$$

with overwhelming probability, since the standard deviations of r_v, r_e, r_f are $\tilde{\mathcal{O}}(\sqrt{d}\alpha)$.

Honest verifier zero-knowledge. The challenge value c is randomly chosen from the set $\mathbb{C} = \{0, \ldots, 2d - 1\}$. The simulator outputs the tuple $(\mathsf{aCommit}(0), c, \perp)$ with the

probability $1 - 1/M$. With the probability $1/M$, the simulator runs as follows.

$$\begin{aligned} r_v, r_e, r_f &\xleftarrow{\$} D_{\tilde{\mathcal{O}}(\sqrt{d}\alpha)} \\ t_1 &= br_v + r_e - X^c c_1 \\ t_2 &= ar_v + r_f - X^c c_2 \\ (c^{(1)}_{\mathrm{aux}}, d^{(1)}_{\mathrm{aux}}) &= \mathsf{aCommit}(t_1) \\ (c^{(2)}_{\mathrm{aux}}, d^{(2)}_{\mathrm{aux}}) &= \mathsf{aCommit}(t_2) \end{aligned}$$

Finally, the simulator outputs

$$(c^{(1)}_{\mathrm{aux}}, c^{(2)}_{\mathrm{aux}}, c, t_1, t_2, d^{(1)}_{\mathrm{aux}}, d^{(2)}_{\mathrm{aux}}, (s_v, s_e, s_f)).$$

From Lemma 2.3, the outputs s_v, s_e and s_f without abort do not depend on v, e and f as the witness. Therefore, the simulator and the actual protocol are indistinguishable. If the protocol aborts, then it is indistinguishable because of the hiding property of $\mathsf{aCommit}$ in Lemma 2.2. For any c, an abort occurs in the same way.

Special soundness. We suppose that both

$$(c^{(1)}_{\mathrm{aux}}, c^{(2)}_{\mathrm{aux}}, c', t'_1, t'_2, d^{(1)'}_{\mathrm{aux}}, d^{(2)'}_{\mathrm{aux}}, (s'_v, s'_e, s'_f))$$

and

$$(c^{(1)}_{\mathrm{aux}}, c^{(2)}_{\mathrm{aux}}, c'', t''_1, t''_2, d^{(1)''}_{\mathrm{aux}}, d^{(2)''}_{\mathrm{aux}}, (s''_v, s''_e, s''_f))$$

pass the verification by the verifier. From the binding property of the auxiliary commitment scheme in Lemma 2.2, we obtain $t_1 := t'_1 = t''_1$ and $t_2 := t'_2 = t''_2$. We can have a similar discussion with regard to t_1 and t_2. Let us confirm t_1. From the verification equation, we have

$$\begin{aligned} X^{c'} c_1 + t_1 &= bs'_v + s'_e \\ X^{c''} c_1 + t_1 &= bs''_v + s''_e. \end{aligned}$$

From the subtraction between these two equations, we obtain

$$(X^{c'} - X^{c''})c_1 = b(s'_v - s''_v) + (s'_e - s''_e).$$

Multiplying by $2(X^{c'} - X^{c''})^{-1}$ to the equation,

$$2c_1 = b\frac{2(s'_v - s''_v)}{X^{c'} - X^{c''}} + \frac{2(s'_e - s''_e)}{X^{c'} - X^{c''}} =: 2b\hat{v} + 2\hat{e}.$$

Therefore, we obtain

$$|2\hat{v}| \leq |s'_v - s''_v| \cdot \sqrt{d} \cdot \left|\frac{2}{X^{c'} - X^{c''}}\right| \leq \tilde{\mathcal{O}}(d^2\alpha).$$

where we had the second inequality applying Lemma 2.5 and 2.6. A similar discussion holds for $\hat{e}$. We also have

$$|2\hat{e}| \leq |s'_e - s''_e| \cdot \sqrt{d} \cdot \left|\frac{2}{X^{c'} - X^{c''}}\right| \leq \tilde{\mathcal{O}}(d^2\alpha).$$

In addition, dealing with the verification equation regarding t_2 in a similar way of t_1, we obtain

$$2c_2 = a\frac{2(s'_v - s''_v)}{X^{c'} - X^{c''}} + \frac{2(s'_f - s''_f)}{X^{c'} - X^{c''}} =: 2a\hat{v} + 2\hat{f}.$$

In the same way, we obtain

$$|2\hat{f}| \leq |s'_f - s''_f| \cdot \sqrt{d} \cdot \left|\frac{2}{X^{c'} - X^{c''}}\right| \leq \tilde{\mathcal{O}}(d^2\alpha).$$

High-entropy commitments. This property directly follows from the security of the auxiliary commitment scheme. □

Theorem 2.1 (Theorem 6 in [MO20c])**.** *Let us apply the protocol in Fig. 2.3 for λ times in parallel (the parallel protocol). Let the parallel protocol be accepting if and only if at least $\lambda/2M$ out of λ proofs were valid under the condition that an honest verifier rejects no proofs. Then, the parallel protocol has both a completeness error and knowledge error of* $\mathrm{negl}(\lambda)$ *under the condition $d \geq 2M$.*

Proof. In the parallel protocol, λ commitments $\{c^{(1)}_{\mathrm{aux}}, c^{(2)}_{\mathrm{aux}}\}_{1,\ldots,\lambda}$ are sent to a verifier. We confirm the probability that at least $\lambda/2M$ out of λ proofs are valid. Each execution in

the parallel protocol is independent. It is a Bernoulli trial. We introduce the lemma of Chernoff bounds as follows.

Lemma 2.8 (Chernoff bounds (see e.g., Theorem 4.4, 4.5 in [MU17])). *Let $x_1, \ldots, x_\lambda$ be independent Bernoulli-distributed random variables with $\Pr[x_i = 1] = p$ and $\Pr[x_i = 0] = 1 - p$; then, for $X := \sum_{i=1}^{\lambda} x_i$ and $\mu := \lambda p$,*

$$\Pr[X \leq (1-\delta)\mu] \leq e^{-\delta^2\mu/2},\ 0 < \delta \leq 1$$
$$\Pr[X \geq (1+\delta)\mu] \leq e^{-\delta^2\mu/3},\ 0 < \delta \leq 1$$
$$\Pr[X \geq (1+\delta)\mu] \leq e^{-\delta\mu/3},\ 1 \leq \delta$$

hold.

First, we confirm a completeness error. From Lemma 2.7, a prover sends proof with a probability $1/M$ for each time. Applying first and second inequalities of Chernoff bounds under the condition that $p = 1/M, \mu = \lambda p = \lambda/M$ and $\delta = 2/3$,

$$\begin{aligned}
&\Pr[(1-\delta)\mu < X < (1+\delta)\mu] \\
&= 1 - \Pr[X \leq (1-\delta)\mu] - \Pr[X \geq (1+\delta)\mu] \\
&\geq 1 - (e^{-\delta^2\mu/2} + e^{-\delta^2\mu/3}) \\
&= 1 - (e^{-2\lambda/9M} + e^{-4\lambda/27M})
\end{aligned}$$

Since $\lambda/2M = \mu/2 \in ((1-\delta)\mu, (1+\delta)\mu) = (\mu/3, 5\mu/3)$, $\lambda/2M$ proofs are made and accepted with overwhelming probability. Therefore, the completeness error is $\mathrm{negl}(\lambda)$.

Second, we confirm a knowledge error. From Lemma 2.7, a knowledge error is $1/2d$ for each time. Let $p = 1/2d, \mu = \lambda p = \lambda/2d$ and $\delta = d/M - 1 \geq 1$ such that $(1+\delta)\mu = \lambda/2M$. Applying the third inequalities of Chernoff bounds,

$$\begin{aligned}
\Pr[X \geq (1+\delta)\mu] &\leq e^{-\delta\mu/3} \\
&= e^{-\lambda/6M+1/3}
\end{aligned}$$

Therefore, the knowledge error is $\mathrm{negl}(\lambda)$. □

2.3.6 Relation for valid ciphertext

We want to show that we know that a given ciphertext is valid without decryption or revealing a ciphertext's message and randomness. We introduce the zero-knowledge proof for the purpose in this subsection. Let us introduce the relation $\mathcal{R}_E$ as follows. In the relation $\mathcal{R}_E$, let a message space $\mathbf{M} = \mathbf{R}_2$.

$$\begin{aligned}\mathcal{R}_E = \{((c_1, c_2, a, y), (m, v, e, f)) \mid c_1 = yv + 2e + m \ \wedge \ c_2 = av + 2f \\ \wedge \ |m|_\infty \leq 1 \ \wedge \ |v|, |e|, |f| \leq \tilde{\mathcal{O}}(\sqrt{d}\alpha)\}\end{aligned}$$

We construct a non-interactive zero-knowledge proof for the RLWE ciphertext, referring to Protocol 4.1 and Theorem 4.2 in [BCK+14]. We present the protocol in Fig. 2.4. This protocol in Fig. 2.4 satisfies Lemma 2.9. We can make the completeness error and knowledge error in Lemma 2.9 negligible, by the parallelization as Theorem 2.1. Let us explain the notation $a_{\ll l}$. One obtains $a_{\ll}$ by an anti-cyclic shift a vector a by l. That is, $a_{\ll l} = aX^l \in \mathbf{R}$. If we write the elements of the vector explicitly, we have

$$a_{\ll l} = (a_0, \ldots, a_{d-1})_{\ll l} = (-a_{d-l}, \ldots, -a_{d-1}, a_0, \ldots, a_{d-l-1}).$$

Lemma 2.9. *The protocol in Fig. 2.4 is an honest verifier zero-knowledge Σ'-protocol for the following relations:*

$$\begin{aligned}\mathcal{R} = \{((c_1, c_2, \tilde{g}, \tilde{h}, (\widetilde{cmt}_i)_{i=0}^{d-1}, p, a, b), (m, v, e, f, (r_i)_{i=0}^{d-1})) : \\ (c_1, c_2) = (bv + pe + m, av + pf) \wedge \bigwedge_{i=0}^{d-1} \widetilde{cmt}_i = r_{m,i}\tilde{g} + r_{r,i}\tilde{h} \\ \wedge |m|_\infty \leq 1 \wedge |v|, |e|, |f| \leq \tilde{\mathcal{O}}(\sqrt{d}\alpha)\} \\ \mathcal{R}' = \{((c_1, c_2, \tilde{g}, \tilde{h}, (\widetilde{cmt}_i)_{i=0}^{d-1}, p, a, b), (m, v, e, f, (r_i)_{i=0}^{d-1})) : \\ (2c_1, 2c_2) = (2bv + 2pe + 2m, 2av + 2pf) \wedge \bigwedge_{i=0}^{d-1} 2\widetilde{cmt}_i = (2m \mod q)_i\tilde{g} + 2r_{r,i}\tilde{h} \\ \wedge |2m| \leq 2d^2 \wedge |2v|, |2e|, |2f| \leq \tilde{\mathcal{O}}(d^2\alpha)\}\end{aligned}$$

where $2v, 2e$ and $2f$ are reduced modulo q and $(2m \mod q)_i$ is the i-coefficient of $2m \in \mathbf{R}_q$. The protocol has a knowledge error of $1/(2d)$, a completeness error of $1 - 1/M$, and high-

Common input: the ciphertext (c_1, c_2), the integer p, the public key (a, b), the public key of commitment $(\tilde{g}, \tilde{h})$, the commitment $(\widetilde{cmt}_i)_{i=0}^{d-1}$
Relation $\mathcal{R} = \{((c_1, c_2, \tilde{g}, \tilde{h}, (\widetilde{cmt}_i)_{i=0}^{d-1}, p, a, b), (m, v, e, f, (r_i)_{i=0}^{d-1})) :$
$(c_1, c_2) = (bv + pe + m, av + pf) \wedge \bigwedge_{i=0}^{d-1} \widetilde{cmt}_i = r_{m,i}\tilde{g} + r_{r,i}\tilde{h}$
$\wedge |m|_\infty \leq 1 \wedge |v|, |e|, |f| \leq \tilde{\mathcal{O}}(\sqrt{d}\alpha)\}$

Prover		Verifier
$r_v, r_e, r_f \xleftarrow{\$} D_{\tilde{\mathcal{O}}(\sqrt{d}\alpha)}$		
$r_m \xleftarrow{\$} D_{\tilde{\mathcal{O}}(\sqrt{d})}$		
$r_{r,i} \xleftarrow{\$} \mathbb{Z}_{\tilde{q}}$ for $i = 0, \ldots, d-1$		
$t_1 = br_v + pr_e + r_m$		
$t_2 = ar_v + pr_f$		
$\tilde{t}_i = r_{m,i}\tilde{g} + r_{r,i}\tilde{h}$ for $i = 0, \ldots, d-1$		
$(c_{\text{aux}}, d_{\text{aux}}) = \mathsf{aCommit}(t_1, t_2, (\tilde{t}_i)_{i=0}^{d-1})$		
	$\xrightarrow{c_{\text{aux}}}$	
$c = h(t_1, t_2, (\tilde{t}_i)_{i=0}^{d-1}, c_{\text{aux}}, d_{\text{aux}})$		
$s_v = r_v + X^c v$		
$s_e = r_e + X^c e$		
$s_f = r_f + X^c f$		
$s_m = r_m + X^c m$		
$s_r = r_r + X^c r$		
accept with probability $\frac{D_{\tilde{\mathcal{O}}(2\sqrt{d}\alpha)}((s_v, s_e, s_f, s_m))}{MD_{(X^c v, X^c e, X^c f, X^c m), \tilde{\mathcal{O}}(2\sqrt{d}\alpha)}((s_v, s_e, s_f, s_m))}$		
	$\xrightarrow{t_1, t_2, (\tilde{t}_i)_{i=0}^{d-1}, d_{\text{aux}}, (s_v, s_e, s_f, s_m, s_r)}$	
		$c = h(t_1, t_2, (\tilde{t}_i)_{i=0}^{d-1}, c_{\text{aux}}, d_{\text{aux}})$
		$X^c c_1 + t_1 \stackrel{?}{=} bs_v + ps_e + s_m$
		$X^c c_2 + t_2 \stackrel{?}{=} as_v + ps_f$
		$(\widetilde{cmt}_0, \ldots, \widetilde{cmt}_{d-1})_{\ll c} + (\tilde{t}_0, \ldots, \tilde{t}_{d-1}) \stackrel{?}{=} s_m\tilde{g} + s_r\tilde{h}$
		$\mathsf{aCOpen}((t_1, t_2, (\tilde{t}_i)_{i=0}^{d-1}), c_{\text{aux}}, d_{\text{aux}}) \stackrel{?}{=} \mathsf{accept}$
		$\lvert s_v\rvert, \lvert s_e\rvert, \lvert s_f\rvert \leq \tilde{\mathcal{O}}(d\alpha)$
		$\lvert s_m\rvert \leq \tilde{\mathcal{O}}(d)$

Figure 2.4: Non-interactive zero-knowledge proof of plaintext knowledge regarding RLWE encryption

entropy commitments.

Proof. Let us confirm the proof from the following points: completeness, honest verifier zero-knowledge, special soundness, and high-entropy commitments.

Completeness. From Lemma 2.3 of the rejection sampling, a prover responds with a probability $1/M$. If the prover does not abort, then

$$\begin{aligned} bs_v + ps_e + s_m &= b(r_v + X^c v) + p(r_e + X^c e) + (r_m + X^c m) \\ &= X^c(bv + pe + m) + (br_v + pr_e + r_m) \\ &= X^c c_1 + t_1 \end{aligned}$$

and

$$\begin{aligned} as_v + ps_f &= a(r_v + X^c v) + p(r_f + X^c f) \\ &= X^c(av + pf) + (ar_v + pr_f) \\ &= X^c c_2 + t_2. \end{aligned}$$

Let us confirm the equation of the commitment

$$(\widetilde{cmt}_0, \ldots, \widetilde{cmt}_{d-1})_{\ll c} + (\tilde{t}_0, \ldots, \tilde{t}_{d-1}) \stackrel{?}{=} s_m \tilde{g} + s_r \tilde{h}.$$

Let us expand the first term of the left hand side.

$$\begin{aligned} (\widetilde{cmt}_0, \ldots, \widetilde{cmt}_{d-1})_{\ll c} &= (m_0 \tilde{g} + r_0 \tilde{h}, \ldots, m_{d-1} \tilde{g} + r_{d-1} \tilde{h})_{\ll c} \\ &= (m_0, \ldots, m_{d-1})_{\ll c} \tilde{g} + (r_0, \ldots, r_{d-1})_{\ll c} \tilde{h} \\ &= X^c m \tilde{g} + X^c r \tilde{h} \end{aligned}$$

We also expand the second term of the left hand side.

$$\begin{aligned} (\tilde{t}_0, \ldots, \tilde{t}_{d-1}) &= (r_{m,0} \tilde{g} + r_{r,0} \tilde{h}, \ldots, r_{m,d-1} \tilde{g} + r_{r,d-1} \tilde{h}) \\ &= (r_{m,0}, \ldots, r_{m,d-1}) \tilde{g} + (r_{r,0}, \ldots, r_{r,d-1}) \tilde{h} \\ &= r_m \tilde{g} + r_r \tilde{h} \end{aligned}$$

Therefore we have

$$\begin{aligned}(\widetilde{cmt}_0, \ldots, \widetilde{cmt}_{d-1})_{\ll c} + (\tilde{t}_0, \ldots, \tilde{t}_{d-1}) &= (X^c m\tilde{g} + X^c r\tilde{h}) + (r_m\tilde{g} + r_r\tilde{h}) \\ &= (r_m + X^c m)\tilde{g} + (r_r + X^c r)\tilde{h} \\ &= s_m\tilde{g} + s_r\tilde{h}.\end{aligned}$$

Let us confirm the norms of s_v, s_e, s_f and s_m. The standard deviations of r_v, r_e, r_f are $\tilde{\mathcal{O}}(\sqrt{d}\alpha)$. Also, the standard deviations of r_m is $\tilde{\mathcal{O}}(\sqrt{d})$. We have

$$\begin{aligned}|s_v| &\leq |r_v| + |v| \leq \tilde{\mathcal{O}}(d\alpha) \\ |s_e| &\leq |r_e| + |e| \leq \tilde{\mathcal{O}}(d\alpha) \\ |s_f| &\leq |r_f| + |f| \leq \tilde{\mathcal{O}}(d\alpha) \\ |s_m| &\leq |r_m| + |m| \leq \tilde{\mathcal{O}}(d)\end{aligned}$$

with the overwhelming probability.

Honest verifier zero-knowledge. A challenge value c is randomly chosen from the set $\mathbb{C} = \{0, \ldots, 2d-1\}$. A simulator outputs the tuple $(\mathsf{aCommit}(0), c, \perp)$ with the probability $1 - 1/M$. With the probability $1/M$, the simulator runs as follows.

$$\begin{aligned}r_v, r_e, r_f &\xleftarrow{\$} D_{\tilde{\mathcal{O}}(\sqrt{d}\alpha)} \\ r_m &\xleftarrow{\$} D_{\tilde{\mathcal{O}}(\sqrt{d})} \\ t_1 &= br_v + pr_e + r_m - X^c c_1 \\ t_2 &= ar_v + pr_f - X^c c_2 \\ \tilde{t}_i &= r_{m,i}\tilde{g} + r_{r,i}\tilde{h} \quad \text{for } i = 0, \ldots, d-1 \\ (c_{\mathrm{aux}}, d_{\mathrm{aux}}) &= \mathsf{aCommit}(t_1, t_2, (\tilde{t}_i)_{i=0}^{d-1})\end{aligned}$$

Finally, the simulator outputs

$$(c_{\mathrm{aux}}, d_{\mathrm{aux}}, c, t_1, t_2, (\tilde{t}_i)_{i=0}^{d-1}, (s_v, s_e, s_f, s_m)).$$

From Lemma 2.3, the outputs s_v, s_e, s_f and s_m without abort do not depend on v, e, f and m as the witness. Therefore, the simulator and the actual protocol are indistinguishable. If

the protocol aborts, then it is indistinguishable because of the hiding property of aCommit. For any c, the abort occurs in the same way.

Special soundness. Let us consider knowledge extractor and suppose that both

$$(c_{\mathrm{aux}}, d'_{\mathrm{aux}}, c', t'_1, t'_2, (\tilde{t}'_i)_{i=0}^{d-1}, (s'_v, s'_e, s'_f, s'_m, s'_r))$$

and

$$(c_{\mathrm{aux}}, d''_{\mathrm{aux}}, c'', t''_1, t''_2, (\tilde{t}''_i)_{i=0}^{d-1}, (s''_v, s''_e, s''_f, s''_m, s''_r))$$

pass the verification by the verifier. From the binding property of aCommit, we have

$$t_1 := t'_1 = t''_1,\ t_2 := t'_2 = t''_2,\ (\tilde{t}_i)_{i=0}^{d-1} := (\tilde{t}'_i)_{i=0}^{d-1} = (\tilde{t}''_i)_{i=0}^{d-1}.$$

Let us confirm $(\tilde{t}_i)_{i=0}^{d-1}$. From the verification, we have

$$\begin{aligned}(\widetilde{cmt}_0, \ldots, \widetilde{cmt}_{d-1})_{\ll c'} + (\tilde{t}_0, \ldots, \tilde{t}_{d-1}) &= s'_m\tilde{g} + s'_r\tilde{h}\\ (\widetilde{cmt}_0, \ldots, \widetilde{cmt}_{d-1})_{\ll c''} + (\tilde{t}_0, \ldots, \tilde{t}_{d-1}) &= s''_m\tilde{g} + s''_r\tilde{h}.\end{aligned}$$

From the subtraction between these two equations, we have

$$(\widetilde{cmt}_0, \ldots, \widetilde{cmt}_{d-1})(X^{c'} - X^{c''}) = (s'_m - s''_m)\tilde{g} + (s'_r - s''_r)\tilde{h}.$$

Multiplying by $2(X^{c'} - X^{c''})^{-1}$ to this equation, we have

$$(\widetilde{cmt}_0, \ldots, \widetilde{cmt}_{d-1}) = \frac{2(s'_m - s''_m)}{X^{c'} - X^{c''}}\tilde{g} + \frac{2(s'_r - s''_r)}{X^{c'} - X^{c''}}\tilde{h} =: 2\hat{m}\tilde{g} + 2\hat{r}\tilde{h}.$$

Then, let us confirm the upper bound of the norm $|2\hat{m}|$. By adopting Lemma 2.5 and 2.6, and the triangle inequality $|a \pm b| \leq |a| + |b|$, we have

$$\begin{aligned}|2\hat{m}| &\leq \sqrt{d} \cdot |s'_m - s''_m| \cdot \left|\frac{2}{X^{c'} - X^{c''}}\right|\\ &\leq \sqrt{d} \cdot |s'_m - s''_m| \cdot \sqrt{d}\\ &\leq d(|s'_m| + |s''_m|) = 2d^2.\end{aligned}$$

We can have a similar discussion with regard to t_1 and t_2. Let us confirm t_1. From

the verification equation, we have

$$X^{c'}c_1 + t_1 = bs'_v + ps'_e + m'$$
$$X^{c''}c_1 + t_1 = bs''_v + ps''_e + m''.$$

From the subtraction between these two equations, we obtain

$$(X^{c'} - X^{c''})c_1 = b(s'_v - s''_v) + p(s'_e - s''_e) + (m' - m'').$$

Multiplying by $2(X^{c'} - X^{c''})^{-1}$ to the equation,

$$2c_1 = b\frac{2(s'_v - s''_v)}{X^{c'} - X^{c''}} + p\frac{2(s'_e - s''_e)}{X^{c'} - X^{c''}} + \frac{2(s'_m - s''_m)}{X^{c'} - X^{c''}} =: 2b\hat{v} + 2p\hat{e} + 2\hat{m}.$$

From the similar discussion in the upper bound of $|2\hat{m}|$, we obtain

$$\begin{aligned} |2\hat{v}| &\leq \sqrt{d} \cdot |s'_v - s''_v| \cdot \left| \frac{2}{X^{c'} - X^{c''}} \right| \\ &\leq d(|s'_v| + |s''_v|) \\ &\leq \tilde{\mathcal{O}}(d^2\alpha). \end{aligned}$$

Since a similar discussion also holds for $\hat{e}$, we have $|2\hat{e}| \leq \tilde{\mathcal{O}}(d^2\alpha)$.

In addition, dealing with the verification equation regarding t_2 in a similar way of t_1, we obtain

$$2c_2 = a\frac{2(s'_v - s''_v)}{X^{c'} - X^{c''}} + p\frac{2(s'_f - s''_f)}{X^{c'} - X^{c''}} =: 2a\hat{v} + 2p\hat{f}.$$

In the same way, we obtain $|2\hat{f}| \leq \tilde{\mathcal{O}}(d^2\alpha)$.

High-entropy commitments. This property directly follows from the security of the auxiliary commitment scheme. □

Chapter 3

Traceability in permissioned blockchain

This chapter is based on the below works with Akira Otsuka.

[**MO19**] T. Mitani and A. Otsuka, "Traceability in Permissioned Blockchain," in 2019 IEEE International Conference on Blockchain, Atlanta, GA, USA, 2019, pp. 286-293.

[**MO20c**] T. Mitani and A. Otsuka, "Traceability in Permissioned Blockchain," in IEEE Access, vol. 8, pp. 21573-21588, 2020. (the extended version of [MO19])

3.1 Introduction

Blockchain attracts attention in commercial activities. It is useful in various situations such as trade finance, fund procurement, contract management and execution, and supply chain management. In commercial activities, corporations want to protect trade privacy by keeping it secret. Therefore, they often use permissioned blockchain with authorized participants. On the other hand, they disclose various types of information in order to fulfill their social responsibilities. For example, transparency of accounting is necessary for proper tax payment and information provision to investors. Tracking objects in supply chain management is also essential. Transparency is vital for corporate activities.

In this way, using blockchain in corporate activities, it is crucial to balance trade privacy and transparency. We request the traceability to balance these two properties.

First, we pay attention to trade privacy. The transaction information, including price and volume, is closely related to corporate activities. A bank generally handles account balances and transaction information as a secret since these are client privacy information. It is natural for the permissioned blockchain to handle the transaction information as a secret to an external party. Then, under the protection of the trade privacy, we consider two properties that consist of transparency. We note that the transparency holds obviously if all transactions are public.

Preservation. An outsider can verify that there is no dishonest increase and decrease inside the permissioned blockchain.

Noninvolvement. If there is a defect in the traded goods, the permissioned blockchain can disclose the participants who participated in the trade as needed later.

How can we balance trade privacy and transparency for any outsiders other than participants? We model traceability based on the hidden Markov model. We encrypt this model by homomorphic encryption. The proof of traceability requires the calculation of more than quadratic degrees. The establishment of the original model is verifiable by applying the non-interactive zero-knowledge proof of the knowledge that the plaintext is equal to zero.

Our result

This work provides the basic technology to achieve meaningful traceability in the blockchain that balances transactions' privacy and transparency. Let us describe the overview, threat model, and scenario regarding our result.

Overview

In our solution, we model the trading dynamics in the permissioned blockchain. We realize trade privacy by encrypting this model with fully homomorphic encryption. We also realize the transparency by proving the encrypted model equations with the zero-knowledge proof of plaintext knowledge. The calculations for traceability require more than quadratic degrees. Since it is not 2-DNF, we cannot use the traditional method [BGN05]. Therefore, we use fully homomorphic encryption. Furthermore, we adapt the work of Benhamouda *et al.* in Asiacrypt 2014 [BCK$^+$14]. We prove the encrypted model's

establishment by the zero-knowledge proof of knowledge in which plaintext is zero. We introduce the three properties of traceability concretely as follows.

Trade privacy Trade privacy is privacy regarding who trades with whom and at what asset amount in the permissioned blockchain. We denote $\vec{x}(t), \vec{u}(t)$ and $\vec{y}(t)$ as a vector of the participants' states in the permissioned blockchain, a vector of inputs and outputs in the permissioned blockchain, respectively. Let $A(t), B(t)$ and $C(t)$ be matrices of the "distribution ratios". $\vec{x}(t), A(t), B(t)$ and $C(t)$ indicate who trades with whom and at what asset amount. These are private information. $\vec{u}(t)$ and $\vec{y}(t)$ are public information. We express the following equations as a hidden Markov model.

$$\vec{x}(t+1) = \vec{x}(t)A(t) + \vec{u}(t)B(t)$$
$$\vec{y}(t+1) = \vec{x}(t)C(t)$$

Moreover, we mention the security of trade privacy. To protect the trade privacy, the private information $A(t), B(t), C(t)$ and $\vec{x}$ must be in hiding. Secure trade privacy means that no probabilistic polynomial time algorithm can distinguish if the distribution ratios $A(t), B(t), C(t)$ and the internal state $\vec{x}(t)$ were encrypted with honest plaintexts or just zeros (false plaintexts). As the $A(t), B(t), C(t)$ and $\vec{x}(t)$ must be private information, this definition makes sense.

Transparency Transparency consists of preservation and noninvolvement.

Preservation the total amount of assets in the blockchain, including input and output, is immutable. That is,

$$\sum_{i=1}^{m_p} x_i(t+1) + \sum_{i=1}^{n_p} y_i(t+1) = \sum_{i=1}^{m_p} x_i(t) + \sum_{i=1}^{l_p} u_i(t)$$

under the condition that

$$\sum_{j=1}^{m_p} a_{ij} + \sum_{j=1}^{n_p} c_{ij} = 1, \text{ for } i = \{1, 2, \ldots, m_p\}$$
$$\sum_{j=1}^{m_p} b_{ij} = 1, \text{ for } i = \{1, 2, \ldots, l_p\}$$

Noninvolvement One can reveal that some blockchain participants were not involved in a series of transactions. For some participant i,

$$a_{ij} = c_{ij} = 0, \text{ for all } j \neq i,$$
$$a_{ji} = c_{ji} = 0, \text{ for all } j \neq i,$$
$$b_{ji} = 0, \text{ for all } j.$$

We encrypt all the above equations with the RLWE encryption. The fact that $f(x) = y$ holds in plaintext means that the ciphertext of $f(x) - y$ is a ciphertext of zero. We suppose that a prover is a permissioned blockchain, and a verifier is a permissionless blockchain. The permissioned blockchain shows that the model is correct to the permissionless blockchain while concealing the plaintexts.

Threat model

We assume an adversary outside the blockchain as a threat. Our solution protects the privacy of the blockchain inside information from an actively external adversary. They can intervene in transactions between the permissioned blockchain and the permissionless blockchain. Their ability is computationally limited. Their goal is to break the trade privacy and create fake transactions and proofs. We also suppose the universal composable security [Can01]. On the other hand, we assume that the permissioned blockchain participants are reliable as honest insiders, and the adversary cannot corrupt them. It is why the distributed ledger of the permissioned blockchain contains the content that participants want to keep secret from outside third parties. Participants trust each other and share the secret in the permissioned blockchain.

Scenario

People trade agricultural and fishery products worldwide. To prevent the falsification of transaction tracking, the use of blockchain is expanding in corporate activities. There are various cases: pork and mango [Kam18], palm oil [Hir18], coffee [TMSB18] and so on. For our solution example, we prove the traceability every 10 minutes for 100 vegetables in the permissioned blockchain consisting of 20,000 participants. The necessary traffic is

approximately 1 Mbps. We describe the details in Section 3.4. This work contributes to the realization of meaningful traceability that balances the privacy and transparency of transactions. It is different from the trivial traceability of only transaction tracking.

We describe how participants issue and verify the proof. Participants in the permissioned blockchain initially share the state information. Therefore, it is allowable that they share the randomness in encrypting it. They can attach a threshold signature to the proof of the ciphertext of zero as follows.

1. A participant chooses randomness and creates a ciphertext and proof.

2. The randomness, ciphertext, and proof are broadcast to the other participants within the permissioned blockchain and shared.

3. They verify them by recreating their ciphertext and proof from the received randomness and plaintext.

4. They attach a ring signature to the verified proof similar to the withdrawal of Dilly *et al.* [DPW+16].

Furthermore, participants make the proofs public in internet storage. For example, Amazon Web Service S3 would be good. Miners in the permissionless blockchain verify the proofs.

3.1.1 Related work

There is considerable interest in protecting privacy in blockchain. One of the approaches is mixing transactions, exemplified in some works [Max13a, Max13b, RMSK14, BNM+14]. The other is the use of highly cryptographic methods. Zerocoin [MGGR13], which is an extension of Bitcoin, is one of the initial proposals that provided unlinkability between individual Bitcoin transactions without introducing a trusted third party. Pinocchio coin [DFKP13] proposes to incorporate the zero-knowledge succinct non-interactive argument of knowledge (zk-SNARK) [GGPR13] into Zerocoin. Zerocash [SCG+14], which is the successor to Zerocoin, uses zk-SNARK as a zero-knowledge proof of arithmetic circuit satisfiability. Recently, zero-knowledge scalable transparent argument of knowledge (zk-STARK) [BSBHR18] and Bulletproofs [BBB+18] have been proposed as similar zero-

knowledge proofs. Both works mention an application to the blockchain. Thus, the zero-knowledge proof is a crucial tool for privacy protection in the blockchain.

Let us refer to some other approaches. The schemes blinding all unspent transaction output (UTXO) while maintaining the public verifiability that no trade produces or destroys coins have been proposed [Max15, PBF+19]. As an application of Hyperledger Fabric, there is a method of protecting the privacy within each channel by setting confidentiality boundaries [ACDCKK18].

The Markov chain often describes the dynamics of the blockchain. There are several kinds of studies on selfish mining about Bitcoin [NKMS16, GKKT16, Wüs16].

Regarding homomorphic encryption, there are some works for encoding integers into plaintext space. Regarding the NTRU encryption scheme [HPS98], Hoffstein and Silverman announced the new technique [HS01]. Regarding the RLWE encryption, Geihs *et al.* [GC15] applied the technique to the Brakerski-Vaikuntanathan (BV) scheme [BV11] and Chen *et al.* [CLPX18] used the technique to the Fan-Vercauteren (FV) scheme [FV12].

Concurrent work

Let us confirm the concurrent works from the viewpoint of traceability in Table 3.1. Dilly *et al.* have proposed the sidechain scheme between Bitcoin and the permissioned blockchain Liquid [DPW+16]. In this scheme, the permissioned blockchain's participants determine the transfer of assets between Bitcoin and the permissioned blockchain through their consensus. A fast and deterministic agreement in the permissioned blockchain realizes the high throughput of transactions. In Dilly *et al.* [DPW+16], the pool account to a permissionless blockchain is public. There are no unauthorized deposits or withdrawals in this account. Thus, preservation holds, although Dilly *et al.* do not specify this matter.

Poelstra *et al.* [PBF+19] mention that there is no unauthorized increase or decrease in coin history, and the number of coins traded is concealed. However, the sender and receiver must be public.

Zether [BAZB20] realized a payment with anonymous accounts at Ethereum. The balance between the sender and the receiver and the transfer amount can be kept confidential. There is no defined way to indicate noninvolvement later. Dummy participants, except the sender and receiver, must be involved in the transaction from the beginning. It is difficult for Zether to achieve noninvolvement and trade privacy simultaneously.

Compared with these works, this work is the first proposal that satisfies the three properties of traceability. It achieves both trade privacy and transparency simultaneously.

Table 3.1: Traceability's Properties. Zether achieves either noninvolvement or trade privacy.

	Preservation	Noninvolvement	Trade privacy
Dilly *et al.* [DPW+16]	yes	no	no
Poelstra *et al.* [PBF+19]	yes	yes	no
Zether [BAZB20]	yes	yes/no	no/yes
This work	yes	yes	yes

3.1.2 Chapter organization

We organize the rest of this chapter as follows. Section 3.2 defines traceability and describes its three properties. Section 3.3 encrypts the model defined in Section 3.2 and confirms the security. Section 3.4 introduces the ring isomorphism encoding, uses this encoding to the traceability model, and confirms the efficiency. Finally, Section 3.5 concludes this work.

3.2 Traceability and modeling

We introduce traceability. We confirm its three properties: trade privacy, preservation, and noninvolvement.

3.2.1 Traceability

According to the review paper [BG13], various researchers define traceability in various ways, none of which are perfect. Some researchers define the flow of products from suppliers to users and vice versa. Many works seem to define traceability by focusing on the "objects" moving between players. In contrast, we focus on how the "state" of each player's asset amount changes from time to next time.

Let us consider the mathematical definition based on this idea. The state means that the amount of an asset of the participant. We represent the state of participants in the permissioned blockchain as the vector $\vec{x}(t)$. That is,

$$\vec{x}(t) = (x_1, x_2, \ldots, x_{m_p}) \in \mathbb{P}^{m_p}.$$

Let us explain each item in this equation. t indicates that the state is at time t. m_p is the number of participants. x_i is the state of the participant i. The field $\mathbb{P} \subset \mathbb{Q}$.

Similarly, we represent the state from the permissionless blockchain to the permissioned blockchain as the vector $\vec{u}$. We also express the state from the permissioned blockchain to the permissionless blockchain as the vector $\vec{y}$. That is,

$$\vec{u} = (u_1, u_2, \ldots, u_{l_p}) \in \mathbb{P}^{l_p}$$
$$\vec{y} = (y_1, y_2, \ldots, y_{n_p}) \in \mathbb{P}^{n_p}.$$

l_p and n_p are the numbers of the accounts.

We formally describe that the state transition is deterministic as follows.

Definition 3.1. *Let $\vec{x}(t), \vec{x}(t+1)$ be the amount of assets in the permissioned blockchain at each time $t, t+1$, respectively. Let $\vec{y}(t+1)$ be the amount of assets moving from the permissioned blockchain to the permissionless blockchain at time $t+1$. Let $\vec{u}(t)$ be the amount of assets moving from the permissionless blockchain to the permissioned blockchain at time t. If $x(0)$ and $u(0)$ are determined at the initial time $t = 0$, then for every time $t \geq 0$, there exists some deterministic polynomial time algorithm $\mathcal{A}$ such that*

$$(\vec{x}(t+1), \vec{y}(t+1)) = \mathcal{A}(\vec{x}(t), \vec{u}(t)).$$

3.2.2 Modeling

We discuss the deterministic polynomial time algorithm specifically. We list the model parameters in Table 3.2.

Table 3.2: Overview of parameters in the model

Parameter	Explanation
$\vec{x}$	account balance of each member in permissioned blockchain
	quantity of each member in permissioned blockchain
$\vec{u}, \vec{y}$	pool account balance in permissionless blockchain
a_{ij}	remittance between members in permissioned blockchain
b_{ij}	deposits and products for permissioned blockchain
c_{ij}	withdrawal and consumption from permissioned blockchain

$\vec{x}$ is information in the permissioned blockchain. A participant in the permissioned blockchain can observe $\vec{x}$. However, any participant in the permissionless blockchain cannot. $\vec{u}$ and $\vec{y}$ are information in the permissionless blockchain. Everyone can observe

them. We call it a hidden Markov model as $\vec{x}$ is a latent variable for an outsider.

We confirm the state transition. We suppose that the total volume moving from $x_i(t)$ to $x_j(t+1)$ is v. That is,

$$x_i(t+1) = x_i(t) - v$$
$$x_j(t+1) = x_j(t) + v$$

Let us define the distribution rate $a_{ij} = v/x_i(t)$. In particular, a_{ii} is the "staying" rate ($x_i(t) \to x_i(t+1)$). In same way, we define the distribution rate $c_{ij} = v/x_i(t)$. The amount $x_i(t)$ is distributed to $x_1(t+1), \ldots, x_i(t+1), \ldots, x_{m_p}(t+1)$ at the ratio of $a_{i1}, \ldots, a_{ii}, \ldots, a_{im_p}$ and distributed to $y_1(t+1), \ldots, y_{n_p}(t+1)$ at the ratio of $c_{i1}, \ldots, c_{in_p}$. The sum of all the ratios must be equal to 1. That is,

$$\sum_{j=1}^{m_p} a_{ij} + \sum_{j=1}^{n_p} c_{ij} = 1.$$

We also define the distribution rate $b_{ij} = v/u_i(t)$ moving from $u_i(t)$ to $x_j(t+1)$. u_i is distributed to $x_1(t+1), \ldots, x_{m_p}(t+1)$. The sum of all these ratios must be equal to 1. That is,

$$\sum_{j=1}^{m_p} b_{ij} = 1.$$

From the above discussion, we can obtain the definitions of trade privacy and preservation.

Definition 3.2 (Trade privacy)**.** *Let* $A = (a_{ij}) \in \mathbb{P}^{m_p \times m_p}, B = (b_{ij}) \in \mathbb{P}^{l_p \times m_p}$ *and* $C = (c_{ij}) \in \mathbb{P}^{m_p \times n_p}$*, respectively. The relation of trade privacy is expressed as*

$$\vec{x}(t+1) = \vec{x}(t)A(t) + \vec{u}(t)B(t)$$
$$\vec{y}(t+1) = \vec{x}(t)C(t)$$

Definition 3.3 (Preservation)**.** *Let* a_{ij}, b_{ij} *and* c_{ij} *be a element of the matrices* A, B *and*

C, respectively. Preservation holds if the following equations hold.

$$\sum_{j=1}^{m_p} a_{ij} + \sum_{j=1}^{n_p} c_{ij} = 1, \ \textit{for } i = \{1, 2, \ldots, m_p\}$$
$$\sum_{j=1}^{m_p} b_{ij} = 1, \ \textit{for } i = \{1, 2, \ldots, l_p\}$$

We confirm the following lemma with regard to Definition 3.2, 3.3.

Lemma 3.1 (Overall preservation)**.** *If Definition 3.2 and 3.3 are satisfied, then the equation*

$$\sum_{i=1}^{m_p} x_i(t+1) + \sum_{i=1}^{n_p} y_i(t+1) = \sum_{i=1}^{m_p} x_i(t) + \sum_{i=1}^{l_p} u_i(t)$$

holds.

Proof. First, we expand the equations of Definition 3.2. For our convenience, we omit "(t)" such as $\vec{x}(t)$ or $A(t)$ in this proof. The same abbreviation applies to elements of vectors and matrices. Applying $\sum_{j=1}^{m_p} b_{ij} = 1$, we obtain

$$\begin{aligned}
\sum_{j=1}^{m_p} x_j(t+1) &= \sum_{j=1}^{m_p}(\sum_{i=1}^{m_p} x_i a_{ij}) + \sum_{j=1}^{m_p}(\sum_{i=1}^{l_p} u_i b_{ij}) \\
&= \sum_{i=1}^{m_p} x_i(\sum_{j=1}^{m_p} a_{ij}) + \sum_{i=1}^{l_p} u_i(\sum_{j=1}^{m_p} b_{ij}) \\
&= \sum_{i=1}^{m_p} x_i(\sum_{j=1}^{m_p} a_{ij}) + \sum_{i=1}^{l_p} u_i
\end{aligned}$$

We also obtain

$$\sum_{j=1}^{n_p} y_j(t+1) = \sum_{j=1}^{n_p}(\sum_{i=1}^{m_p} x_i c_{ij}) = \sum_{i=1}^{m_p} x_i(\sum_{j=1}^{n_p} c_{ij})$$

Applying $\sum_{j=1}^{m_p} a_{ij} + \sum_{j=1}^{n_p} c_{ij} = 1$, we finally obtain

$$\begin{aligned}
&\sum_{j=1}^{m_p} x_j(t+1) + \sum_{j=1}^{n_p} y_j(t+1) \\
&= (\sum_{i=1}^{m_p} x_i(\sum_{j=1}^{m_p} a_{ij}) + \sum_{i=1}^{l_p} u_i) + \sum_{i=1}^{m_p} x_i(\sum_{j=1}^{n_p} c_{ij}) \\
&= \sum_{i=1}^{m_p} x_i(\sum_{j=1}^{m_p} a_{ij} + \sum_{j=1}^{n_p} c_{ij}) + \sum_{i=1}^{l_p} u_i \\
&= \sum_{i=1}^{m_p} x_i + \sum_{i=1}^{l_p} u_i
\end{aligned}$$

The equation in the lemma holds. □

Plain example. We describe the simple example. We suppose that Alice, Bob, and Carol are participants in the permissioned blockchain. They have \$2, \$3, and \$4 in each account, respectively. We formulate this situation as follows.

$$\vec{x}(t) = (x_1(t), x_2(t), x_3(t)) = (2, 3, 4)$$

We suppose that they execute the following three transactions during $[t, t+1]$.

- Alice gives Bob \$1.
- Bob gives Carol \$2.
- Carol gives Alice \$3.

We assume that there is neither deposit nor withdrawal between the permissioned blockchain and permissionless blockchain. We consider only internal transactions within the permissioned blockchain. Their accounts are represented by the following state.

$$\vec{x}(t+1) = (x_1(t+1), x_2(t+1), x_3(t+1)) = (4, 2, 3)$$

We represent the increase or decrease in the asset quantity by each transaction as the

ratio,

$$x_1(t+1) = x_1(t) - \frac{1}{2}x_1(t) + \frac{3}{4}x_3(t) = \frac{1}{2}x_1(t) + \frac{3}{4}x_3(t)$$
$$x_2(t+1) = x_2(t) + \frac{1}{2}x_1(t) - \frac{2}{3}x_2(t) = \frac{1}{2}x_1(t) + \frac{1}{3}x_2(t)$$
$$x_3(t+1) = x_3(t) - \frac{3}{4}x_3(t) + \frac{2}{3}x_2(t) = \frac{2}{3}x_2(t) + \frac{1}{4}x_3(t)$$

We obtain $\vec{x}(t+1) = \vec{x}(t)A(t)$, where

$$A(t) = \begin{pmatrix} 1/2 & 1/2 & 0 \\ 0 & 1/3 & 2/3 \\ 3/4 & 0 & 1/4 \end{pmatrix}$$

Then, $\sum_j a_{ij} = 1$ holds.

We define noninvolvement and describe local preservation.

Definition 3.4 (Noninvolvement)**.** *Regarding Definition 3.2, noninvolvement of a participant i holds at $[t, t+1]$, if and only if*

$$a_{ij} = c_{ij} = 0, \text{ for all } j \neq i,$$
$$a_{ji} = c_{ji} = 0, \text{ for all } j \neq i,$$
$$b_{ji} = 0, \text{ for all } j.$$

Definition 3.5 (Subset of involvement)**.** *S_r is a subset of involvement such that the subset excludes noninvolvement participants from the set of blockchain participants $\{1, \ldots, m_p\}$.*

Lemma 3.2 (Local preservation)**.** *If Definition 3.2 and 3.3 are satisfied and S_r is a subset of involvement, then the equation*

$$\sum_{i \in S_r} x_i(t+1) + \sum_{i=1}^{n_p} y_i(t+1) = \sum_{i \in S_r} x_i(t) + \sum_{i=1}^{l_p} u_i(t)$$

holds.

Proof. From Lemma 3.1,

$$\sum_{i=1}^{m_p} x_i(t+1) + \sum_{i=1}^{n_p} y_i(t+1) = \sum_{i=1}^{m_p} x_i(t) + \sum_{i=1}^{l_p} u_i(t)$$

holds. Since every participant belongs to S_r or not,

$$\sum_{i=1}^{m_p} x_i(t+1) = \sum_{i \in S_r} x_i(t+1) + \sum_{i \notin S_r} x_i(t+1)$$
$$\sum_{i=1}^{m_p} x_i(t) = \sum_{i \in S_r} x_i(t) + \sum_{i \notin S_r} x_i(t)$$

As any participant $i \notin S_r$ is represented by noninvolvement, x_i is invariant. That is,

$$\sum_{i \notin S_r} x_i(t+1) = \sum_{i \notin S_r} x_i(t).$$

Therefore, we obtain

$$\sum_{i \in S_r} x_i(t+1) + \sum_{i=1}^{n_p} y_i(t+1) = \sum_{i \in S_r} x_i(t) + \sum_{i=1}^{l_p} u_i(t)$$

□

We denote the three properties Definition 3.2, 3.3, and 3.4 as the "traceability model" as follows.

Definition 3.6 (Traceability model)**.**

Trade privacy

$$x(t)A(t) + u(t)B(t) - x(t+1) = 0$$
$$x(t)C(t) - y(t+1) = 0$$

Preservation

$$\sum_{j=1}^{m_p} a_{ij} + \sum_{j=1}^{n_p} c_{ij} - 1 = 0, \ \textit{for } i = \{1, 2, \ldots, m_p\}$$
$$\sum_{j=1}^{m_p} b_{ij} - 1 = 0, \ \textit{for } i = \{1, 2, \ldots, l_p\}$$

Noninvolvement *For some i,*

$$a_{ij} = c_{ij} = 0, \ \textit{for all } j \neq i,$$
$$a_{ji} = c_{ji} = 0, \ \textit{for all } j \neq i,$$
$$b_{ji} = 0, \ \textit{for all } j.$$

Remark 3.1. *One hidden Markov model corresponds to one kind of object (apple, banana, coin, and so on). If we deal with multiple objects, we can evaluate each model for each object. We also make its creation and consumption correspond to the input and the output to the permissioned blockchain.*

3.3 Encrypted model and its security

To conceal the state in the permissioned blockchain, we intend to encrypt the traceability model of Definition 3.6 with the RLWE encryption. In this section, we introduce the encrypted traceability model and describe its security.

3.3.1 Encrypted traceability model

We confirm that the traceability model of Definition 3.6 corresponds to the ciphertext of zero. We define the set $\mathbf{C}_0$ as a set of ciphertext obtained by encrypting plaintext $m = 0$ as follows.

Definition 3.7 (Ciphertext of zero)**.**

$$\mathbf{C}_0 = \{(c_1, c_2) \mid c_1, c_2 \in \mathbf{R}_q \wedge \ ((c_1 - s \cdot c_2 \mod q) \mod p) = 0\}.$$

s is a secret key.

We change the traceability model of Definition 3.6 into implicit functions f.

Trade privacy

$$f_1 : \vec{x}(t)A(t) + \vec{u}(t)B(t) - \vec{x}(t+1)$$
$$f_2 : \vec{x}(t)C(t) - \vec{y}(t+1)$$

Preservation

$$f_3^i : \sum_{j=1}^{m_p} a_{ij} + \sum_{j=1}^{n_p} c_{ij} - 1, \quad \text{for} \quad i = \{1, 2, \ldots, m_p\},$$
$$f_4^i : \sum_{j=1}^{m_p} b_{ij} - 1, \quad \text{for} \quad i = \{1, 2, \ldots, l_p\}$$

Noninvolvement a_{ij}, b_{ij} and c_{ij} themselves correspond to f directly.

$\mathbb{P}$ includes all the elements of the traceability model of Definition 3.6. The above functions f_1, f_2, f_3^i and f_4^i are also closed by $\mathbb{P}$. An element in $\mathbb{P}$ corresponds to $\mathbf{R}_p$ by the ring isomorphic encoding. We will introduce the ring isomorphic encoding and its efficiency in Section 3.4. $\mathbf{R}_p$ corresponds to $\mathbf{R}_q^2$ with RLWE encryption. We denote the ciphertext $\mathbf{a} \in \mathbf{R}_q^2$ corresponding to $a \in \mathbb{P}$. In this section, let us write the ciphertext in bold to distinguish it from plaintext. We represent the vectors and matrices obtained by encrypting each element in the same way. We denote encrypted vectors as $\vec{\mathbf{u}} = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_l), \vec{\mathbf{x}} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$, and $\vec{\mathbf{y}} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n)$. We denote encrypted matrices as $\mathbf{A} = (\mathbf{a}_{ij}), \mathbf{B} = (\mathbf{b}_{ij})$ and $\mathbf{C} = (\mathbf{c}_{ij})$. Therefore, we denote the function $\mathbf{f}$ for ciphertexts in $\mathbf{R}_q^2$ corresponding to the function f for plaintexts in $\mathbb{P}$. $\mathbf{f}_1, \mathbf{f}_2$ correspond to Definition 3.2 and $\mathbf{f}_3^i, \mathbf{f}_4^i$ correspond to Definition 3.3. For some i, $\mathbf{a}_{ij}, \mathbf{b}_{ij}$ and $\mathbf{c}_{ij}$ themselves correspond to Definition 3.4. We obtain encrypted functions as follows.

Definition 3.8 (Encrypted traceability model)**.**

Trade privacy

$$\mathbf{f}_1(\vec{\mathbf{x}}(t), \mathbf{A}(t), \vec{\mathbf{u}}, \mathbf{B}(t), \vec{\mathbf{x}}(t+1)) \in \mathbf{C}_0^{m_p}$$
$$\mathbf{f}_2(\vec{\mathbf{x}}(t), \mathbf{C}(t), \vec{\mathbf{y}}(t+1)) \in \mathbf{C}_0^{n_p}$$

Preservation

$$\mathbf{f}_3^i(\mathbf{a}_{ij}, \mathbf{c}_{ij}) \in \mathbf{C}_0 \text{ for } i = \{1, 2, \ldots, m_p\}$$
$$\mathbf{f}_4^i(\mathbf{b}_{ij}) \in \mathbf{C}_0 \text{ for } i = \{1, 2, \ldots, l_p\}$$

Noninvolvement *For some i,*

$$\mathbf{a}_{ij}, \mathbf{c}_{ij} \in \mathbf{C}_0 \text{ for all } j \neq i,$$
$$\mathbf{a}_{ji}, \mathbf{c}_{ji} \in \mathbf{C}_0 \text{ for all } j \neq i,$$
$$\mathbf{b}_{ji} \in \mathbf{C}_0 \text{ for all } j.$$

Remark 3.2. *We write a ciphertext and its function in bold. It is why to distinguish from plaintext and its function as the above description.*

3.3.2 Security of trade privacy

We confirm the security of the trade-privacy equations $\mathbf{f}_1, \mathbf{f}_2$. We intend to hide the private information $A(t), B(t), C(t)$ and $x(t)$ in these equations. We consider the trade-privacy equations calculated by ciphertexts of honest plaintexts and fake plaintexts (all zeros). We say that trade privacy is secure if and only if both are indistinguishable for an adversary. Since the trade-privacy equations $\mathbf{f}_1, \mathbf{f}_2$ consist of inner products, we propose the security challenge experiment in Fig. 3.1.

Theorem 3.1 (Security of trade privacy)**.** *For any $\lambda \in \mathbb{N}$ and any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage*

$$\mathrm{Adv}^{\mathsf{tp-ind}}_{\mathsf{TP},\mathcal{A}}(\lambda) := \mid \Pr[\mathrm{Exp}^{\mathsf{tp-ind}}_{\mathsf{TP},\mathcal{A}}(\lambda) = 1] - 1/2 \mid$$

is negligible in λ, where $\mathrm{Exp}^{\mathsf{tp-ind}}_{\mathsf{TP},\mathcal{A}}(\lambda)$ is as defined in Fig. 3.1.

$\underline{\mathrm{Exp}^{\mathsf{tp-ind}}_{\mathsf{TP},\mathcal{A}}(\lambda):}$
$(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{RLWE.Gen}(1^\lambda)$
$\beta \leftarrow \{0,1\}$
$\beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{TP}}(\cdot,\cdot)}(1^\lambda, \mathsf{pk})$
if $\beta = \beta'$ return 1
else return 0

$\underline{\mathcal{O}_{\mathsf{TP}}(c_i, x_i):}$
if $\beta = 0$
  $\mathbf{c}_i \leftarrow \mathsf{RLWE.Enc}(\mathsf{pk}, c_i)$
  $\mathbf{x}_i \leftarrow \mathsf{RLWE.Enc}(\mathsf{pk}, x_i)$
else
  $\mathbf{c}_i \leftarrow \mathsf{RLWE.Enc}(\mathsf{pk}, 0)$
  $\mathbf{x}_i \leftarrow \mathsf{RLWE.Enc}(\mathsf{pk}, 0)$
$\mathbf{y} := \sum_i \mathbf{c}_i \cdot \mathbf{x}_i$
return $\mathbf{y}$

Figure 3.1: Security challenge experiment for trade privacy

Proof. To prove the security of trade privacy, we consider the games shown in Fig. 3.2. We denote events as follows:

$\underline{\mathrm{G}^0_{\mathsf{TP},\mathcal{A}}(\lambda):}$
$(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{RLWE.Gen}(1^\lambda)$
$\beta \leftarrow \{0,1\}$
$\beta' \leftarrow \mathcal{A}^{\mathcal{O}^0_{\mathsf{TP}}(\cdot,\cdot)}(1^\lambda, \mathsf{pk})$
if $\beta = \beta'$ return 1
else return 0

$\underline{\mathcal{O}^0_{\mathsf{TP}}(c_i, x_i):}$
if $\beta = 0$
  $\mathbf{c}_i \leftarrow \mathsf{RLWE.Enc}(\mathsf{pk}, c_i)$
  $\mathbf{x}_i \leftarrow \mathsf{RLWE.Enc}(\mathsf{pk}, x_i)$
else
  $\mathbf{c}_i \leftarrow \mathsf{RLWE.Enc}(\mathsf{pk}, 0)$
  $\mathbf{x}_i \leftarrow \mathsf{RLWE.Enc}(\mathsf{pk}, 0)$
$\mathbf{y} := \sum_i \mathbf{c}_i \cdot \mathbf{x}_i$
return $\mathbf{y}$

$\underline{\mathrm{G}^1_{\mathsf{TP},\mathcal{A}}(\lambda):}$
$(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{RLWE.Gen}(1^\lambda)$
$\beta \leftarrow \{0,1\}$
$\beta' \leftarrow \mathcal{A}^{\mathcal{O}^1_{\mathsf{TP}}(\cdot,\cdot)}(1^\lambda, \mathsf{pk})$
if $\beta = \beta'$ return 1
else return 0

$\underline{\mathcal{O}^1_{\mathsf{TP}}(c_i, x_i):}$
if $\beta = 0$
  $\mathbf{c}_i, \mathbf{x}_i \xleftarrow{\$} \mathbf{R}_q^2$
else
  $\mathbf{c}_i, \mathbf{x}_i \xleftarrow{\$} \mathbf{R}_q^2$
$\mathbf{y} := \sum_i \mathbf{c}_i \cdot \mathbf{x}_i$
return $\mathbf{y}$

Figure 3.2: Games $\mathrm{G}^0_{\mathsf{TP},\mathcal{A}}(\lambda)$ and $\mathrm{G}^1_{\mathsf{TP},\mathcal{A}}(\lambda)$ in the proof of Theorem 3.1.

- $E_0 : \mathrm{G}^0_{\mathsf{TP},\mathcal{A}}(\lambda) = 1$
- $E_1 : \mathrm{G}^1_{\mathsf{TP},\mathcal{A}}(\lambda) = 1$
- $B_0 : \beta = 0$
- $B_1 : \beta = 1$

We have

$$\begin{aligned}\Pr[E_0] &= \Pr[E_0 \cap B_0] + \Pr[E_0 \cap B_1] \\ &= \Pr[E_0|B_0] \cdot \Pr[B_0] + \Pr[E_0|B_1] \cdot \Pr[B_1].\end{aligned}$$

In the same way, we obtain

$$\Pr[E_1] = \Pr[E_1|B_0] \cdot \Pr[B_0] + \Pr[E_1|B_1] \cdot \Pr[B_1].$$

From the these equations, we construct an adversary $\mathcal{B}$ from an adversary $\mathcal{A}$ who can distinguish $\mathrm{G}^0_{\mathsf{TP},\mathcal{A}}(\lambda)$ and $\mathrm{G}^1_{\mathsf{TP},\mathcal{A}}(\lambda)$. That is,

$$\begin{aligned}&| \Pr[\mathrm{G}^0_{\mathsf{TP},\mathcal{A}}(\lambda) = 1] - \Pr[\mathrm{G}^1_{\mathsf{TP},\mathcal{A}}(\lambda) = 1] | \\ =&| \Pr[E_0] - \Pr[E_1] | \\ =&| \Pr[B_0] \cdot (\Pr[E_0|B_0] - \Pr[E_1|B_0]) + \Pr[B_1] \cdot (\Pr[E_0|B_1] - \Pr[E_1|B_1]) | \\ \leq& \Pr[B_0] \cdot | \Pr[E_0|B_0] - \Pr[E_1|B_0] | + \Pr[B_1] \cdot | \Pr[E_0|B_1] - \Pr[E_1|B_1] | \\ =& (\Pr[B_0] + \Pr[B_1]) \cdot \mathrm{Adv}^{\mathsf{pr}}_{\mathsf{RLWE},\mathcal{B}}(\lambda) \\ =& \mathrm{Adv}^{\mathsf{pr}}_{\mathsf{RLWE},\mathcal{B}}(\lambda)\end{aligned}$$

where we applied $\Pr[B_0] + \Pr[B_1] = 1$ in the last equation.

In game $\mathrm{G}^1_{\mathsf{TP},\mathcal{A}}(\lambda)$ the view of $\mathcal{A}$ is independent of β, as the $\mathbf{c}_i, \mathbf{x}_i$ and $\mathbf{y}$ are chosen uniformly at random over $\mathbf{R}_q^2$. We have thus

$$\Pr[\mathrm{G}^1_{\mathsf{TP},\mathcal{A}}(\lambda) = 1] = 1/2.$$

Moreover, $\mathrm{G}^0_{\mathsf{TP},\mathcal{A}}(\lambda)$ is $\mathrm{Exp}^{\mathsf{tp-ind}}_{\mathsf{TP},\mathcal{A}}(\lambda)$ itself. Because of Definition 2.4, we finally obtain that

$$\mathrm{Adv}^{\mathsf{tp-ind}}_{\mathsf{TP},\mathcal{A}}(\lambda) \leq \mathrm{Adv}^{\mathsf{pr}}_{\mathsf{RLWE},\mathcal{B}}(\lambda) < \mathrm{negl}(\lambda).$$

□

3.3.3 Non-interactive zero-knowledge proof

The permissioned and permissionless blockchains do not move synchronously. The non-interactive zero-knowledge proof between blockchains is desirable. By using Theorem 2.1, we can prove with zero-knowledge that the encrypted model of Definition 3.8 corresponds to the model of Definition 3.6.

3.4 Encoding and its efficiency

We need to associate the rational number field $\mathbb{P}$ of the traceability model with the plaintext space $\mathbf{M}$ of the RLWE encryption. For the purpose, we adopt the ring isomorphism encoding. In this section, we introduce the ring isomorphism encoding. Furthermore, we confirm the efficiency of applying it to the encrypted traceability model and the zero-knowledge proof.

3.4.1 Ring isomorphism encoding

We discuss the correspondence between $\mathbb{P}$ and the plaintext space $\mathbf{M} = \mathbf{R}_p = \mathbb{Z}[X]/\langle X^d + 1, p\rangle$. We consider that the rational numbers in $\mathbb{P}$ are fixed-point numbers. A fixed-point number is a number with a fixed number of digits. For example, we can handle 8.22 or 82.2 as the integer 822. We can map fixed-point numbers to integers. We define $\mathbb{P}$ as follows.

$$\mathbb{P} = \{(n, d) \mid n = \{-(2^{n_1-1} - 1), \ldots, 2^{n_1-1} - 1\} \subset \mathbb{Z}, d = 2^{n_2}\}, \quad \text{for } \exists n_1, n_2 \in \mathbb{N}.$$

We introduce the efficient encoding and decoding, the ring isomorphism encoding [GC15].

Definition 3.9 (Ring isomorphism encoding (Section 3.2 in [GC15]))**.**

- *The encoding of an integer $z \in \mathbb{Z}$ with $|z| \leq 2^{n-1}$ to a polynomial $m \in \mathbf{R}_p$ is*

$$m = \sum_{i=0}^{n-1} z_i \cdot x^i.$$

- *The decoding of a polynomial $m \in \mathbf{R}_p$ to an integer $z \in \mathbb{Z}$ is*

$$z = \begin{cases} z' - (2^n + 1) & \text{if } z' \geq 2^{n-1} \ , \\ z' & \text{otherwise,} \end{cases}$$

where

$$z' = \sum_{i=0}^{n-1} m_i \cdot 2^i \mod (2^n + 1).$$

Theorem 3.2 (Theorem 1 in [GC15])**.** *For $a \in \mathbb{Z}$, the map $\phi : \mathbb{Z}/\langle x^n + 1, x - a\rangle \to \mathbb{Z}/\langle a^n + 1\rangle$ given by*

$$f(x) + \langle x^n + 1, x - a\rangle \mapsto f(a) + \langle a^n + 1\rangle$$

is an isomorphism.

Letting $a = 2$ in Theorem 3.2, we obtain $\mathbf{R}_{x-2} \cong \mathbb{Z}_{2^n+1}$. Letting $n = n_1$, we choose the input integer space as $\mathbb{Z}/\langle 2^{n_1} - 1\rangle$. That is, the input integers exist in $\{-(2^{n_1-1} - 1), \ldots, 2^{n_1-1} - 1\}$. As $|z| \leq 2^{n_1-1}$, the decoding becomes simpler from Definition 3.9, that is, $z = z'$.

3.4.2 Somewhat homomorphic encryption

We show that somewhat homomorphic encryption is feasible for the traceability model of Definition 3.6. We define the benchmark function f as follows, considering the inner product in the traceability model of Definition 3.6.

$$f(z) = \sum_{j=1}^{l_{\text{add}}} \prod_{i=1}^{l_{\text{mul}}} z_{ij} \quad \text{for} \quad z_{ij} \in \mathbb{Z}.$$

z_{ij} corresponds to an element of the matrices and vectors of the traceability model. We confirm the range of integers. Since $|z_{ij}| \leq 2^{n_1-1}$,

$$2^{d-1} \geq \left| \sum_{j=1}^{l_{\text{add}}} \prod_{i=1}^{l_{\text{mul}}} z_{ij} \right| \geq l_{\text{add}} \cdot 2^{l_{\text{mul}}(n_1-1)}.$$

We obtain

$$d > \log_2 l_{\text{add}} + l_{\text{mul}}(n_1 - 1). \tag{3.1}$$

Let us confirm the decryptable noise magnitude. We confirm noise growth. Let $c = (c_1, c_2) \in \mathbf{R}_q^2$ and $c_1 - s \cdot c_2 = m + p \cdot e$. s is a secret key, m is a plaintext and e is a noise. We denote the norm as $\mathrm{DN}(c) = |m + p \cdot e|_\infty$. From the condition that we can decrypt $c \in \mathbf{R}_q^2$ correctly, we have $\mathrm{DN}(c) < q/2$. The relationship between the operations of ciphertexts and the noise growth are as follows.

- Let $c_1, c_2 \in \mathbf{R}_q^2, c = c_1 + c_2$. Then, we have $\mathrm{DN}(c) = \mathrm{DN}(c_1) + \mathrm{DN}(c_2)$.

- Let $c_1, c_2 \in \mathbf{R}_q^2, c = c_1 \cdot c_2$. Then, we have $\mathrm{DN}(c) \leq d \cdot \mathrm{DN}(c_1) \cdot \mathrm{DN}(c_2)$.

  We used $|s \cdot t|_\infty \leq d \cdot |s|_\infty \cdot |t|_\infty$ from Lemma 2.6.

The size of the noise is small. Let $|s|_\infty, |v|_\infty, |e|_\infty, |f|_\infty \leq B_\chi$.

According to [GC15], let B_{fresh} be the size of noise in no operation of ciphertexts. We have

$$B_{\mathrm{fresh}} = 1 + 3 \cdot (2dB_\chi^2 + B_\chi).$$

From the above equations, we can transform the condition $\mathrm{DN}(c) < q/2$ into

$$q/2 > l_{\mathrm{add}} B_{\mathrm{fresh}}^{l_{\mathrm{mul}}} d^{l_{\mathrm{mul}}-1}.$$

We obtain

$$\log_2 q > \log_2 l_{\mathrm{add}} + l_{\mathrm{mul}} \log_2(B_{\mathrm{fresh}} d) - \log_2 d + 1. \tag{3.2}$$

We confirm the parameters in the RLWE encryption. Following Table 1 in [NLV11], we choose the concrete parameters: $d = 2048, \log_2 q = 64$ and $\sigma = 8$. σ is the standard deviation of the discrete Gaussian distribution for the randomness. As our intention toward the traceability model of Definition 3.6, we choose $n_1 = 65$ which is a 64-bit integer and $l_{\mathrm{mul}} = 2$. Let $B_\chi = \sigma\sqrt{d} \approx 362$ from Lemma 2.1.

We confirm the upper limit of the number of additions l_{add} in the traceability model of Definition 3.6 satisfying the inequalities 3.1 and 3.2. We show the upper limits of the number of additions l_{add} corresponding to d and q in Table 3.3.

We can calculate the traceability model of Definition 3.6 with sufficiently numerous additions l_{add} under the practical parameters. That is, we can realize it as somewhat homomorphic encryption.

Table 3.3: Numbers of additions calculable with somewhat homomorphic encryption

d	$\lceil \log_2 q \rceil$	$\lfloor \log_2 l_{\mathrm{add}} \rfloor$
2048	89	15.8
4096	94	15.8
4096	120	41.8
8192	264	180.8
16384	423	334.8

3.4.3 Efficiency of zero-knowledge proof

The computation of the encrypted traceability model is limited to low degrees. We can achieve it with somewhat homomorphic encryption. In other words, bootstrapping is unnecessary. Therefore, the method is effectively practical for computation in the real world.

We confirm the proof size issued by the prover with the non-interactive zero-knowledge proof. There are multiple ciphertexts for which we want to prove that the plaintext is zero. There is an excellent way to reduce the traffic per ciphertext. We pack these ciphertexts into elements of one vector. The inner product of the vector is also one ciphertext. We apply the protocol in Fig. 2.3 to the ciphertext. We can reduce the proof size per ciphertext. An additional inner product calculation increases the degree by one. The addition increases by k_{add} times the number of equations we want to prove. Compared to the case where there is no inner product calculation, we need to enlarge the lattice dimension and the ciphertext space. In the same way, we can express the inequalities as follows.

$$d > \log_2 k_{\mathrm{add}} + 2(\log_2 l_{\mathrm{add}} + l_{\mathrm{mul}}(n_1 - 1))$$

$$\log_2 q - 1 > 2\log_2 l_{\mathrm{add}} + 2l_{\mathrm{mul}} \log_2 B_{\mathrm{fresh}} + (2l_{\mathrm{mul}} - 1)\log_2 d + \log_2 k_{\mathrm{add}}$$

In same way as shown in Table 3.3, we can also estimate the proof size in Table 3.4. We set the zero-knowledge proof count to $\lambda = 128$ for 128-bit security and $k_{\mathrm{add}} = l_{\mathrm{add}}$. The proof size is at most 627.7 kB.

We naively evaluate the traffic regarding the example of vegetables distributed in Japan. We state elements as follows.

Table 3.4: Proof size for 128-bit security

$l_{\mathrm{mul}} = 1$

d	$\lceil \log_2 q \rceil$	$\lfloor \log_2 l_{\mathrm{add}} \rfloor$	proof size (kB)
2048	89	5.2	103.7
4096	94	5.2	119.5
4096	120	13.9	152.6
8192	264	60.2	363.7
16384	423	111.6	627.7

$l_{\mathrm{mul}} = 2$

d	$\lceil \log_2 q \rceil$	$\lfloor \log_2 l_{\mathrm{add}} \rfloor$	proof size (kB)
8192	264	28.5	363.7
16384	423	77.8	627.7

- There are approximately 90 items whose production volume is determined by statistics, according to the Ministry of Agriculture, Forestry, and Fisheries, Japan[1]. The traffic is proportional to the number of items since one item corresponds to one encrypted model.

- SEVEN-ELEVEN JAPAN CO., LTD. has opened approximately 21,009 stores in Japan as of October 31, 2019[2]. The number of participants corresponds to the number of stores. The number of participants is approximately 21,000. It does not affect the traffic because it is much smaller than the upper limit $2^{28.5}$ in Table 3.4.

- Frequency of proof is once every 10 minutes, as in Bitcoin. We assume that the permissioned blockchain proves the traceability once every 10 minutes.

- Properties of traceability amount to three. In Table 3.4, we apply $l_{\mathrm{mul}} = 2$ regarding trade privacy and $l_{\mathrm{mul}} = 1$ regarding preservation and noninvolvement. This fact triples the traffic.

- The proof size is 363.7 kB according to Table 3.4.

Therefore, the traffic is equal to

$$363.7 \text{ kB} \times 3 \text{ property} \times 90 \text{ item}/600 \text{ sec} = 1.3 \text{ Mbps}.$$

[1] http://www.maff.go.jp/j/seisan/ryutu/yasai/yasai_jousei_0111.pdf (Website in Japanese)
[2] https://www.sej.co.jp/company/en/n_stores.html

The global averages on fixed broadband are 70.68 Mbps (download) and 38.23 Mbps (upload) as of October 2019, according to Speedtest by Ookla[3]. The traffic is sufficiently smaller than the averages. It is feasible.

3.5 Conclusion

We have achieved privacy protection and high transparency in a permissioned blockchain. Meaningful traceability consists of three properties. That is trade privacy, preservation, and noninvolvement. This work is the first proposal wherein both preservation and noninvolvement hold while protecting trade privacy. We have constructed a traceability model based on the three properties and encrypted it with the RLWE encryption. Moreover, we have encrypted the model with somewhat homomorphic encryption by using the ring isomorphism encoding. We have confirmed that the encrypted model is feasible. Finally, we have confirmed that one can prove with zero-knowledge that the encrypted traceability model is proper.

Limitation. In this work, we have expressed the traceability model of Definition 3.6 with a fixed-point number. As this extension, we can take an arbitrary denominator. Then, the number of multiplications is approximately the same as the number of participants since a reduction to a common denominator is necessary. Because of many numbers of multiplication, we can extend this work theoretically by fully homomorphic encryption.

[3] https://www.speedtest.net/global-index

Chapter 4

Confidential and auditable payments

This chapter is based on the below work with Akira Otsuka.

[**MO20b**] T. Mitani and A. Otsuka, "Confidential and Auditable Payments," in Financial Cryptography and Data Security. FC 2020. Lecture Notes in Computer Science, vol 12063, pp 466-480. Springer, Cham.

4.1 Introduction

Bitcoin is transparent because all the transactions are public. However, a bank typically keeps the customer's transaction information confidential. One hopes that blockchain also keeps their transaction information concealed. Many works realize anonymity and confidentiality in the blockchain. However, these works have rarely a forcibly auditable function. It is a problem that excessive confidentiality of transaction information may cause money laundering.

In this work, we construct the Confidential and Auditable Payments (CAP) scheme. The scheme allows a court or an authority to audit transactions while keeping the transaction information confidential. Every participant writes their transactions as a ciphertext of homomorphic encryption in a ledger with a unique public key. The court or authority controls its secret key. They can forcefully decrypt the ciphertexts and confirm the information. The CAP scheme eliminates concerns about money laundering caused by confidentiality and contributes to blockchain's sound use.

4.1.1 Our approach

The proposed scheme is not only confidential but also auditable. We describe auditability at first. A pair of a public key pk_0 and a secret key sk_0 of homomorphic encryption is issued. A court or an authority controls the secret key sk_0. They can decrypts ciphertexts according to the appropriate procedure when requested. In this sense, the CAP scheme is auditable.

Let us state confidentiality. Suppose that a column vector of each participant's account balance $\vec{x}$ is updated to the next time state $\vec{x}'$ by the transition matrix A. For example, let us present a state transition for three participants. Let $\vec{x} = (x_1, x_2, x_3), \vec{x}' = (x'_1, x'_2, x'_3)$ and $A = (a_{ij})$. A is a matrix of size 3×3. The equation $\vec{x}' = \vec{x}A$ holds. The transition matrix A corresponds to the remittance by each participant. Participants use the public key pk_0 in common. Participants fill their balance in their ledger as ciphertext $\vec{\mathbf{x}}$. As for the transition matrix, each participant writes each element in the ledger as ciphertext $\mathbf{A}$. Then, participants write their balance of the next time, reflecting the remittance in the ledger as a ciphertext $\vec{\mathbf{x}}'$. All the ciphertexts in the ledger are the ciphertexts of homomorphic encryption. $\vec{\mathbf{x}}' - \vec{\mathbf{x}}\mathbf{A}$ is the ciphertext of zero if and only if $\vec{x}' = \vec{x}A$.

Let us confirm the remittance procedure when a sender i transfers to a recipient j. We name the sender Alice and the recipient Bob. The element a_{ij} of the transition matrix A corresponds to this remittance. We use the zero-knowledge proof of plaintext knowledge to show that the remittance is legitimate. Alice writes the proof regarding a_{ij} in the ledger. This proof shows that she has not sent more than her account balance. The knowledge corresponds to the randomness in creating the ciphertext of homomorphic encryption. At the time of remittance, she encrypts the randomness with Bob's public key pk_j. She sends it to him. He decrypts it with his secret key sk_j and obtains the randomness of the remittance. From the knowledge of the randomness collected, he proves that his updated account balance x'_j is correct by the operation $\vec{x}' = \vec{x}A$.

4.1.2 Related work

In Bitcoin, transaction information is open. For this reason, there are some works for concealing transaction information. Zerocoin [MGGR13] and Zerocash [SCG+14] are extensions based on Bitcoin. They realized strong anonymity and confidentiality by designing anonymous coins that skillfully combined commitments. Zerocash uses the zero-knowledge

succinct non-interactive argument of knowledge (zk-SNARK) [GGPR13] to show others that there is no fraud such as double-spending. Zether [BAZB20] is also a cryptocurrency that can conceal transaction information. It has been proposed as an extension based on Ethereum. Zether has also used the zero-knowledge proof Bulletproofs [BBB$^+$18] proposed by their group. Neither Zerocoin, Zerocash, nor Zether is forcibly auditable because of their strong anonymity and confidentiality.

Mitani and Otsuka expressed the state transition of a permissioned blockchain by using homomorphic encryption [MO19, MO20c]. In their scheme, the zero-knowledge proof of plaintext knowledge shows that the encrypted model's equation is established to outsiders of the blockchain. The validity of the state transition is certifiable. So their scheme is auditable concerning the permissioned blockchain.

4.1.3 Chapter organization

We organize the rest of this chapter as follows. Section 4.2 describes the definitions for a secure CAP scheme. Section 4.3 describes the construction of the CAP scheme, including data structures, algorithms, and security proofs. Section 4.4 states the conclusion.

4.2 Secure CAP scheme

In this section, we define the security of a CAP scheme. Regarding security, we follow Zerocash [SCG$^+$14]. Zerocash defines and satisfies the three properties: ledger indistinguishability, transaction non-malleability, and balance. In this sense, Zerocash is secure. We adjust the three properties for a CAP scheme. We modify ledger indistinguishability. Balance corresponds to the proof regarding the transition matrix. We combine transaction non-malleability and balance into non-malleability. In this sense, a CAP scheme is secure. Let us confirm the below definitions.

Definition 4.1 (Ledger indistinguishability)**.** *A CAP scheme*

$$\Pi := (\mathsf{Setup}, \mathsf{CreateAccount}, \mathsf{Transition}, \mathsf{Send}, \mathsf{Receive})$$

satisfies ledger indistinguishability, if for any $\lambda \in \mathbb{N}$ *and any probabilistic polynomial time*

algorithm $\mathcal{A}$, the advantage

$$\mathrm{Adv}^{\mathsf{L-IND}}_{\Pi,\mathcal{A}}(\lambda) := \mid \Pr[\mathrm{Exp}^{\mathsf{L-IND}}_{\Pi,\mathcal{A}}(\lambda) = 1] - 1/2 \mid$$

is negligible in λ, where $\mathrm{Exp}^{\mathsf{L-IND}}_{\Pi,\mathcal{A}}(\lambda)$ is defined in Fig. 4.1. In Fig. 4.1, a pair of the queries (Q, Q') must be the same type: CreateAccount *or* Transition*. If the query type is* Transition*, then $Q = (\mathsf{Transition}, A)$ and $Q' = (\mathsf{Transition}, A')$. A and A' must be the same size. Each element is generally different. That is, $A \neq A'$. $\vec{Q}$ and $\vec{Q}'$ are the lists of all the queries that $\mathcal{A}$ sent to the oracles.* Append *is the function to append the latest query to the lists.*

$\underline{\mathrm{Exp}^{\mathsf{L-IND}}_{\Pi,\mathcal{A}}(\lambda)}$:
$\mathsf{pp}, \mathsf{cp} \leftarrow \mathsf{Setup}(1^\lambda)$
$L_0 \leftarrow \mathcal{O}^\Pi_0(\mathsf{pp})$; $L_1 \leftarrow \mathcal{O}^\Pi_1(\mathsf{pp})$
$b \xleftarrow{\$} \{0, 1\}$
while:
  $(Q, Q') \leftarrow \mathcal{A}(\mathsf{pp}, \vec{Q}, \vec{Q}', L_b, L_{1-b})$
  $L_b \leftarrow \mathcal{A}^{\mathcal{O}^\Pi_b(\cdot)}(\mathsf{pp}, Q, L_b)$; $L_{1-b} \leftarrow \mathcal{A}^{\mathcal{O}^\Pi_{1-b}(\cdot)}(\mathsf{pp}, Q', L_{1-b})$
  $\vec{Q} \leftarrow \mathsf{Append}(\vec{Q}, Q)$; $\vec{Q}' \leftarrow \mathsf{Append}(\vec{Q}', Q')$
  $c \leftarrow \mathcal{A}(\mathsf{pp}, \vec{Q}, \vec{Q}', L_b, L_{1-b})$
  if $c = 1$ break
  else continue
$b' \leftarrow \mathcal{A}(\mathsf{pp}, \vec{Q}, \vec{Q}', L_b, L_{1-b})$
if $b = b'$ return 1
else return 0

Figure 4.1: Security challenge experiment for ledger indistinguishability

Definition 4.2 (Non-malleability)**.** *A CAP scheme*

$$\Pi := (\mathsf{Setup}, \mathsf{CreateAccount}, \mathsf{Transition}, \mathsf{Send}, \mathsf{Receive})$$

satisfies non-malleability, if for any $\lambda \in \mathbb{N}$ and any probabilistic polynomial time algorithm $\mathcal{A}$, the advantage

$$\mathrm{Adv}^{\mathsf{NM}}_{\Pi,\mathcal{A}}(\lambda) := \Pr[\mathrm{Exp}^{\mathsf{NM}}_{\Pi,\mathcal{A}}(\lambda) = 1]$$

is negligible in λ, where $\mathrm{Exp}^{\mathsf{NM}}_{\Pi,\mathcal{A}}(\lambda)$ is defined in Fig. 4.2. Besides, $\mathcal{O}^\Pi$ and Append *are the same as Definition 4.1.*

$\underline{\mathrm{Exp}^{\mathsf{NM}}_{\Pi,\mathcal{A}}(\lambda)}$:
$\mathsf{pp}, \mathsf{cp} \leftarrow \mathsf{Setup}(1^\lambda)$
$L \leftarrow \mathcal{O}^{\Pi}(\mathsf{pp})$
while:
  $Q \leftarrow \mathcal{A}(\mathsf{pp}, \vec{Q}, L)$
  $L \leftarrow \mathcal{A}^{\mathcal{O}^{\Pi}(\cdot)}(\mathsf{pp}, Q, L)$
  $\vec{Q} \leftarrow \mathsf{Append}(\vec{Q}, Q)$
  $c \leftarrow \mathcal{A}(\mathsf{pp}, \vec{Q}, L)$
  if $c = 1$ break
  else continue
$A^* \leftarrow \mathcal{A}(\mathsf{pp}, \vec{Q}, L)$
$L' \leftarrow \mathcal{A}^{\mathcal{O}^{\Pi}(\cdot)}(\mathsf{pp}, (\mathsf{Transition}, A^*), L)$
if ($\mathsf{Verify}(\pi^a)$ or $\mathsf{Verify}(\vec{\pi}^b)$ or $\mathsf{Verify}(\Pi^c)$ or $\mathsf{Verify}(\vec{\pi}')$)
    and $(\sum_i (\mathbf{x}'_i - \mathbf{x}_i) \notin \mathbf{C}_0)$ in L'
  return 1
else return 0

Figure 4.2: Security challenge experiment for non-malleability

Definition 4.3 (Security)**.** *A CAP scheme*

$$\Pi := (\mathsf{Setup}, \mathsf{CreateAccount}, \mathsf{Transition}, \mathsf{Send}, \mathsf{Receive})$$

is secure if it satisfies ledger indistinguishability of Definition 4.1 and non-malleability of Definition 4.2.

4.3 Construction

In this section, we describe notation, data structures, algorithms, and security analysis.

4.3.1 Notation

We introduce the zero-knowledge proof in Fig. 2.3 and execute this protocol in parallel according to Theorem 2.1. This chapter denotes the zero-knowledge proof of plaintext $m = 0$ knowledge in the RLWE encryption and its verification as below syntax. We represent the ciphertext in bold to distinguish from its plaintext.

- $\mathsf{Prove}(\mathbf{x}; r)$ inputs the ciphertext $\mathbf{x}$ and its randomness r and outputs the proof π for the knowledge of $\mathsf{RLWE.Dec}(\mathsf{sk}, \mathbf{x}) = 0$. We denotes $\mathsf{Prove}(\mathbf{x}; r_1, \ldots, r_n)$ if there are several randomnesses $r_1, \ldots, r_n$.

- $\mathsf{Verify}(\pi)$ inputs the proof π and outputs 1 if it is valid, otherwise 0.

4.3.2 Data structures

Let us describe the data structures of the CAP scheme. We consider that the state of the assets held by each participant will transition to the next state as Chapter 3.

- Let $\vec{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{Q}^n$. $\vec{x}$ is the amount of assets in the blockchain at time t. n is the number of the participants in a blockchain.

- Let $\vec{x}' = (x'_1, x'_2, \ldots, x'_n) \in \mathbb{Q}^n$. $\vec{x}'$ is the amount of assets in the blockchain at time $t+1$.

- Let $A = (a_{ij})$ be a transition matrix such that $\vec{x}' = \vec{x}A$. Its size is $n \times n$. Let v be the volume moving from x_i to x'_j. That is, $x'_i = x_i - v$ and $x'_j = x_j + v$. We define the distribution rate $a_{ij} := v/x_i$. a_{ii} is the staying rate ($x_i \to x'_i$). In the transition, the amount x_i is distributed to $x'_1, \ldots, x'_i, \ldots, x'_n$ at the ratio of $a_{i1}, \ldots, a_{ii}, \ldots, a_{in}$. The sum of all the ratios must be equal to 1 because the total amount is constant. That is, $\sum_{j=1}^{n} a_{ij} = 1$. Let us confirm the following lemma, which is the variant of Lemma 3.1.

  Lemma 4.1. *If $\vec{x}' = \vec{x}A$ and $\sum_{j=1}^{n} a_{ij} = 1$, then the equation*

  $$\sum_{i=1}^{n} x'_i = \sum_{i=1}^{n} x_i$$

  holds.

- Let Ledger L be a distributed ledger that each participant in the blockchain holds and updates. L contains the ciphertexts $\vec{\mathbf{x}}, \vec{\mathbf{x}}', \mathbf{A}$ and their related proofs.

Remark 4.1. *It is impractical to force all participants to join the transition matrix A every time. We assume a transition matrix A' that pertains only to participants trading at a particular time t. That is, A' is the submatrix of the transition matrix A for all participants. Since there is virtually no difference, we will discuss A and A' indiscriminately.*

4.3.3 Algorithms

We describe the CAP scheme as follows.

Definition 4.4 (The CAP scheme)**.** *Tha CAP scheme*

$$\Pi := (\mathsf{Setup}, \mathsf{CreateAccount}, \mathsf{Transition}, \mathsf{Send}, \mathsf{Receive})$$

is in Fig. 4.3.

We can verify the completeness of the CAP scheme Π by confirming the construction.

4.3.4 Security analysis

We describe the below lemmas for the security of the CAP scheme Π in Definition 4.4.

Lemma 4.2 (Ledger indistinguishability)**.** *The CAP scheme Π in Definition 4.4 satisfies ledger indistinguishability in Definition 4.1.*

Proof. We consider the games in Fig. 4.4. We denote events as follows.

- $E_0 : \mathrm{G}^0_{\mathsf{CPA},\mathcal{A}}(\lambda) = 1$
- $E_1 : \mathrm{G}^1_{\mathsf{CPA},\mathcal{A}}(\lambda) = 1$
- $B_0 : \beta = 0$
- $B_1 : \beta = 1$

Since B_0 and B_1 are disjoint events, we have

$$\Pr[E_i] = \Pr[E_i \cap B_0] + \Pr[E_i \cap B_1] = \Pr[E_i|B_0] \cdot \Pr[B_0] + \Pr[E_i|B_1] \cdot \Pr[B_1].$$

Then, $i = \{0, 1\}$. We construct an adversary $\mathcal{B}$ from an adversary $\mathcal{A}$ who can distinguish

Setup

- Inputs: security parameter λ
- Outputs: public parameters pp and court parameters cp

1. Compute $(\mathsf{pk}_0, \mathsf{sk}_0) := \mathsf{RLWE.Gen}(1^\lambda)$.
2. Set $\mathsf{pp} := \mathsf{pk}_0$.
3. Set $\mathsf{cp} := \mathsf{sk}_0$.
4. Output pp and cp.

CreateAccount

- Inputs: public parameters pp and security parameter λ
- Outputs: account ciphertext $\mathbf{x}$, proof π^a in L and key pair $(\mathsf{pk}, \mathsf{sk})$

1. Choose randomly $r_{\mathbf{x}}$.
2. Compute $\mathbf{x} := \mathsf{RLWE.Enc}(\mathsf{pk}_0, 0, r_{\mathbf{x}})$.
3. Compute $\pi^a := \mathsf{Prove}(\mathbf{x}; r_{\mathbf{x}})$.
4. Compute $(\mathsf{pk}, \mathsf{sk}) := \mathsf{RLWE.Gen}(1^\lambda)$.
5. Output $\mathbf{x}$, π and $(\mathsf{pk}, \mathsf{sk})$.

Send

- Inputs:
  - public parameters pp
  - account ciphertext $\mathbf{x}$ in Ledger L
  - randomness $r_{\mathbf{x}}$
  - plaintext x
  - sending volume v
  - public key pk_j
- Outputs:
  - copy proof π^c
  - copy account $\mathbf{x}^c$
  - distribution ratio $\mathbf{a}_{ij}$
  - volume $\mathbf{v}$
  - randomness $\mathbf{r}_{\mathbf{v}}$

1. Choose randomly $r_{\mathbf{x}^c}$, where $r_{\mathbf{x}^c} \neq r_{\mathbf{x}}$.
2. Compute $\mathbf{x}^c := \mathsf{RLWE.Enc}(\mathsf{pk}_0, x, r_{\mathbf{x}^c})$.
3. Compute $\pi^c := \mathsf{Prove}(\mathbf{x}^c - \mathbf{x}; r_{\mathbf{x}^c} - r_{\mathbf{x}})$.
4. Compute $a := v/x$.
5. Choose randomly $r_{\mathbf{a}_{ij}}$.
6. Compute $\mathbf{a}_{ij} := \mathsf{RLWE.Enc}(\mathsf{pk}_0, a, r_{\mathbf{a}_{ij}})$.
7. Compute $r_{\mathbf{v}}$ from $r_{\mathbf{x}^c}$ and $r_{\mathbf{a}_{ij}}$.
8. Compute $\mathbf{v} := \mathsf{RLWE.Enc}(\mathsf{pk}_j, v, r_{\mathbf{v}})$.
9. Compute $\mathbf{r}_{\mathbf{v}} := \mathsf{RLWE.Enc}(\mathsf{pk}_j, r_{\mathbf{v}}, r_{\mathbf{r}_{\mathbf{v}}})$.
10. Output $\pi^c, \mathbf{x}^c, \mathbf{a}_{ij}, \mathbf{v}, \mathbf{r}_{\mathbf{v}}$.

Transition

- Inputs:
  - public parameters pp
  - old accounts state $\vec{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ in L
  - accounts randomness $r_{\mathbf{x}_1}, \ldots, r_{\mathbf{x}_n}$
- Outputs in L:
  - new accounts state $\vec{\mathbf{x}}' = (\mathbf{x}'_1, \ldots, \mathbf{x}'_n)$
  - new transition matrix $\mathbf{A} = (\mathbf{a}_{ij})$
  - balance proofs $\vec{\pi}^b = (\pi^b_1, \ldots, \pi^b_n)$
  - accounts copy proofs $\Pi^c = (\pi^c_{ij})$
  - transition proofs $\vec{\pi}' = (\pi'_1, \ldots, \pi'_n)$

1. For $i \in \{1, \ldots, n\}$ (i sends v to j.)
   (a) For $j \in \{1, \ldots, n\}$
      i. $\mathsf{Send}(\mathsf{pp}, \mathbf{x}_i, r_{\mathbf{x}_i}, x_i, v, \mathsf{pk}_j)$.
      ii. Write $\pi^c_{ij}, \mathbf{x}^c_{ij}, \mathbf{a}_{ij}$ in L.
   (b) Compute $\pi^b_i := \mathsf{Prove}(\sum_j \mathbf{a}_{ij} - \mathbf{1}; r_{\mathbf{a}_{i1}}, \ldots, r_{\mathbf{a}_{in}})$.
   (c) Write π^b_i in L.
2. $\mathsf{Verify}(\Pi^c)$
3. $\mathsf{Verify}(\vec{\pi}^b)$
4. For $j \in \{1, \ldots, n\}$ (j receives v from i.)
   (a) For $i \in \{1, \ldots, n\}$ $\mathsf{Receive}(\mathsf{pp}, \mathsf{sk}_j, \mathbf{x}^c_i, \mathbf{a}_{ij}, \mathbf{v}, \mathbf{r}_{\mathbf{v}}, \pi^c_{ij})$.
   (b) Compute $x'_j := \sum_i v_{ij}$.
   (c) Choose $r_{\mathbf{x}'_j}$ randomly.
   (d) Compute $\mathbf{x}'_j := \mathsf{RLWE.Enc}(\mathsf{pk}_0, x'_j, r_{\mathbf{x}'_j})$.
   (e) Compute $\pi'_j := \mathsf{Prove}(\mathbf{x}'_j - \sum_i \mathbf{a}_{ij} \cdot \mathbf{x}^c_i; r_{\mathbf{x}'_j}, r_{\mathbf{v}_{1j}}, \ldots, r_{\mathbf{v}_{nj}})$.
   (f) Write $\mathbf{x}'_j, \pi'_j$ in L.
5. $\mathsf{Verify}(\vec{\pi}')$

Receive

- Inputs:
  - public parameters pp
  - secret key sk_j
  - ciphertexts $\mathbf{x}^c, \mathbf{a}_{ij}, \mathbf{v}, \mathbf{r}_{\mathbf{v}}$
  - proof π^c in L
- Outputs:
  - volume v
  - randomness $r_{\mathbf{v}}$

1. Compute $v := \mathsf{RLWE.Dec}(\mathsf{sk}_j, \mathbf{v})$.
2. Compute $r_{\mathbf{v}} := \mathsf{RLWE.Dec}(\mathsf{sk}_j, \mathbf{r}_{\mathbf{v}})$.
3. If $\mathbf{a}_{ij} \cdot \mathbf{x}^c = \mathsf{RLWE.Enc}(\mathsf{pk}_0, v, r_{\mathbf{v}})$ and $|r_{\mathbf{v}}|$ is small: Output $v, r_{\mathbf{v}}$.
4. Else: Output $\perp$.

Figure 4.3: Construction of the CAP scheme

$\underline{\mathrm{G}^0_{\mathsf{CPA},\mathcal{A}}(\lambda)}$:
$(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{RLWE.Gen}(1^\lambda)$
$\beta \leftarrow \{0,1\}$
$\beta' \leftarrow \mathcal{A}^{\mathcal{O}_0^{\mathsf{CPA}}(\cdot,\cdot)}(1^\lambda, \mathsf{pk})$
if $\beta = \beta'$ return 1
else return 0

$\underline{\mathrm{G}^1_{\mathsf{CPA},\mathcal{A}}(\lambda)}$:
$(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{RLWE.Gen}(1^\lambda)$
$\beta \leftarrow \{0,1\}$
$\beta' \leftarrow \mathcal{A}^{\mathcal{O}_1^{\mathsf{CPA}}(\cdot,\cdot)}(1^\lambda, \mathsf{pk})$
if $\beta = \beta'$ return 1
else return 0

$\underline{\mathcal{O}_0^{\mathsf{CPA}}(m_0,m_1)}$:
if $\beta = 0$
  Choose r randomly.
  $c \leftarrow \mathsf{RLWE.Enc}(\mathsf{pk}, m_0, r)$
else
  Choose r randomly.
  $c \leftarrow \mathsf{RLWE.Enc}(\mathsf{pk}, m_1, r)$
return c

$\underline{\mathcal{O}_1^{\mathsf{CPA}}(m_0,m_1)}$:
$c \xleftarrow{\$} \mathbf{R}_q^2$
return c

Figure 4.4: Security challenge experiment for plaintexts

$\mathrm{G}^0_{\mathsf{CPA},\mathcal{A}}(\lambda)$ and $\mathrm{G}^1_{\mathsf{CPA},\mathcal{A}}(\lambda)$. That is,

$$
\begin{aligned}
&| \Pr[\mathrm{G}^0_{\mathsf{CPA},\mathcal{A}}(\lambda) = 1] - \Pr[\mathrm{G}^1_{\mathsf{CPA},\mathcal{A}}(\lambda) = 1] | \\
=&| \Pr[E_0] - \Pr[E_1] | \\
=&| (\Pr[E_0|B_0] \cdot \Pr[B_0] + \Pr[E_0|B_1] \cdot \Pr[B_1]) \\
&\quad - (\Pr[E_1|B_0] \cdot \Pr[B_0] + \Pr[E_1|B_1] \cdot \Pr[B_1]) | \\
=&| \Pr[B_0] \cdot (\Pr[E_0|B_0] - \Pr[E_1|B_0]) + \Pr[B_1] \cdot (\Pr[E_0|B_1] - \Pr[E_1|B_1]) | \\
\leq& \Pr[B_0] \cdot | \Pr[E_0|B_0] - \Pr[E_1|B_0] | + \Pr[B_1] \cdot | \Pr[E_0|B_1] - \Pr[E_1|B_1] | \\
=& (\Pr[B_0] + \Pr[B_1]) \cdot \mathrm{Adv}^{\mathsf{pr}}_{\mathsf{RLWE},\mathcal{B}}(\lambda) \\
=& \mathrm{Adv}^{\mathsf{pr}}_{\mathsf{RLWE},\mathcal{B}}(\lambda)
\end{aligned}
$$

where we applied $\Pr[B_0] + \Pr[B_1] = 1$ in the last equation. Since the game $\mathrm{G}^1_{\mathsf{CPA},\mathcal{A}}(\lambda)$ is independent of β, we have thus $\Pr[\mathrm{G}^1_{\mathsf{CPA},\mathcal{A}}(\lambda) = 1] = 1/2$. Moreover, $\mathrm{Exp}^{\mathsf{L-IND}}_{\Pi,\mathcal{A}}(\lambda)$ consists of the game $\mathrm{G}^0_{\mathsf{CPA},\mathcal{A}}(\lambda)$. Because of Definition 2.4, we obtain

$$\mathrm{Adv}^{\mathsf{L-IND}}_{\Pi,\mathcal{A}}(\lambda) \leq | \Pr[\mathrm{G}^0_{\mathsf{CPA},\mathcal{A}}(\lambda) = 1] - 1/2 | \leq \mathrm{Adv}^{\mathsf{pr}}_{\mathsf{RLWE},\mathcal{B}}(\lambda) < \mathrm{negl}(\lambda).$$

□

Lemma 4.3 (Non-malleability)**.** *The CAP scheme Π in Definition 4.4 satisfies non-malleability in Definition 4.2.*

Proof. Let us consider the below violations of verification in the cases $\mathcal{A}$ wins without the

knowledge of randomness.

- $\mathcal{A}$ wins but violates $\mathsf{Verify}(\pi^a)$ in $\mathsf{CreateAccount}$.
- $\mathcal{A}$ wins but violates $\mathsf{Verify}(\vec{\pi}^b)$ in $\mathsf{Transition}$.
- $\mathcal{A}$ wins but violates $\mathsf{Verify}(\Pi^c)$ in $\mathsf{Transition}$.
- $\mathcal{A}$ wins but violates $\mathsf{Verify}(\vec{\pi}')$ in $\mathsf{Transition}$.

Any violations result from the violation of the zero-knowledge proof. This is the knowledge error of zero-knowledge proof. From Theorem 2.1, the knowledge error of zero-knowledge proof is negligible. □

Finally, we can lead the below theorem from Lemma 4.2 and Lemma 4.3.

Theorem 4.1 (The secure CAP scheme)**.** *The CAP scheme* Π *in Definition 4.4 is secure in the sense of Definition 4.3.*

4.4 Conclusion

We have constructed the Confidential and Auditable Payments(CAP) scheme. The CAP scheme allows a court or an authority to audit transactions while keeping the transaction information confidential. Every participant writes the ciphertexts of transaction information in a ledger. We have confirmed the concealment of the transaction information and the soundness of the CAP scheme. The CAP scheme is secure in this sense. A court or an authority can forcibly reveal transaction information with a unique secret key. In this sense, we realized auditability in the CAP scheme.

Limitation. In the CAP scheme, the secret key can decrypt all transaction information. Therefore, we expect the court or authority to disclose minimum requisite information.

Chapter 5

Anonymous probabilistic payment in payment hub

This chapter is based on the below preprint with Akira Otsuka.

[**MO20a**] T. Mitani and A. Otsuka, "Anonymous probabilistic payment in payment hub," Cryptology ePrint Archive, Report 2020/748, 2020. `https://eprint.iacr.org/2020/748`

5.1 Introduction

There are many problems that blockchain needs to be solved. Scalability and privacy protection are significant problems in blockchain. However, there are a few proposals to solve these problems at the same time.

Let us state privacy protection. The transaction links a payer and a payee publicly in Bitcoin. One can know their balance and trading frequency by analyzing the links of transactions. Privacy is not protected. Breaking the transaction link is necessary to protect privacy.

Let us describe scalability. It is costly to write all the small transactions into the blockchain. The payment of a small amount of money is called micropayment. For such micropayments, Wheeler [Whe97] and Rivest [Riv97] proposed a probabilistic payment before blockchain appears. It reduces costs for micropayment. In probabilistic payment, we pay an ordinary mount m with a certain probability p. We pay a small amount mp as an expected value. By the probability p (e.g. $p \approx 0.1 \sim 0.001$), we can reduce $1/p$

times transactions than a naively deterministic payment. Micropay [Ps15], the DAM scheme [CGL+17] and Microcash [ABC20] have proposed a new micropayment on the blockchain. It is attracting attention as one of the leading solutions for scalability in the blockchain.

5.1.1 Our contribution

We propose an anonymous probabilistic payment. It aims to solve both scalability and privacy protection. In this work, we realize a kind of unlinkability: "k-anonymity in an epoch." TumbleBit [HAB+17] and their earlier work [HBG16] mentioned this definition. The link means that which payer pays which payee via a tumbler within an epoch. Anonymity means the link is broken. Even a tumbler never knows the link. The "k" is the number of participants trading via the tumbler. The epoch is the period during which transactions are completed.

Our proposal includes a probabilistic payment. In the probabilistic payment, we pay an ordinary mount m with a certain probability p (e.g. $p \approx 0.1 \sim 0.001$) and we pay a small amount mp as an expected value. We can reduce $1/p$ times transactions than a deterministic payment. We introduce a novel fractional oblivious transfer for adopting the probabilistic payment. We apply this ingredient to a probabilistic payment from the payer to the tumbler. We call it the ring fractional oblivious transfer since this is based on the ring learning with errors (RLWE) encryption. The functionality required for our proposal is hashed time lock contract (HTLC). Various cryptocurrencies adapt HTLC. This request is general, not restricted to any particular cryptocurrency.

5.1.2 Related work

In this subsection, we describe the related work regarding anonymity in blockchain, probabilistic payment, fractional oblivious transfer, and the comparison with a concurrent work.

Anonymity in blockchain

There are researches on a new anonymous cryptocurrency. Zerocash [SCG+14] is a famous anonymous cryptocurrency and is implemented as ZCash. They also have proposed continuous researches such as BOLT [GM17] and DAM scheme [CGL+17]. Mon-

ero is also a famous anonymous cryptocurrency and is provided with incredible works [Noe15, SALY17, YSL+20, MSRL+19]. There are studies to realize anonymity for existing cryptocurrencies by using off-chain technology. TumbleBit [HAB+17] is compatible with Bitcoin. Zether [BAZB20] is compatible with Ethereum.

Probabilistic payments

Wheeler [Whe97] and Rivest [Riv97] proposed a probabilistic payment. Micropay [Ps15] is compatible with Bitcoin. Microcash [ABC20] can be implemented as a smart contract. The DAM scheme, which is the extension of anonymous ZCash, also realizes a probabilistic payment.

Fractional oblivious transfer over the ring

Bellare and Micali [BM90] and Bellare and Rivest [BR99] proposed the fractional oblivious transfer based on the computational Diffie-Hellman assumption as the early works. The DAM scheme [CGL+17, CGL+16] also proposed a novel fractional oblivious transfer based on the decisional Diffie-Hellman assumption as fractional message transfer. Note that [CGL+16] is the full version of [CGL+17].

Brakerski and Döttling proposed an oblivious transfer based on the learning with errors [BD18]. This work is the first oblivious transfer in the post-quantum cryptography. Liu and Hu first proposed an efficient 1-out-of-2 oblivious transfer (OT) on the RLWE assumption and extends 1-out-of-n OT [LH19]. To the best of our knowledge, we first propose the fractional oblivious transfer over the ring.

Comparison with a concurrent work

The DAM scheme [CGL+17, CGL+16] is an anonymous probabilistic payment, which passes ZCash transaction by a fractional oblivious transfer. Since the scheme includes a ZCash transaction in the message, it is a ZCash specific implementation. Regarding our proposal, the required functionality is the hashed time lock contract (HTLC). Various cryptocurrencies such as Bitcoin, implement HTLC. Therefore our proposal is not limited to a specific cryptocurrency.

5.1.3 Our approach

We present the anonymous probabilistic payment approach and the ring fractional oblivious transfer, which is a vital ingredient in our proposal.

Anonymous probabilistic payment

We propose an anonymous probabilistic payment that a payer pays a payee via a tumbler. We name the payer Alice and the payee Bob. Suppose that she wants to send one coin to him through the tumbler with a certain probability p. We show the overview of the protocol in Fig. 5.1. Our protocol mostly follows TumbleBit [HAB+17]. The difference is the use of RLWE encryption and RFOT.

Let us explain each phase in the protocol: setup, puzzle promise, and puzzle solver as follows. In the setup phase, the tumbler generates a one-time key pair consisting of a public key and a private key. The tumbler also attaches proof of the zero-knowledge proof. The tumbler publishes this proof and the public key.

In the promise phase, the tumbler and Bob interact. The tumbler prepares an escrow transaction that pays one coin to him from the tumbler. Both the signatures of the tumbler and he can execute this escrow transaction. The tumbler does not present its signature to him in this phase. Instead, the tumbler creates puzzles and promises from its signature and presents them to him. If he shows this puzzle answer, then he obtains the tumbler signature from the promise and puzzle.

To gain the answer, Bob will ask Alice to solve the puzzle. Then, he masks the puzzle to delete his link and sends it to her. She also masks the received puzzle from him to delete her link. She will pay and get the answer to the puzzle with a probability p in the next phase.

In the solver phase, Alice interacts with the tumbler to get the puzzle answer. She makes a probabilistic payment to the tumbler, paying one coin with probability p. Then, she sends the double-masked puzzle to the tumbler. She issues a transaction to the tumbler and asks the tumbler to post the puzzle answer. She asks the tumbler to decrypt in exchange for the commitment to the transaction. The tumbler posts the answer and executes the transaction. Alice demasks the double-masked answer and sends the single masked answer to Bob.

In the cash-out phase, Bob receives the single masked answer from Alice. He demasks

it and obtains the answer. As the tumbler and he promised in the promise phase, he gets the tumbler signature from the answer. Finally, he executes the escrow transaction and gains one coin.

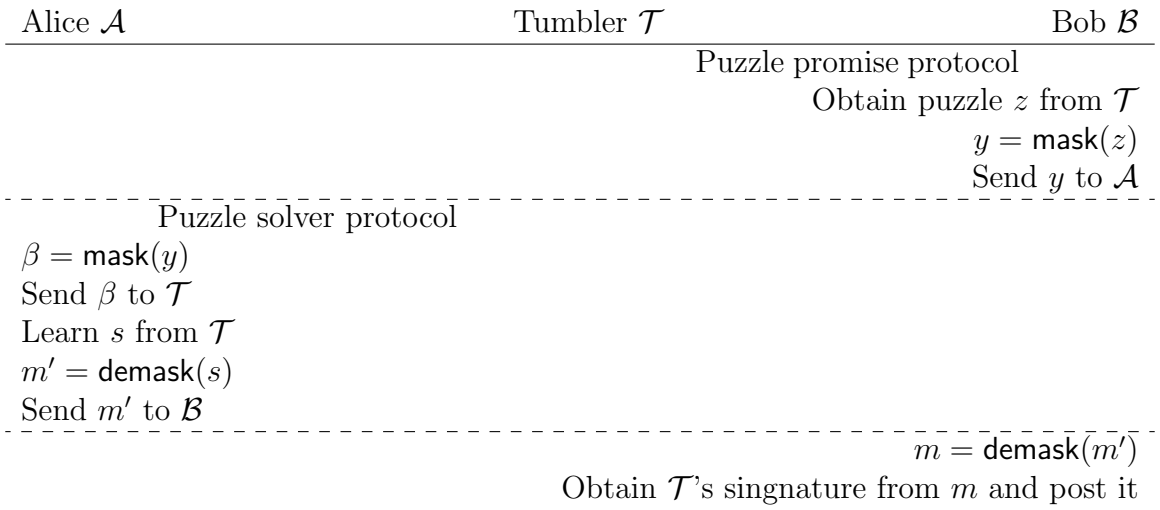

Figure 5.1: Overview of the proposed protocol

Ring fractional oblivious transfer

For the probabilistic payment in the solver phase, we introduce a novel ingredient. It is a fractional oblivious transfer based on the RLWE encryption. We call it the ring fractional oblivious transfer (RFOT). Let us describe RFOT.

Basic idea. In the RLWE encryption, it is essential to operate on the ring $\mathbf{R}_q = \mathbb{Z}_q[X]/\langle X^d + 1\rangle$. An element is a polynomial of $n-1$ degree. That is, $a = a_0 + a_1 X + \cdots + a_{d-1}X^{d-1}$. Let us compare an element $a \in \mathbf{R}_q$ with aX^k multiplied by a monomial X^k with an integer k. One can choose k randomly from the set $\{1, \ldots, n\}$, where $n < d$. Because $X^d = -1$, the original coefficient of the element a is cyclically shifted regardless of the sign. One sample coefficients of the element a randomly from the uniform distribution in key generation or encryption. The distribution is positive and negative symmetric. The element a chosen from the distributions and the shifted aX^k are indistinguishable. Looking at the aX^k, one cannot identify how much the shift is.

Procedure. Suppose that a sender sends a message m to a receiver. We name the sender Alice and the receiver Bob. First, he creates a one-time key pair, a public key,

and private key. He randomly chooses an integer $k \in \{1, \ldots, n\}$, cyclically shifts the public key by multiplying a monomial X^{-k}. Furthermore, Bob issues this shifted public key to her. She creates the ciphertext by the shifted public key. She does not know the shift amount of k. She also randomly chooses an integer $l \in \{1, \ldots, n\}$, and shifts the ciphertext by multiplying a monomial X^l. He receives the shifted ciphertext and decrypts it by the secret key. If both shifted amounts l and k match, he can obtain a message m in the specified format. If they do not match, he has a broken message $\emptyset$. The probability that both l and k match is $1/n$.

Security. We prove that the RFOT satisfies the fractional hiding and fractional binding. The DAM scheme [CGL+17, CGL+16] introduced the two properties. Fractional hiding is the property that a ciphertext created by an honest encryptor can be decrypted exactly with probability p. Fractional binding is the property that a malicious encryptor cannot create valid ciphertext that can be decrypted with probability $p' \neq p$. We prove the fractional hiding and fractional binding security with the simulation-based security like [CGL+17, CGL+16].

Let us explain the trapdoor for the simulated RFOT. We use the scheme for the trapdoor proposed in the identity-based encryption over NTRU lattice [DLP14]. We regard the one-time public key and private key as a user key and the simulated trapdoor as a master key. It can directly produce the small elements of the RLWE encryption. One can simulate a shifted secret key or plaintext and randomness from the randomly sampled elements s_1, s_2 such that $y = as_1 + 2s_2$ for the given y and the trapped a. The trapdoor requires the decisional small polynomial ratio (DSPR) assumption.

A trapdoor using a gadget matrix is known. Micciancio and Peikert proposed a gadget matrix based trapdoor [MP12]. Following this work, Genise and Micciancio proposed an efficient Gaussian sampler for the trapdoor [GM18]. Cousins *et al.* proposed the implementation of the RLWE encryption [CDG+18]. However, this method does not output the small elements required for RLWE encryption.

5.1.4 Chapter organization

We describe the organization of this work. In Section 5.2, we introduce the ring fractional oblivious transfer. In Section 5.3, we introduce the puzzle solver protocol and the puzzle

promise protocol and discuss the security of the protocols. In Section 5.4, we conclude this work.

5.2 Ring fractional oblivious transfer

In this section, we introduce a fractional oblivious transfer over the ring. First, we describe the trapdoor and the zero-knowledge proof. Next, we introduce the syntax and the definition of the scheme and its simulator, and its properties, referring to [CGL$^+$16]. Finally, we present the construction of the ring fractional oblivious transfer and discuss the security.

5.2.1 Trapdoor

We introduce the trapdoor based on DSPR assumption according to [DLP14]. We use $\mathsf{IBE.MasterKeygen}(d, q)$ and $\mathsf{IBE.Extract}(\mathbf{B}, t)$ of [DLP14]. We show the functions in Fig. 5.2. We state the difference between our functions and [DLP14]. [DLP14] calculates the hash value of the second argument in the function IBE.Extract. However, we use the target value t as it is.

- Master key generation: $\mathsf{IBE.MasterKeygen}(d, q) \to (a, \mathbf{B}, g_a, f_a)$

  On input a dimension d of the ring $\mathbf{R}$ and a modulo q of the ring $\mathbf{R}_q = \mathbf{R}/q\mathbf{R}$, IBE.MasterKeygen outputs a master public key a, a master secret key $\mathbf{B}$, and small polynomials g_a, f_a such that $a = g_a \cdot f_a^{-1}$.

- Extractor: $\mathsf{IBE.Extract}(\mathbf{B}, t) \to (s_1, s_2)$

  On input a master secret key $\mathbf{B}$ and a target value t, IBE.Extract outputs a pair of small polynomials (s_1, s_2) such that $t = as_1 + 2s_2$, where a is the master public key.

Figure 5.2: Trapdoor by identity-based encryption [DLP14]

5.2.2 Non-interactive zero-knowledge proof

Let us state the non-interactive zero-knowledge proof (NIZK). We show the syntax of NIZK in Fig. 5.3, referring to [CGL$^+$16]. In this work, we represent NIZK according to the syntax. $\mathcal{R}$ is a relation regarding an instance x and a witness w. If x and w satisfy the relation $\mathcal{R}$, then we denote $(x, w) \in \mathcal{R}$. The non-interactive zero-knowledge proof for the relation $\mathcal{R}$ is a protocol that satisfies the properties in Fig. 5.4.

- Setup: $\mathsf{NIZK.Setup}(1^\lambda, \mathcal{R}) \to \mathsf{crs}$

  On input a security parameter λ and a relation $\mathcal{R}$, NIZK.Setup outputs a common reference strings crs.

- Prove: $\mathsf{NIZK.Prove}(\mathsf{crs}, x, w) \to \pi$

  On input a common reference strings crs, an instance x, and a witness w, NIZK.Prove outputs a proof π.

- Verify: $\mathsf{NIZK.Verify}(\mathsf{crs}, x, \pi) \to 1$ or 0

  On input a common reference strings crs, an instance x, and a proof π, NIZK.Verify outputs 1 if π is valid, otherwise 0.

- Simulated setup: $\mathsf{NIZK.SimSetup}(1^\lambda, \mathcal{R}) \to (\mathsf{crs}, \mathsf{td})$

  On input a security parameter λ and a relation $\mathcal{R}$, NIZK.SimSetup outputs a common reference strings crs and a trapdoor td.

- Knowledge extractor: $\mathsf{NIZK.Extract}(\mathsf{crs}, \mathsf{td}, x, \pi) \to w$

  On input a common reference strings crs, a trapdoor td, an instance x, and a proof π, NIZK.Extract outputs a witness w.

- Simulator: $\mathsf{NIZK.Simulate}(\mathsf{crs}, \mathsf{td}, x) \to \pi$

  On input a common reference strings crs, a trapdoor td, and an instance x, NIZK.Simulate outputs a proof π.

- Knowledge extracting simulator: $\mathsf{NIZK.ExtSimulate}(\mathsf{crs}, \mathsf{td}, x) \to \pi$

  On input a common reference strings crs, a trapdoor td, and an instance x, NIZK.ExtSimulate outputs a proof π.

Figure 5.3: Syntax of NIZK [CGL+16]

1. Completeness For any $(x, w) \in \mathcal{R}$, the conditional probability

$$\begin{aligned}&\Pr[\mathsf{NIZK.Verify}(\mathsf{crs}, x, \pi) = 1 \wedge (x, w) \in \mathcal{R} \mid \\ &\quad \mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda, \mathcal{R}), \pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, x, w)] \geq 1 - \mathsf{negl}(\lambda).\end{aligned}$$

2. Soundness

   Let a knowledge extractor $\mathsf{NIZK.Extract}$ for any adversary $\mathcal{A}$ exist. For all $(x, w) \notin \mathcal{R}$, the conditional probability

$$\begin{aligned}&\Pr[\mathsf{NIZK.Verify}(\mathsf{crs}, x, \pi') = 1 \wedge (x, w') \notin \mathcal{R} \mid \\ &\quad (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK.SimSetup}(1^\lambda, \mathcal{R}), \pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, x, w), \\ &\quad w' \leftarrow \mathsf{NIZK.Extract}(\mathsf{crs}, \mathsf{td}, x, \pi), \pi' \leftarrow \mathcal{A}(\mathsf{crs}, x, w')] \leq \mathsf{negl}(\lambda).\end{aligned}$$

3. Zero-knowledge

   For any adversary, the below two distribution ensembles $\mathcal{E}_Z^{\mathsf{real}}, \mathcal{E}_Z^{\mathsf{ideal}}$ are computationally indistinguishable.

$$\begin{aligned}\mathcal{E}_Z^{\mathsf{real}} &= \{(\mathcal{R}, \mathsf{crs}, x, \pi) \mid \mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda, \mathcal{R}), \pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, x, w)\} \\ \mathcal{E}_Z^{\mathsf{ideal}} &= \{(\mathcal{R}, \mathsf{crs}, x, \pi) \mid (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK.SimSetup}(1^\lambda, \mathcal{R}), \\ &\qquad\qquad \pi \leftarrow \mathsf{NIZK.Simulate}(\mathsf{crs}, \mathsf{td}, x)\}\end{aligned}$$

Figure 5.4: The properties of NIZK

Let us explain NIZK.ExtSimulate regarding the relation $\mathcal{R}_E$ for a randomly sampled pair (c_1, c_2) as follows. We adopt the special trapdoor td_I since a pair (c_1, c_2) is randomly sampled.

Definition 5.1 (Extract Simulate with trapdoor)**.** NIZK.ExtSimulate *regarding the relation* $\mathcal{R}_E$ *for a given pair* $(c_1, c_2) \xleftarrow{\$} \mathbf{C}$ *is stated as follows.*

- $\mathsf{NIZK.ExtSimulate}(\mathsf{crs}_E, \mathsf{td}_I, x) \to \pi$

  1. *Parse* x *as* (c_1, c_2, a, y).
  2. *Parse* td_I *as* $(\mathbf{B}, g_a, f_a)$.
  3. *Compute* $(v', f') = \mathsf{IBE.Extract}(\mathbf{B}, c_2)$.
  4. *Compute* $(s_1, s_2) = \mathsf{IBE.Extract}(\mathbf{B}, (c_1 - yv')f_a^{-1})$.
  5. *Compute* m', e' *from* $(c_1 - yv')f_a^{-1} = as_1 + 2s_2$.
  6. *Compute* $\pi = \mathsf{NIZK.Prove}(\mathsf{crs}_E, x, (m', v', e', f'))$.
  7. *Output* π.

Remark 5.1. *Let us confirm how to obtain* m' *and* e' *from* $(c_1 - yv')f_a^{-1} = as_1 + 2s_2$. *Note that* $a = g_a f_a^{-1}$, *where* f_a *and* g_a *are small polynomials.*

$$
\begin{aligned}
(c_1 - yv')f_a^{-1} &= as_1 + 2s_2 \\
(c_1 - yv')f_a^{-1} &= s_1 g_a f_a^{-1} + 2s_2 \\
c_1 - yv' &= s_1 g_a + 2s_2 f_a \\
c_1 &= yv' + s_1 g_a + 2s_2 f_a
\end{aligned}
$$

Let $m' = s_1 g_a \mod 2$ *and* $e' = s_2 f_a + s_1 g_a - m'$. *We obtain* $c_1 = yv' + 2e' + m'$.

Lemma 5.1. *Let* NIZK.ExtSimulate *be defined as Definition 5.1. The below two distribution ensembles* $\mathcal{E}_{Z'}^{\mathsf{real}}$ *and* $\mathcal{E}_{Z'}^{\mathsf{ideal}}$ *are computationally indistinguishable.*

$$
\begin{aligned}
\mathcal{E}_{Z'}^{\mathsf{real}} &= \{(\mathcal{R}_E, \mathsf{crs}, x, \pi) \mid \mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda, \mathcal{R}_E), \pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, x, w)\} \\
\mathcal{E}_{Z'}^{\mathsf{ideal}} &= \{(\mathcal{R}_E, \mathsf{crs}, x, \pi) \mid (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK.SimSetup}(1^\lambda, \mathcal{R}_E), \\
&\qquad\qquad \pi \leftarrow \mathsf{NIZK.ExtSimulate}(\mathsf{crs}, \mathsf{td}, x)\}
\end{aligned}
$$

Proof. Let us confirm Definition 5.1 of NIZK.ExtSimulate. A simulated witness $w' = (m', v', e', f')$ is generated from the trapdoor td_I. Then the faithful witness w is entirely unnecessary. We can choose a small witness w' that fits the norm constraint by IBE.Extract. Besides, we can call the actual proving function NIZK.Prove inline. The simulated proof by NIZK.ExtSimulate is computationally indistinguishable from the proof by NIZK.Prove. We conclude that the two distribution ensembles $\mathcal{E}^{\mathsf{real}}_{Z'}$ and $\mathcal{E}^{\mathsf{ideal}}_{Z'}$ are computationally indistinguishable. □

5.2.3 Syntax and definition

We introduce the scheme's definitions, the correctness, the simulator, fractional hiding, and fractional binding. These definitions appear initially as fractional message transfer in [CGL+16]. We follow the syntax and the definition in [CGL+16].

Syntax and definition of scheme and simulator

We state the syntax and the definition regarding the scheme and its simulator. We show the syntax and definition of the scheme in Fig. 5.5.

Definition 5.2 (Correctness [CGL+16])**.** *A scheme is correct if for every security parameter λ, public parameters $\mathsf{pp} \in \mathsf{RFOT.Setup}(1^\lambda)$, probability $p \in [0,1]$, key pair $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{RFOT.Keygen}(\mathsf{pp}, p)$, and message $m \in \mathbf{M}$,*

$$\mathsf{RFOT.Decrypt}(\mathsf{pp}, \mathsf{sk}, \mathsf{RFOT.Encrypt}(\mathsf{pp}, \mathsf{pk}, m)) = \begin{cases} m & \textit{with probability } p \\ \emptyset & \textit{with probability } 1-p \end{cases}$$

Remark 5.2. *The types of a decrypted message m' are as follows. Let us name a valid/invalid message hit/miss since we illustrate a probabilistic payment as a lottery ticket.*

- *m : well-formatted valid message. We also call it "hit."*
- *$\emptyset$: broken formatted invalid message. We also call it "miss."*
- *$\perp$: error message reporting error occurrence*

The syntax of the scheme

- Setup: $\mathsf{RFOT.Setup}(1^\lambda) \to \mathsf{pp}$

  On input a security parameter λ, RFOT.Setup outputs a public parameter pp.

- Key generation: $\mathsf{RFOT.Keygen}(\mathsf{pp}, p) \to (\mathsf{pk}, \mathsf{sk})$

  On input a public parameter pp and a probability $p \in [0, 1]$, RFOT.Keygen outputs a public key pk and a secret key sk.

- Encryption: $\mathsf{RFOT.Encrypt}(\mathsf{pp}, \mathsf{pk}, m) \to c$

  On input a public parameter pp, a public key pk, and a message m, RFOT.Encrypt outputs a ciphertext c.

- Decryption: $\mathsf{RFOT.Decrypt}(\mathsf{pp}, \mathsf{sk}, c) \to m'$

  On input a public parameter pp, a secret key sk, and a ciphertext c, RFOT.Decrypt outputs a message m'.

The simulator regarding the security for the scheme

- Simulated setup: $\mathsf{RFOT.SimSetup}(1^\lambda) \to (\mathsf{pp}, \mathsf{td})$

  On input a security parameter λ, RFOT.SimSetup outputs a public parameter pp and a trapdoor td.

- Simulated key generation: $\mathsf{RFOT.SimKeygen}(\mathsf{pp}, \mathsf{td}, p) \to (\mathsf{pk}, \mathsf{sk})$

  On input a public parameter pp, a trapdoor td, and a probability $p \in [0, 1]$, RFOT.SimKeygen outputs a public key pk and a secret key sk.

- Simulated encryption: $\mathsf{RFOT.SimEncrypt}(\mathsf{pp}, \mathsf{td}, \mathsf{pk}, b, m') \to c$

  On input a public parameter pp, a trapdoor td, a public key pk, a bit b, and a message m', RFOT.SimEncrypt outputs a ciphertext c.

- Extracting decryption: $\mathsf{RFOT.ExtDecrypt}(\mathsf{pp}, \mathsf{td}, \mathsf{sk}, c) \to m$

  On input a public parameter pp, a trapdoor td, a secret key sk, and a ciphertext c, RFOT.ExtDecrypt outputs a message m.

- Simulated decryption: $\mathsf{RFOT.SimDecrypt}(\mathsf{pp}, \mathsf{td}, \mathsf{sk}, b) \to \mathsf{sk}'$

  On input a public parameter pp, a trapdoor td, a secret key sk, and a bit b, RFOT.SimDecrypt outputs a simulated secret key sk′.

Figure 5.5: Syntax of the scheme and its simulator [CGL+16]

Fractional hiding and binding

Let us introduce the definitions of fractional hiding and binding, according to [CGL+16].

Definition 5.3 (Fractional hiding (Claim A.7 in [CGL+16]))**.** *The hiding property holds when the following distribution ensembles* $\mathcal{E}_H^{\mathsf{real}}$ *and* $\mathcal{E}_H^{\mathsf{ideal}}$ *are computationally indistinguishable for any adversary* $\mathcal{A}_H$*, where*

$$\begin{aligned}\mathcal{E}_H^{\mathsf{real}} = \{\mathsf{out} \mid\ & \mathsf{RFOT.Setup}(1^\lambda) \to \mathsf{pp}, \mathcal{A}_H(\mathsf{pp}) \to (\mathsf{pk}, m),\\ & \mathsf{RFOT.Encrypt}(\mathsf{pp}, \mathsf{pk}, m) \to c, \mathcal{A}_H(c) \to \mathsf{out}\}\end{aligned}$$

and

$$\begin{aligned}\mathcal{E}_H^{\mathsf{ideal}} = \{\mathsf{out} \mid\ & \mathsf{RFOT.SimSetup}(1^\lambda) \to (\mathsf{pp}, \mathsf{td}), \mathcal{A}_H(\mathsf{pp}) \to (\mathsf{pk}, m),\\ & \text{"1 } \textit{with probability } p \textit{ and } 0 \textit{ otherwise"} \to b,\\ & \textit{if } b = 1\textit{, set } m' = m\textit{; else set } m' = \emptyset,\\ & \mathsf{RFOT.SimEncrypt}(\mathsf{pp}, \mathsf{td}, \mathsf{pk}, b, m') \to c, \mathcal{A}_H(c) \to \mathsf{out}\}.\end{aligned}$$

Definition 5.4 (Fractional binding (Claim A.8 in [CGL+16]))**.** *The binding property holds when the following distribution ensembles* $\mathcal{E}_B^{\mathsf{real}}$ *and* $\mathcal{E}_B^{\mathsf{ideal}}$ *are computationally indistinguishable for any adversary* $\mathcal{A}_B$*, where*

$$\begin{aligned}\mathcal{E}_B^{\mathsf{real}} = \{(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}, c, m) \mid\ & \mathsf{RFOT.Setup}(1^\lambda) \to \mathsf{pp}, \mathsf{RFOT.Keygen}(\mathsf{pp}, p) \to (\mathsf{pk}, \mathsf{sk}),\\ & \mathcal{A}_B(\mathsf{pp}, \mathsf{pk}) \to c, \mathsf{RFOT.Decrypt}(\mathsf{pp}, \mathsf{sk}, c) \to m\}\end{aligned}$$

and

$$\begin{aligned}\mathcal{E}_B^{\mathsf{ideal}} = \{(\mathsf{pp}, \mathsf{pk}, \mathsf{sk}', c, m') \mid\ & \text{"1 } \textit{with probability } p \textit{ and } 0 \textit{ otherwise"} \to b,\\ & \mathsf{RFOT.SimSetup}(1^\lambda) \to (\mathsf{pp}, \mathsf{td}),\\ & \mathsf{RFOT.SimKeygen}(\mathsf{pp}, \mathsf{td}, p) \to (\mathsf{pk}, \mathsf{sk}),\\ & \mathcal{A}_B(\mathsf{pp}, \mathsf{pk}) \to c, \mathsf{RFOT.ExtDecrypt}(\mathsf{pp}, \mathsf{td}, \mathsf{sk}, c) \to m,\\ & \mathsf{RFOT.SimDecrypt}(\mathsf{pp}, \mathsf{td}, \mathsf{sk}, b) \to \mathsf{sk}',\\ & \textit{if } b = 1\textit{, set } m' = m\textit{; else set } m' = \emptyset\}.\end{aligned}$$

5.2.4 Construction

We describe our proposed scheme, its correctness, and its simulation-based security.

Scheme

Let us introduce the formal definition of our scheme.

Definition 5.5 (Ring fractional oblivious transfer)**.** *The ring fractional oblivious transfer (*RFOT*)*

$$\mathsf{RFOT} = (\mathsf{RFOT.Setup}, \mathsf{RFOT.Keygen}, \mathsf{RFOT.Encrypt}, \mathsf{RFOT.Decrypt})$$

is defined in Fig. 5.6.

Remark 5.3. *We denote the encryption as* $c = \mathsf{RFOT.Encrypt}(\mathsf{pk}, m; v, e, f)$ *if an encryptor specifies the randomness* v, e, f*.*

Correctness

Let us confirm the correctness of decryption in our scheme. If $l = k$, then

$$\begin{aligned}(c_1 - c_2 s)X^{-k} &= ((yX^{-k}v + 2e + m)X^l - (av + 2f)s)X^{-k} \\ &= ((yX^{l-k} - as)v + 2(eX^l - fs))X^{-k} + mX^{l-k} \\ &= ((y - as)v + 2(eX^l - fs))X^{-k} + m \\ &= 2(e_s v + eX^l - fs)X^{-k} + m\end{aligned}$$

$(e_s v + eX^l - fs)X^{-k}$ is small. We have $m' = (c_1 - c_2 s)X^{-k} \mod 2 = m$.

Security

Let us state the simulation-based security.

- $\mathsf{RFOT.Setup}(1^\lambda) \to \mathsf{pp}$
  1. Compute $\mathsf{crs}_K \leftarrow \mathsf{NIZK.Setup}(1^\lambda, \mathcal{R}_K)$.
  2. Compute $\mathsf{crs}_E \leftarrow \mathsf{NIZK.Setup}(1^\lambda, \mathcal{R}_E)$.
  3. Set $a \xleftarrow{\$} \mathbf{R}_q$.
  4. Output $\mathsf{pp} = (\mathsf{crs}_K, \mathsf{crs}_E, a)$.
- $\mathsf{RFOT.Keygen}(\mathsf{pp}, 1/n) \to (\mathsf{pk}, \mathsf{sk})$
  1. Parse pp as $(\mathsf{crs}_K, \mathsf{crs}_E, a)$.
  2. Set $s, e_s \xleftarrow{\$} \chi$.
  3. Compute $y = as + 2e_s$.
  4. Set $k \xleftarrow{\$} \{1, \ldots, n\}$.
  5. Compute $y_0 = yX^{-k}$.
  6. Compute $\pi_K = \mathsf{NIZK.Prove}(\mathsf{crs}_K, (a, y_0), (sX^{-k}, e_sX^{-k}))$.
  7. Set $\mathsf{pk} = (1/n, a, y_0, \pi_K)$.
  8. Set $\mathsf{sk} = (1/n, s, e_s, k, \pi_K)$.
  9. Output key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{RFOT.Encrypt}(\mathsf{pp}, \mathsf{pk}, m) \to c$
  1. Parse pp as $(\mathsf{crs}_K, \mathsf{crs}_E, a)$.
  2. Parse pk as $(1/n, a, y_0, \pi_K)$.
  3. Set $l \xleftarrow{\$} \{1, \ldots, n\}$.
  4. Compute $c_1' = y_0v + 2e + m,\ c_2 = av + 2f$.
  5. Compute $c_1 = c_1'X^l = (y_0v + 2e + m)X^l$.
  6. Compute $\pi_E = \mathsf{NIZK.Prove}(\mathsf{crs}_E, (c_1, c_2, a, y_0), (m, v, e, f))$.
  7. Output $c = (l, c_1, c_2, \pi_E)$.
- $\mathsf{RFOT.Decrypt}(\mathsf{pp}, \mathsf{sk}, c) \to m'$
  1. Parse pp as $(\mathsf{crs}_K, \mathsf{crs}_E, a)$.
  2. Parse sk as $(1/n, s, e_s, k, \pi_K)$.
  3. Parse c as (l, c_1, c_2, π_E).
  4. If $l \notin \{1, \ldots, n\}$, then output $\perp$.
  5. If $\mathsf{NIZK.Verify}(\mathsf{crs}_E, (c_1, c_2, a, y_0), \pi_E) = 0$, then output $\perp$.
  6. If $l \neq k$, then output $m' = \emptyset$.
  7. If $l = k$, then output $m' = (c_1 - c_2s)X^{-k} \mod 2$.

Figure 5.6: Construction of the ring fractional oblivious transfer

Definition 5.6 (Simulated ring fractional oblivious transfer)**.** *The simulated ring fractional oblivious transfer*

$$\mathsf{RFOT}^{\mathsf{sim}} = (\mathsf{RFOT.SimSetup}, \mathsf{RFOT.SimKeygen}, \mathsf{RFOT.SimEncrypt}, \mathsf{RFOT.ExtDecrypt}, \mathsf{RFOT.SimDecrypt})$$

is defined in Fig. 5.7.

Remark 5.4. *Let us confirm that one can extract the plaintext with* RFOT.ExtDecrypt*, even if* $l \neq k$*.*

$$\begin{aligned} c_1 X^{-l} - c_2 s X^{-k} &= (y_0 v + 2e + m) - (av + 2f) s X^{-k} \\ &= (y X^{-k} v + 2e + m) - (av + 2f) s X^{-k} \\ &= (y - as) v X^{-k} + 2(e - f s X^{-k}) + m \\ &= 2(e_s v + e - fs) X^{-k} + m \end{aligned}$$

$(e_s v + e - fs) X^{-k}$ *is small. We obtain* $(c_1 X^{-l} - c_2 s X^{-k}) \mod 2 = m$*.*

Lemma 5.2. *The below two distribution ensembles are computationally indistinguishable:*

$$\{\mathsf{pp} \mid \mathsf{RFOT.Setup}(1^\lambda) \to \mathsf{pp}\} \text{ and } \{\mathsf{pp} \mid \mathsf{RFOT.SimSetup}(1^\lambda) \to (\mathsf{pp}, \mathsf{td})\}.$$

Proof. We suppose that an adversary distinguishes the two pp. Let us confirm the variable $a \in \mathsf{pp}$. The one is randomly sampled from $\mathbf{R}_q$, and the other is equal to $g_a \cdot f_a^{-1}$. If the adversary distinguishes the two a, then the adversary could break the DSPR assumption regarding Definition 2.2. It is a contradiction. We conclude that RFOT.Setup and RFOT.SimSetup are computationally indistinguishable. □

Lemma 5.3. *The below two distribution ensembles are computationally indistinguishable:*

$$\{(\mathsf{pp}, \mathsf{pk}) \mid \mathsf{RFOT.Setup}(1^\lambda) \to \mathsf{pp},\ \mathsf{RFOT.Keygen}(\mathsf{pp}, p) \to (\mathsf{pk}, \mathsf{sk})\} \text{ and}$$
$$\{(\mathsf{pp}, \mathsf{pk}) \mid \mathsf{RFOT.SimSetup}(1^\lambda) \to (\mathsf{pp}, \mathsf{td}),\ \mathsf{RFOT.SimKeygen}(\mathsf{pp}, \mathsf{td}, p) \to (\mathsf{pk}, \mathsf{sk})\}.$$

Proof. The difference between RFOT.Keygen and RFOT.SimKeygen is each function that outputs each proof. One is NIZK.Prove. The other is NIZK.Simulate. The proofs by

- $\mathsf{RFOT.SimSetup}(1^\lambda) \to (\mathsf{pp}, \mathsf{td})$
  1. Compute $(\mathsf{crs}_K, \mathsf{td}_K) \leftarrow \mathsf{NIZK.SimSetup}(1^\lambda, \mathcal{R}_K)$.
  2. Compute $(\mathsf{crs}_E, \mathsf{td}_E) \leftarrow \mathsf{NIZK.SimSetup}(1^\lambda, \mathcal{R}_E)$.
  3. Compute $(a, \mathbf{B}, g_a, f_a) = \mathsf{IBE.MasterKeygen}(d, q)$.
  4. Set $\mathsf{td}_I = (\mathbf{B}, g_a, f_a)$.
  5. Set $\mathsf{pp} = (\mathsf{crs}_K, \mathsf{crs}_E, a)$.
  6. Set $\mathsf{td} = (\mathsf{td}_K, \mathsf{td}_E, \mathsf{td}_I)$.
  7. Output $(\mathsf{pp}, \mathsf{td})$.
- $\mathsf{RFOT.SimKeygen}(\mathsf{pp}, \mathsf{td}, 1/n) \to (\mathsf{pk}, \mathsf{sk})$
  1. Parse pp as $(\mathsf{crs}_K, \mathsf{crs}_E, a)$.
  2. Parse td as $(\mathsf{td}_K, \mathsf{td}_E, \mathsf{td}_I)$.
  3. Set $s, e_s \xleftarrow{\$} \chi$.
  4. Compute $y = as + 2e_s$.
  5. Set $k \xleftarrow{\$} \{1, \ldots, n\}$.
  6. Compute $y_0 = yX^{-k}$.
  7. Compute $\pi_K = \mathsf{NIZK.Simulate}(\mathsf{crs}_K, \mathsf{td}_K, (a, y_0))$.
  8. Set $\mathsf{pk} = (1/n, a, y_0, \pi_K)$.
  9. Set $\mathsf{sk} = (1/n, s, e_s, k, \pi_K)$.
  10. Output key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{RFOT.SimEncrypt}(\mathsf{pp}, \mathsf{td}, \mathsf{pk}, b, m') \to c$
  1. Parse pp as $(\mathsf{crs}_K, \mathsf{crs}_E, a)$.
  2. Parse td as $(\mathsf{td}_K, \mathsf{td}_E, \mathsf{td}_I)$.
  3. Parse pk as $(1/n, a, y_0, \pi_K)$.
  4. Set $l \xleftarrow{\$} \{1, \ldots, n\}$.
  5. If $b = 0$, then
     - Set $(c_1, c_2) \xleftarrow{\$} \mathbf{C}$.
     - Compute $\pi_E = \mathsf{NIZK.ExtSimulate}(\mathsf{crs}_E, \mathsf{td}_I, (c_1, c_2, a, y_0))$.
  6. If $b = 1$, then
     - Compute $c_1' = y_0 v + 2e + m',\ c_2 = av + 2f,\ c_1 = c_1' X^l$.
     - Compute $\pi_E = \mathsf{NIZK.Prove}(\mathsf{crs}_E, (c_1, c_2, a, y_0), (m', v, e, f))$.
  7. Output $c = (l, c_1, c_2, \pi_E)$.
- $\mathsf{RFOT.ExtDecrypt}(\mathsf{pp}, \mathsf{td}, \mathsf{sk}, c) \to m$
  1. Parse sk as $(1/n, s, e_s, k, \pi_K)$.
  2. Parse c as (l, c_1, c_2, π_E).
  3. Output $m = (c_1 X^{-l} - c_2 s X^{-k}) \mod 2$.
- $\mathsf{RFOT.SimDecrypt}(\mathsf{pp}, \mathsf{td}, \mathsf{sk}, b) \to \mathsf{sk}'$
  1. Parse pp as $(\mathsf{crs}_K, \mathsf{crs}_E, a)$.
  2. Parse td as $(\mathsf{td}_K, \mathsf{td}_E, \mathsf{td}_I)$.
  3. Parse td_I as $(\mathbf{B}, g_a, f_a)$.
  4. Parse sk as $(1/n, s, e_s, k, \pi_K)$.
  5. If $b = 0$,
     - Compute $(s', e_s') = \mathsf{IBE.Extract}(\mathbf{B}, y_0)$
     - Set $\mathsf{sk}' = (1/n, s', e_s', k, \pi_K)$.
  6. If $b = 1$, then output sk.

Figure 5.7: Simulator of the ring fractional oblivious transfer

NIZK.Prove and NIZK.Simulate are computationally indistinguishable because NIZK scheme satisfies zero-knowledge. We conclude that the lemma holds. □

Lemma 5.4 (Fractional hiding)**.** *The scheme of Definition 5.5 holds the fractional hiding of Definition 5.3.*

Proof. Let us compare $\mathcal{E}_H^{\mathsf{real}}$ with $\mathcal{E}_H^{\mathsf{ideal}}$. There are two differences between the two ensembles. The one is NIZK.Prove or NIZK.ExtSimulate. These proofs are indistinguishable from Lemma 5.1.

The other difference is the encryption or random sampling. $c_1' = y_0 v + 2e + m', c_1 = c_1' X^l, c_2 = av + 2f$ in $\mathcal{E}_H^{\mathsf{real}}$ or $(c_1, c_2) \xleftarrow{\$} \mathbf{C}$ in $\mathcal{E}_H^{\mathsf{ideal}}$. If an adversary $\mathcal{A}$ can distinguish $\mathcal{E}_H^{\mathsf{real}}$ and $\mathcal{E}_H^{\mathsf{ideal}}$, then $\mathcal{A}$ could distinguish the true ciphertext (c_1, c_2) from the randomly sampled element $(c_1, c_2) \xleftarrow{\$} \mathbf{C}$. This situation goes against the RLWE assumption regarding Definition 2.1. We conclude that $\mathcal{E}_H^{\mathsf{real}}$ and $\mathcal{E}_H^{\mathsf{ideal}}$ are indistinguishable. The scheme holds the fractional hiding.

□

Lemma 5.5 (Fractional binding)**.** *The scheme of Definition 5.5 holds the fractional hiding of Definition 5.4.*

Proof. Let us compare $\mathcal{E}_B^{\mathsf{real}}$ with $\mathcal{E}_B^{\mathsf{ideal}}$. RFOT.Keygen and RFOT.SimKeygen are computationally indistinguishable because of Lemma 5.3. RFOT.ExtDecrypt outputs a unique plaintext m for a valid ciphertext even if $l \neq k$. The simulated secret key sk' is distributed independently of the actual secret key sk. When $b = 1$, output the actual secret key itself. When $b = 0$, a different secret key (s', e_s') is output, satisfying the relationship $y_0 = as' + 2e_s'$ with the fixed public key (a, y_0) with IBE.Extract. We conclude that the scheme holds the fractional binding. □

5.3 Protocol and security

This section presents the protocols, the ideal functionalities, the theorems that each protocol realizes each ideal functionality, and the proofs regarding the puzzle solver and puzzle promise. We present a simulation-based proof of the real/ideal world paradigm. We build simulators in the cases of each corrupt participant. We discuss the indistinguishability of the game sequence by using a hybrid argument. The discussion is based on

TumbleBit [HAB+17, HAB+16]. We also combine the DAM scheme [CGL+17, CGL+16] regarding the puzzle solver protocol related to RFOT.

5.3.1 Puzzle solver protocol

We present the puzzle solver protocol, the ideal functionality, the theorem, and the proof. We show the puzzle solver protocol in Fig. 5.8 and the ideal functionality in Fig. 5.9. We also show the simulators in Fig. 5.10 and Fig. 5.11. The simulator in Fig. 5.10 corresponds to the corrupt Alice. The simulator in Fig. 5.11 corresponds to the corrupt the tumbler. Let us present the quick look at the protocol, the overview of the ideal functionality, and proof sketch as the below paragraphs.

Quick look at the protocol. We change the procedure by the RSA encryption in TumbleBit [HAB+17, HAB+16], into the one by the RLWE encryption. Also, we combine RFOT with the procedure. The tumbler and Alice interact at the protocol. We use a cut-and-choose technique. Alice creates real puzzles and fake puzzles, respectively.

At Step 1, She further masks the masked puzzle (ciphertext) she receives for the real puzzles. She creates ciphertext by randomly choosing a plaintext by herself. She adds this to the puzzle. At Step 2, she creates a ciphertext from randomly chosen plaintext for the fake puzzles. She does not include the received puzzle in the fake puzzles. At Step 3, she mixes these puzzles and executes the RFOT encryption. She permutes and mixes real puzzles and fake puzzles. These puzzles are just the RLWE ciphertexts. She chooses the integer l randomly and shifts all puzzles by the single amount l. In this way, she produces the RFOT ciphertext from the RLWE ciphertexts. She passes the puzzles to the tumbler. At Step 4, the tumbler decrypts all the puzzles it receives. If the puzzles are miss/error, then the tumbler cannot obtain the correct message. In this case, the tumbler aborts. (Even if the tumbler chooses s randomly, she will notice that it does not match her plaintext at Step 7. Moreover, the tumbler does not know which puzzle is in the fake set at this stage.) The tumbler creates a ciphertext c for the decrypted message by using the symmetric key encryption. The tumbler also selects the hash value h of k randomly. The tumbler sends these (c, h) to her for commitment. At Step 5, upon receiving the commitment (c, h), she informs the tumbler of which puzzle belongs to the fake set. She sends a message and randomness to the tumbler for opening. At Step 6,

the tumbler verifies the received message and randomness, by matching the ciphertext received earlier. If all can be verified correctly, the tumbler sends k belonging to the fake set to her. At Step 7, she obtains s by the decryption process from the received k and the ciphertext c of the symmetric key encryption. If s matches the original plaintext, she continues. Otherwise, she aborts. At Step 8, she fills in the transaction on the blockchain. Both the preimages of h and the tumbler's signature can fulfill this transaction. She sends the received puzzle to the tumbler and the message and the randomness in the real set. At Step 9, the tumbler verifies the ciphertext from the received message, randomness, and puzzle. If it is OK, the tumbler fills k in the transaction offered at Step 10. At Step 11, she gets k from the transaction, executes the symmetric key encryption, and gets s. She removes her message by the XOR operation and gets the message of the received puzzle y.

Overview of the ideal functionality. We combine the functionality of TumbleBit and the functionality of fractional message transfer (FMT) in the DAM scheme for the ideal functionality. First, we incorporate mutual fairness into functionalities, following TumbleBit. Fairness for Alice means that the tumbler earns one coin if and only if she gets the correct answer. Fairness for the tumbler means that the protocol executes decrypting the puzzle selected by her. Upon setup, the functionality receives the key from the tumbler and verifies if it is valid. If the verification is successful, the functionality sends the key to her and the simulator. The functionality receives the request containing the ciphertext from her and sends it to the tumbler at the evaluation. The functionality receives the plaintext result from the tumbler, sends it to Alice, and pays to the tumbler.

Next, we append a probabilistic payment to the functionality, following the DAM scheme. The functionality conveys a valid message with the probability p. Alice sends to the functionality at the request, including the bit b and the probability $p_{\mathcal{A}}$. The functionality confirms the bit b to determine whether there is an error. The tumbler sends the key to the functionality together with the probability $p_{\mathcal{T}}$. The functionality judges that a lottery is an error in the case that $b = 0$ or the probabilities do not match. If $b = 1$ and the probabilities are equal, the functionality runs that the ciphertext is a hit with the probability p and is a failure with the probability $1 - p$. The functionality stores the judged value in $\mathcal{Q}$ as a pair $(\mathsf{sid}, \mathsf{mid})$. Let us confirm the behavior after the functionality receives the evaluation result from the tumbler. The functionality searches

$\mathcal{Q}$ for mid paired with sid. If mid is a hit and the message x from the tumbler is valid, the functionality sends her x and pays the tumbler. Otherwise, the tumbler refunds to her as a missing lottery or an error.

Proof sketch. In the case of corrupt Alice, she cannot learn more than the decrypted message. Let us confirm the case of corrupt the tumbler. The protocol adopts a cut-and-choose technique like TumbleBit. Therefore, the tumbler must present invalid ciphertexts to the real set while correctly responding to the fake set. The tumbler cannot obtain the information regarding the fake set and the real set before her opening. The probability of corrupt tumbler's success is negligible.

Theorem 5.1. *Let λ be a security parameter. Let $d \geq 2\lambda$. Assume that H and H^{prg} are independent random oracles, and the RLWE and DSPR problems are hard. Then, the protocol in Fig. 5.8 securely realizes the functionality $\mathcal{F}_{\mathsf{solver}}$ in Fig. 5.9 with the following security guarantees. The security for $\mathcal{T}$ is $1 - \mathsf{negl}(\lambda)$ and the security for $\mathcal{A}$ is $1 - 1/\binom{\mu+\eta}{\eta} - \mathsf{negl}(\lambda)$.*

Proof. We divide the proof into two cases. One is the case of the corrupt Alice. The other is the case of the corrupt the tumbler. The simulator S in Fig. 5.10 corresponds to the corrupt Alice $\mathcal{A}^*$. The simulator S in Fig. 5.11 corresponds to the corrupt the tumbler $\mathcal{T}^*$.

Case that Alice is corrupt. We adopt a hybrid argument and confirm the indistinguishability between the real world and the ideal world. The simulator S in Fig. 5.10 plays a role of the corrupt Alice $\mathcal{A}^*$.

$\ni_0$: This is the real game.

$\ni_1$: The difference between $\ni_0$ and $\ni_1$ is how the key k_i is computed. If $x' = \mathsf{RFOT.Decrypt}(\mathsf{sk}, \beta_i) \neq \emptyset$ with salt, the simulator in this hybrid $\ni_1$ sends $k'_i = \mathsf{Equivocate}(c_i, x, h_i)$, instead of k_i. If $x' = \emptyset$, then both games receive refund and halt. From the assumption regarding the random oracle, c_i and h_i statistically hide the encrypted message and the preimage. The probability of the event $\mathsf{Collision}$ is negligible. So the view in $\ni_1$ is indistinguishable from the view in $\ni_0$.

$\ni_2$: The difference between $\ni_1$ and $\ni_2$ is what one inputs $\mathsf{Equivocate}$. The simulator in this hybrid $\ni_2$ computes $\mathsf{Equivocate}(c_i, \rho_i, h_i)$ instead of $\mathsf{Equivocate}(c_i, x + r_i, h_i)$. Then

Public input: pk. π_K in pk proves validity of pk in a one time setup phase.

Alice $\mathcal{A}$. Input: puzzle y		the tumbler $\mathcal{T}$. Secret input: sk
1. Prepare real puzzles R $\mathsf{salt} \xleftarrow{\$} \{0,1\}^\lambda$ For $j = 1, \ldots, \mu$: $r_j \xleftarrow{\$} \{0,1\}^\lambda, v_j, e_j, f_j \xleftarrow{\$} \chi$ $d_j = \mathsf{RLWE.Enc}(\mathsf{pk}, \mathsf{salt}\|\|r_j; v_j, e_j, f_j) + y$		
2. Prepare fake values F For $i = 1, \ldots, \eta$: $\rho_i \xleftarrow{\$} \{0,1\}^\lambda, v_i, e_i, f_i \xleftarrow{\$} \chi$ $\delta_i = \mathsf{RLWE.Enc}(\mathsf{pk}, \mathsf{salt}\|\|\rho_i; v_i, e_i, f_i)$		
3. Mix sets and RFOT Permute $\{d_1, \ldots, d_\mu, \delta_1, \ldots, \delta_\eta\}$ randomly and obtain $\{\beta'_1, \ldots, \beta'_{\mu+\eta}\}$ $l \xleftarrow{\$} \{1, \ldots, n\}$ For $i = 1, \ldots, \mu + \eta$: Parse β'_i as (c'_1, c_2) and set $c_1 = c'_1 X^l$ Set $\beta_i = (c'_1, c_2)$ Let R be the indices of the d_i Let F be the indices of the δ_i	$\xrightarrow{\mathsf{salt}, l, \beta_1, \ldots, \beta_{\mu+\eta}}$	
		4. Evaluation. For $i = 1, \ldots, \mu + \eta$: Evaluate β_i: $s_i = \mathsf{RFOT.Decrypt}(\mathsf{sk}, \beta_i)$ Check if $s_i = \emptyset$ with salt. If not, abort. Encrypt the result s_i: $k_i \xleftarrow{\$} \{0,1\}^{\lambda_1}$ $c_i = H^{\mathsf{prg}}(k_i) \oplus s_i$ Commit to the keys: $h_i = H(k_i)$
	$\xleftarrow{(c_1,h_1),\ldots,(c_{\mu+\eta},h_{\mu+\eta})}$	
5. Identify fake set.	$\xrightarrow{F, \rho_i, v_i, e_i f_i \ \forall i \in F}$	
		6. Check fake set. For all $i \in F$: Verify $\beta_i =$ $\mathsf{RFOT.Encrypt}(\mathsf{pk}, \mathsf{salt}\|\|\rho_i; v_i, e_i, f_i, l)$ If yes, reveal $k_i \ \forall i \in F$. Else abort.
	$\xleftarrow{k_i \ \forall i \in F}$	
7. Check fake set. For all $i \in F$: Verify $h_i = H(k_i)$ Decrypt $s_i = H^{\mathsf{prg}}(k_i) \oplus c_i$ Verify $s_i = \rho_i$ Abort if any check fails		
8. Post transaction T_{puzzle} T_{puzzle} offers one coin within timewindow tw_1 under condition "the fulfilling transaction is signed by $\mathcal{T}$ and has preimages of $h_j \ \forall j \in R$".	$\xrightarrow{y, r_j, v_j, e_j, f_j \ \forall j \in R}$	
		9. Check β_i unblind to $y \ \forall j \in R$ For all $j \in R$: Verify $\beta_i =$ $\mathsf{RFOT.Encrypt}(\mathsf{pk}, \mathsf{salt}\|\|r_j; v_j, e_j, f_j, l) + y$ If not, abort.
		10. Post transaction T_{puzzle} T_{puzzle} contains $k_j \ \forall j \in R$
11. Obtain puzzle solution. For $j \in R$: Learn k_j from T_{puzzle} Decrypt $s_j = H^{\mathsf{prg}}(k_j) \oplus c_j$ Verify $\beta_j = \mathsf{RFOT.Encrypt}(\mathsf{pk}, \mathsf{salt}\|\|s_j; v_j, e_j, f_j, l) + y$ Obtain solution $s_j + r_j \mod 2$		

Figure 5.8: Puzzle solver protocol. We model H and H^{prg} as random oracles.

Parties.

- $\mathcal{A}, \mathcal{T}$ and adversary $\mathcal{S}$.

Setup.

- Receive $(\mathsf{Setup}, \mathsf{pk}, \mathsf{sk}, p_{\mathcal{T}})$ from $\mathcal{T}$.
- If pk or sk are invalid, then
  - do nothing.
- Else,
  - Send $(\mathsf{Setup}, \mathsf{pk})$ to $\mathcal{A}$ and $\mathcal{S}$.

Evaluation.

- On input $(\mathsf{request}, \mathsf{sid}, y, 1\mathsf{coin}, p_{\mathcal{A}}, b)$ from $\mathcal{A}$:
  - If $b = 1$ and $p_{\mathcal{A}} = p_{\mathcal{T}}$,
    - Set $\mathsf{mid} = \mathsf{hit}$ with probability $p_{\mathcal{A}}$ or $\mathsf{mid} = \mathsf{miss}$ with probability $1 - p_{\mathcal{A}}$.
  - Else if $b = 0$ or $p_{\mathcal{A}} \neq p_{\mathcal{T}}$, set $\mathsf{mid} = \mathsf{error}$.
  - Send $(\mathsf{request}, \mathsf{sid}, \mathcal{A}, y)$ to $\mathcal{T}$.
  - Append $(\mathsf{sid}, \mathsf{mid})$ to $\mathcal{Q}_{\mathsf{id}}$.
  - Start counter $tw_{\mathsf{sid}} = 0$.
- On input $(\mathsf{evaluate}, \mathsf{sid}, \mathcal{A}, x)$ from $\mathcal{T}$:
  - Obtain mid corresponding sid from $\mathcal{Q}_{\mathsf{id}}$.
  - If $\mathsf{mid} = \mathsf{hit}$ and $x \neq \emptyset$, then
    - Send (sid, x) to $\mathcal{A}$.
    - Send $(\mathsf{payment}, \mathsf{sid}, 1\mathsf{coin})$ to $\mathcal{T}$.
  - Else, send $(\mathsf{refund}, \mathsf{sid}, 1\mathsf{coin})$ to $\mathcal{A}$.
- If $tw_{\mathsf{sid}} = tw$, send $(\mathsf{refund}, \mathsf{sid}, 1\mathsf{coin})$ to $\mathcal{A}$.

Figure 5.9: Ideal functionality $\mathcal{F}_{\mathsf{solver}}$

1. Receive $\mathsf{salt}, l, \beta_1, \ldots, \beta_{\mu+\eta}$ from Adv. Choose $c_i \xleftarrow{\$} \{0,1\}^{\lambda}$ and $h_i \in \{0,1\}^{\lambda_2}$ for $i \in [\mu + \eta]$. Send them to Adv.

2. Receive F, ρ_i, v_i, e_i, f_i for $i \in F$. For all $i \in F$, check if $\beta_i = \mathsf{RFOT.Encrypt}(\mathsf{pk}, \mathsf{salt}||\rho_i; v_i, e_i, f_i, l)$. If check fails, output whatever Adv outputs, send $(\mathsf{request}, \mathsf{sid}, y, 1\mathsf{coin}, p_{\mathcal{A}}, b = 0)$ to $\mathcal{F}_{\mathsf{solver}}$ and halt. Else, run $k_i' = \mathsf{Equivocate}(c_i, \rho_i, h_i)$. Send k_i' for $i \in F$ to Adv.

3. Receive y, r_i, v_i, e_i, f_i for $i \in R$. Check if $\beta_i = \mathsf{RFOT.Encrypt}(\mathsf{pk}, \mathsf{salt}||r_i; v_i, e_i, f_i, l) + y$ for all $i \in R$. If the check succeeds, transaction T_{puzzle} is correctly formed, execute as follows.

   - Send $(\mathsf{request}, \mathsf{sid}, y, 1\mathsf{coin}, p_{\mathcal{A}}, b = 1)$ to $\mathcal{F}_{\mathsf{solver}}$ and obtain x or refund.
   - If obtain x,
     - Run $k_i' = \mathsf{Equivocate}(c_i, x + r_i, h_i)$ with $i \in R$.
     - Send transaction T_{solve} with values k_i'.
   - Else if obtain refund, halt.

   Else, checks have failed so output whatever Adv outputs, send $(\mathsf{request}, \mathsf{sid}, y, 1\mathsf{coin}, p_{\mathcal{A}}, b = 0)$ to $\mathcal{F}_{\mathsf{solver}}$ and halt.

Procedure random oracle simulation for $\mathcal{Q}_H, \mathcal{Q}_{H^{\mathsf{prg}}}$ is as follows:
Receive query q for $H(\mathsf{resp.}, H^{\mathsf{prg}})$:

1. If query $q \in \mathcal{Q}_H(\mathsf{resp.}, H^{\mathsf{prg}})$, retrieve entry (q, a) from the set and output a.

2. Else $a \xleftarrow{\$} \{0,1\}^{\lambda_2}(\mathsf{resp.}, \lambda_1)$, and (q, a) to $\mathcal{Q}_H(\mathsf{resp.}, H^{\mathsf{prg}})$ and output a.

Procedure $\mathsf{Equivocate}(c_i, m_i, h_i)$ is stated as below:

1. $k_i' \xleftarrow{\$} \{0,1\}^{\lambda_1}$. If $k_i' \in \mathcal{Q}_H$ or $\mathcal{Q}_{H^{\mathsf{prg}}}$, output $\mathsf{Collision}$ and abort.

2. Compute $a_i = c_i \oplus m_i$, then append (k_i', a_i) to $\mathcal{Q}_{H^{\mathsf{prg}}}$.

3. Append (k_i', h_i) to $\mathcal{Q}_H$.

4. Output k_i'.

Figure 5.10: Simulator for the puzzle solver protocol in the case that Alice is corrupt

1. If pk and sk are valid, S receives $(\mathsf{request}, \mathsf{sid}, \mathcal{A}, y)$ from $\mathcal{F}_{\mathsf{solver}}$.
2. Compute $x' = \mathsf{RFOT.ExtDecrypt}(\mathsf{pp}, \mathsf{td}, \mathsf{sk}, y)$ and extract x from $x' = \mathsf{salt}||x$.
3. Receiving $\{k_i\}$ for all $i \in F$ from Adv, S checks if all $\{c_i\}_{i \in F}$ are correct. If yes, S sends message $(\mathsf{evaluate}, \mathsf{sid}, \mathcal{A}, x)$ to $\mathcal{F}_{\mathsf{solver}}$. Then, $\mathcal{F}_{\mathsf{solver}}$ sends the puzzle solution x or refund to $\mathcal{A}$. Meanwhile, S sends T_{puzzle} to Adv.
4. S receives T_{solve} from Adv. If all keys $\{k_i\}$ $\forall i \in R$ decrypt ciphertexts c_i, not containing valid puzzle solutions, then S outputs BAD and aborts. Else, S outputs whatever Adv outputs and halts.

Figure 5.11: Simulator for the puzzle solver protocol in the case that the tumbler is corrupt

x is a valid message of y. x is taken from $\mathcal{F}_{\mathsf{solver}}$ after Adv has sent T_{puzzle}. Both games receive refund instead of x and halt. The view in $\ni_2$ is generated by just a single value x. The view in $\ni_2$ is indistinguishable from the view in $\ni_1$. The simulator in $\ni_2$ corresponds to Simulator S in Fig. 5.10.

Case that the tumbler is corrupt. The simulator S in Fig. 5.11 plays a role of the corrupt the tumbler $\mathcal{T}^*$. Because of the RLWE assumption, all ciphertexts $\beta_1, \ldots, \beta_{\mu+\eta}$ are uniformly distributed. They reveal no information about the sets F and R. Furthermore, H and H^{prg} are modeled as a random oracle. The encryption of the key is binding. It is impossible for an adversary Adv to change the values after the sets F and R are revealed. We denote the event that an adversary presumes the set F as the event BAD. The probability of the event BAD is as follows.

$$\Pr[\mathsf{BAD}] = \frac{1}{\binom{\mu+\eta}{\eta}} + \frac{1}{2^{\lambda_1}}$$

The difference between the transcript by the simulator S in Fig. 5.11 in the ideal world and the one in the real world is if the event BAD happens or not. The two worlds are distinguishable with the negligible probability $\Pr[\mathsf{BAD}]$.

We conclude that the protocol in Fig. 5.8 securely realizes the functionality $\mathcal{F}_{\mathsf{solver}}$ in Fig. 5.9. □

5.3.2 Puzzle promise protocol

We present the puzzle promise protocol, the ideal functionality, the theorem, and the proof. We show the protocol in Fig. 5.12 and the ideal functionality in Fig. 5.13. We also show the simulators in Fig. 5.14 and Fig. 5.15. The simulator in Fig. 5.14 corresponds to the corrupt Bob. The simulator in Fig. 5.15 corresponds to the corrupt the tumbler. Let us present the quick look at the protocol, the overview of the ideal functionality, and proof sketch as the below paragraphs.

Quick look at the protocol. The tumbler and Bob interact at the protocol. As with the puzzle solver protocol, we also use the cut-and-choose technique. At Step 1, the tumbler sets up the escrow transaction. From Step 2 to Step 4, Bob creates real and fake hash values and sends the shuffled hash values to the tumbler. At Step 5, the tumbler signs everything does corresponding puzzles and sends these with promises. At Step 6, he opens a fake set to the tumbler. At Step 7, the tumbler verifies them and presents the puzzle answers corresponding to the fake values. At Step 8, he verifies these answers. Note that we must arrange a situation where it is sufficient for him to send one of the real puzzles to Alice. At Step 9, the tumbler sends the difference from other elements and proves that the ciphertext is zero. The tumbler uses the non-interactive zero-knowledge proof for the relation $\mathcal{R}_0$. At Step 10 and 12, he can agree that it is sufficient to send either one to her by verifying the proof and the difference. At Step 11, the tumbler posts the transaction.

The difference from TumbleBit is the adoption of the RLWE encryption instead of the RSA encryption. The RLWE encryption is probabilistic encryption, unlike the RSA encryption. One can verify the ciphertext by its plaintext and randomness by reproducing and verifying it with the encryption algorithm. We make the tumbler encrypt this randomness with the symmetric key encryption and pass it to Bob. Let the key of the symmetric key encryption be the plaintext itself. Even if we adopt the RLWE encryption, we can use the only plaintext to answer the puzzle by this trick.

Overview of the ideal functionality. We show the ideal functionality in Fig. 5.13. This ideal functionality is the same as TumbleBit. Let us outline the ideal functionality briefly. The ideal functionality plays a role as a trusted third party. The functionality

signs a transaction by calling the signature oracle. It is fairness for Bob that he obtains a promise that contains a valid signature for at least one genuine transaction. It is fairness for the tumbler that he has no knowledge of anything but the fake transaction signature. Upon receiving fake and real transactions from him, the functionality stores them and sends fake transactions to the tumbler. Upon receiving a promise from the tumbler, the functionality signs the fake transactions. The functionality records fake transaction signatures and the promises to real transactions. Finally, the functionality sends the promises to him. Upon receiving the signature verification from any party, the functionality tells them that the functionality has already recorded the fake transactions' signatures. Since the functionality does not store real transactions, the functionality records them.

Proof sketch. Regarding corrupt Bob, we change the RSA encryption in the simulator in TumbleBit into the RLWE encryption. We show the simulator in Fig. 5.14. The simulator in Fig. 5.15 is the same as TumbleBit for corrupt the tumbler. The proof in TumbleBit holds in the same way.

Theorem 5.2. *Let λ be a security parameter. Let $d \geq 2\lambda$. Assume that H, H', and H^{shk} are independent random oracles, and the RLWE and DSPR problems are hard. Then, the protocol in Fig. 5.12 securely realizes the functionality $\mathcal{F}_{\mathsf{promise\ sign}}$ in Fig. 5.13 with the following security guarantees. The security for $\mathcal{T}$ is $1 - \mathsf{negl}(\lambda)$ and the security for $\mathcal{B}$ is $1 - 1/\binom{\mu+\eta}{\eta} - \mathsf{negl}(\lambda)$.*

Proof. We divide the proof into two cases. One is the case of corrupt Bob. The other is the case of the corrupt the tumbler.

Case that Bob is corrupt. Let us confirm the indistinguishability between the real world and the ideal world with a hybrid argument. The simulator S in Fig. 5.14 plays a role of the corrupt Bob $\mathcal{B}^*$.

$\ni_0$: This is the real world.

$\ni_{0.5}$: When the simulator receives h_R, h_F and $\{\beta_1, \ldots, \beta_{\mu+\eta}\}$ from $\mathcal{B}^*$, the simulator checks the set of queries $\mathcal{Q}_H$ and extracts the pair $(\mathsf{salt}||R, h_R)$ and $(\mathsf{salt}||F, h_F)$. If there is no pair regarding h_R and h_F, then the simulator aborts. This abort is the difference between $\ni_0$ and $\ni_{0.5}$. The probability of the abort is $1/2^{\lambda_2}$, so the two games are statistically

Public input: $(\mathsf{pk}, \mathsf{PK}_{\mathcal{T}}^{eph})$. π_K in pk proves validity of pk in a one time setup phase.
$\mathcal{T}$ chooses a fresh ephemeral ECDSA-Secp256k1 key $(\mathsf{SK}_{\mathcal{T}}^{eph}, \mathsf{PK}_{\mathcal{T}}^{eph})$.

Bob $\mathcal{B}$		the tumbler $\mathcal{T}$. Secret input: sk
		1. Set up $T_{\mathsf{escr}(\mathcal{T},\mathcal{B})}$ Sign but do not post transaction $T_{\mathsf{escr}(\mathcal{T},\mathcal{B})}$ timelocked for tw_2 offering one coin under the condition: "the fulfilling transaction
	$\xleftarrow{T_{\mathsf{escr}(\mathcal{T},\mathcal{B})}}$	is signed under key $\mathsf{PK}_{\mathcal{T}}^{eph}$ and key $\mathsf{PK}_{\mathcal{B}}$"
2. Prepare μ real unsigned $T_{\mathsf{escr}(\mathcal{T},\mathcal{B})}$. For $i \in 1,\ldots,\mu$: $\rho_i \xleftarrow{\$} \{0,1\}^\lambda,\ T^i_{\mathsf{escr}(\mathcal{T},\mathcal{B})} = \mathsf{CashOutFormat}(\rho_i),\ ht_i = H'(T^i_{\mathsf{fulfill}})$.		
3. Prepare fake set. For $i \in 1,\ldots,\eta$: $r_i \xleftarrow{\$} \{0,1\}^\lambda,\ ft_i = H'(\mathsf{FakeFormat}\|\|r_i)$.		
4. Mix sets. Permute $\{ft_1,\ldots,ft_\eta,ht_1,\ldots,ht_\mu\}$ randomly and obtain $\{\beta_1,\ldots,\beta_{\mu+\eta}\}$	$\xrightarrow{\beta_1,\ldots,\beta_{\mu+\eta}}$	
Let R be the indices of the ht_i Let F be the indices of the ft_i Choose $\mathsf{salt} \xleftarrow{\$} \{0,1\}^\lambda$ $h_R = H(\mathsf{salt}\|\|R),\ h_F = H(\mathsf{salt}\|\|F)$	$\xrightarrow{h_R,h_F}$	
		5. Evaluation. For $i = 1,\ldots,\mu+\eta$: ECDSA sign β_i to get $\sigma_i = \mathsf{Sig}(\mathsf{SK}_{\mathcal{T}}^{eph}, \beta_i)$ $\epsilon_i \xleftarrow{\$} \{0,1\}^\lambda, v_i, e_i, f_i \xleftarrow{\$} \chi$. Create promise $c_i = H^{\mathsf{shk}}(\epsilon_i) \oplus \sigma_i$ Encrypt $\bar{v}_i = \mathsf{Enc}(v_i;\epsilon_i), \bar{e}_i = \mathsf{Enc}(e_i;\epsilon_i)$ and $\bar{f}_i = \mathsf{Enc}(f_i;\epsilon_i)$
	$\xleftarrow{(c_1,z_1,\bar{v}_1,\bar{e}_1,\bar{f}_1),\ldots,(c_{\mu+\eta},z_{\mu+\eta},\bar{v}_{\mu+\eta},\bar{e}_{\mu+\eta},\bar{f}_{\mu+\eta})}$	Create puzzle $z_i = \mathsf{RLWE.Enc}(\mathsf{pk}, 0^\lambda\|\|\epsilon_i; v_i, e_i, f_i)$
6. Identify fake set.	$\xrightarrow{R,F,r_i\ \forall i\in F,\mathsf{salt}}$	
		7. Check fake set. Check $h_R = H(\mathsf{salt}\|\|R)$ and $h_F = H(\mathsf{salt}\|\|F)$ For all $i \in F$: Verify $\beta_i = H'(\mathsf{FakeFormat}\|\|r_i)$
	$\xleftarrow{\epsilon_i\ \forall i\in F}$	Abort if any check fails
8. Check fake set. For all $i \in F$: Validate $\epsilon_i \in \{0,1\}^\lambda$ Decrypt $v_i = \mathsf{Dec}(\bar{v}_i;\epsilon_i), e_i = \mathsf{Dec}(\bar{e}_i;\epsilon_i), f_i = \mathsf{Dec}(\bar{f}_i;\epsilon_i)$ Validate puzzle $z_i = \mathsf{RLWE.Enc}(\mathsf{pk}, 0^\lambda\|\|\epsilon_i; v_i, e_i, f_i)$ Validate promise c_i: Decrypt $\sigma_i = H^{\mathsf{shk}}(\epsilon_i) \oplus c_i$ Verify σ_i such that $(\mathsf{PK}_{\mathcal{T}}^{eph}, H'(ft_i), \sigma_i) = 1$ Abort if any check fails		
		9. Prepare differences For $R = \{j_1,\ldots,j_\mu\}$: Set $d_2 = \epsilon_{j_2} - \epsilon_{j_1},\ldots,d_\mu = \epsilon_{j_\mu} - \epsilon_{j_{\mu-1}}$
	$\xleftarrow{d_2,\ldots,d_\mu,\pi_2,\ldots,\pi_\mu}$	Set π_j proving $z_j + d_j \in \mathbf{C}_0$
10. Difference test. For $R = \{j_1,\ldots,j_\mu\}$: Verify π_j Abort if any check fails		
		11. Post transaction $T_{\mathsf{escr}(\mathcal{T},\mathcal{B})}$
12. Start Payment Phase. Set $z_t = z_{j_1}$. $m_b \xleftarrow{\$} \{0,1\}^\lambda$. Send $z_{tb} = z_t + \mathsf{RLWE.Enc}(\mathsf{pk}, 0^\lambda\|\|m_b)$ to Alice $\mathcal{A}$.		

Figure 5.12: Puzzle promise protocol. We model H, H' and H^{shk} as random oracles.

Parties.

- $\mathcal{B}, \mathcal{T}$ and adversary $\mathcal{S}$.

Setup.

- Inform $\mathcal{F}_{\mathsf{promise\ sign}}$ if $\mathcal{T}$ is corrupt or honest.

Key generation.

- Receive message $(\mathsf{Keygen}, \mathcal{B})$ from $\mathcal{B}$.
- Send $(\mathsf{Keygen}, \mathcal{B})$ to $\mathcal{S}$.
- Receive response $(\mathsf{PK}_{\mathcal{T}}^{eph}, \mathsf{Sig})$ from $\mathcal{S}$.
- Send $(\mathsf{Setup}, \mathsf{PK}_{\mathcal{T}}^{eph})$ to $\mathcal{B}$.
- Record $(\mathsf{PK}_{\mathcal{T}}^{eph}, \mathsf{Sig})$.

Signature request.

- Receive message $(\mathsf{sign\ request}, \mathsf{PK}_{\mathcal{T}}^{eph'}, \{\mathsf{FkTxn}_i\}_{i\in[\eta]}, \{m_i\}_{i\in[\mu]})$ from $\mathcal{B}$.
- If $\mathsf{PK}_{\mathcal{T}}^{eph'} \neq \mathsf{PK}_{\mathcal{T}}^{eph}$, then do nothing.
- If $\forall i, \mathsf{FkTxn}_i$ compiles with FakeFormat, then send $(\mathsf{sign\ request}, \mathcal{B}, \mathsf{PK}_{\mathcal{T}}^{eph}, \{\mathsf{FkTxn}_i\}_{i\in[\eta]})$ to $\mathcal{T}$.
- Else, do nothing.

Promise.

- Receive $(\mathsf{promise}, \mathcal{B}, \mathsf{ans}, \mathsf{Set})$ from $\mathcal{T}$.
- If $\mathsf{ans} = \mathsf{no}$, then set all signatures to $\perp$.
- Else, if $\mathsf{Set} \neq \emptyset$, compute signatures as follows:
  - If $\mathcal{T}$ is honest, Set $\mathsf{FkSign}_i = \mathsf{Sig}(\mathsf{FkTxn}_i, \mathsf{PK}_{\mathcal{T}}^{eph})$ for $i \in [\eta]$.
  - Else $\mathcal{T}$ is corrupt, Send $(\mathsf{Sign}, \mathsf{FkTxn}_i, \mathcal{B})$ to adversary $\mathcal{S}$, and obtain respective signatures.
  - Abort if there is a recorded entry $(\mathsf{FkTxn}_i, \mathsf{FkSign}_i, \mathsf{PK}_{\mathcal{T}}^{eph}, 0)$.
  - Record entries $(\mathsf{FkTxn}_i, \mathsf{FkSign}_i, \mathsf{PK}_{\mathcal{T}}^{eph}, 1)$ and $(m_j, \mathsf{PK}_{\mathcal{T}}^{eph}, \mathsf{promise})$.
- Send $(\mathsf{sign\ promise}, \mathsf{ans})$ to $\mathcal{B}$.

Signature verification.

- Receive $(\mathsf{Verify}, \mathsf{sid}, m, \sigma, \mathsf{PK}_{\mathcal{T}}^{eph'})$ from any party $\mathcal{P}$:
  - If $\mathsf{PK}_{\mathcal{T}}^{eph'} \neq \mathsf{PK}_{\mathcal{T}}^{eph}$, then do nothing.
  - Else, if $\mathcal{T}$ is honest:
    - If there is a recorded entry $(m, \sigma, \mathsf{PK}_{\mathcal{T}}^{eph}, 1)$, then set $\mathsf{ver} = 1$. (completeness condition)
    - If there is no recorded entry $(m, \sigma, \mathsf{PK}_{\mathcal{T}}^{eph}, 1)$, then set $\mathsf{ver} = 0$ and set the entry $(m, \sigma, \mathsf{PK}_{\mathcal{T}}^{eph}, 0)$. (unforgeability condition)
  - Else, if $\mathcal{T}$ is corrupt, then let ver be set by $\mathcal{S}$. (corrupt signer case)
- Send $(\mathsf{Verify}, \mathsf{sid}, m, \sigma, \mathsf{ver})$ from party $\mathcal{P}$.

Figure 5.13: Ideal functionality $\mathcal{F}_{\mathsf{promise\ sign}}$ (Fig. 8 in [HAB+16])

- Simulator S simulates the messages that Adversary $\mathcal{B}^*$ expects from $\mathcal{T}$ as follows.
  1. Inform $\mathcal{F}_{\mathsf{promise\ sign}}$ that $\mathcal{T}$ is honest.
  2. Compute $(\mathsf{PK}_{\mathcal{T}}^{eph}, \mathsf{SK}_{\mathcal{T}}^{eph}) = \mathsf{Keygen}(1^\lambda)$.
  3. Send $\mathsf{PK}_{\mathcal{T}}^{eph}$ to $\mathcal{B}^*$ and $\mathcal{F}_{\mathsf{promise\ sign}}$.
- Receive h_R, h_F and $\{\beta_1, \ldots, \beta_{\mu+\eta}\}$ from $\mathcal{B}^*$, runs as follows:
  1. Extracts the sets F, R from the random oracle, checking the set of queries $\mathcal{Q}_H$ and extracting the pair $(\mathsf{salt}||R, h_R)$ and $(\mathsf{salt}||F, h_F)$. If there is no pair regarding h_R, h_F, then set $R = F = \perp$.
  2. Send $\mathcal{B}^*$ the pair (c_i, z_i) which is made as:
     (a) For all i, $c_i \xleftarrow{\$} \{0,1\}^s$.
     (b) For $i \in F$, $\epsilon_i \xleftarrow{\$} \{0,1\}^\lambda$ and $z_i = \mathsf{RLWE.Enc}(\mathsf{pk}, 0^\lambda||\epsilon_i)$.
     (c) For $R = \{j_1, \ldots, j_\mu\}, z_{j_1}, d_2, \ldots, d_\mu \xleftarrow{\$} \{0,1\}^\lambda$ and $z_{j_i} = \mathsf{RLWE.Enc}(\mathsf{pk}, 0^\lambda||d_i) + (0^\lambda||z_{j_{i-1}})$
- Receive (F', R', r_i) from $\mathcal{B}^*$, runs as follows:
  1. If $F' \neq F$ or $R' \neq R$, then abort.
  2. If any $(\mathsf{FakeFortmat}||r_i, \beta_i) \notin \mathcal{Q}_{H'}$ for $i \in F$, then abort.
  3. For $j \in R$, set $m_j = \gamma$ if $(\gamma, \beta_i) \in \mathcal{Q}_{H'}$. Else set $m_j = \perp$.
  4. Send to $\mathcal{F}_{\mathsf{promise\ sign}}$ the message $(\mathsf{sign\ request}, \mathsf{PK}_{\mathcal{T}}^{eph}, \{\mathsf{FakeFormat}||r_i\}_{i\in F}, \{m_j\}_{j\in R})$.
  5. Obtain response $(\mathsf{promise}, \mathcal{B}^*, \mathsf{ans}, \{\mathsf{FkSign}\}_{i\in[\eta]})$.
  6. If $\mathsf{ans} = \mathsf{no}$, then halt and output whatever $\mathcal{B}^*$ outputs.
  7. Compute $h_{j_l} = c_{j_l} \oplus \mathsf{FkSign}_l$.
  8. Store the pair $(\epsilon_{j_l}, h_{j_l})$ in $\mathcal{Q}_{H^{\mathsf{shk}}}$.
  9. Send ϵ_i for $i \in F$ and the differences $d_2, \ldots, d_\mu$.
- Finally, output whatever $\mathcal{B}^*$ outputs and halt.

Procedure RO1, which is the random oracle simulation for H is as follows:

1. Receive query q for H.
2. If $q \in \mathcal{Q}_H$, retrieve (γ, a) for $\mathcal{Q}_H$.
3. Else $a \xleftarrow{\$} \{0,1\}^{\lambda_2}$.
4. Append (γ, a) to $\mathcal{Q}_H$.
5. Output a.

Procedure RO2, which is the random oracle simulation for H^{shk} is as follows:

1. Receive query q for H^{shk}.
2. If $q \in \mathcal{Q}_{H^{\mathsf{shk}}}$, retrieve (γ, a) for $\mathcal{Q}_{H^{\mathsf{shk}}}$.
3. If $\gamma = \mathsf{RLWE.Dec}(\mathsf{sk}, z_i)$ for some $i \in R$ and $(\gamma, a) \notin \mathcal{Q}_{H^{\mathsf{shk}}}$, then output $\mathsf{RLWE\ failure}$.
4. Else $a \xleftarrow{\$} \{0,1\}^{\lambda_2}$.
5. Append (γ, a) to $\mathcal{Q}_{H^{\mathsf{shk}}}$.
6. Output a.

Figure 5.14: Simulator for the puzzle promise protocol in the case that Bob is corrupt

The simulator S executes $\mathcal{T}^*$ internally.
First, we introduce the algorithm $\mathsf{Sig}(m_i, \mathsf{PK}_{\mathcal{T}}^{eph})$ as follows.

- $L_{\mathsf{fake}}, L_{\mathsf{real}}$ are internal variables.
- If $(m_i, \beta_i, \sigma_i) \in L_{\mathsf{fake}}$, output signature (β_i, σ_i).
- Else if $(m_i, \beta_i, \sigma_i) \in L_{\mathsf{real}}$, then append (m_i, β_i) to $\mathcal{Q}_{H'}$, and output signature (β_i, σ_i).
- Else, abort.

Receive $(\mathsf{Keygen}, \mathcal{B})$. Send request to $\mathcal{T}^*$. Obtain $\mathsf{PK}_{\mathcal{T}}^{eph}$. Send $(\mathsf{PK}_{\mathcal{T}}^{eph}, \mathsf{Sig})$ to $\mathcal{F}_{\mathsf{promise\ sign}}$.
Receive $(\mathsf{sign\ request}, \mathcal{B}, \mathsf{PK}_{\mathcal{T}}^{eph}, \{\mathsf{FkTxn}_i\}_{i\in[\eta]})$.
Pick randomly F, R with $F \cap R = \emptyset$. Compute the RO outputs h_F, h_R and $\beta_1, \ldots, \beta_{\mu+\eta}$. Send them to $\mathcal{T}^*$.
Receive a pair (c_i, z_i) from $\mathcal{T}^*$:

1. Extract ϵ_i by queries to H^{shk}
2. Let σ_i be the signature decrypted from c_i with ϵ_i
3. If $c_i, z_i, \epsilon_i, \beta_i, \sigma_i$ for $i \in F$ are valid, store $(\mathsf{FkTxn}_i, \beta_i, \sigma_i)$ in L_{fake}.
4. If $c_i, z_i, \epsilon_i, \beta_i, \sigma_i$ for $i \in F$ are valid, append i to set Set and append (β_i, σ_i) to L_{real}. If such i does not exist, set $\mathsf{real} = \mathsf{no}$.

For all $i \in F$, $r_i \xleftarrow{\$}$. Send r_i to $\mathcal{T}^*$. Append the pair $(\mathsf{FkTxn}_i || r_i, \beta_i)$ to $\mathcal{Q}_{H'}$.
Receive the openings ϵ_i' to fake messages $i \in F$. Obtain σ_i' with ϵ_i', c_i. If any $(i, c_i, z_i, \epsilon_i', \beta_i, \sigma_i')$ is invalid, send $(\mathsf{promise}, \mathcal{B}, \mathsf{no}, \perp)$ to $\mathcal{F}_{\mathsf{promise\ sign}}$. Else, send $(\mathsf{promise}, \mathcal{B}, \mathsf{yes}, \mathsf{Set})$ to $\mathcal{F}_{\mathsf{promise\ sign}}$.
Now let us confirm the below two cases:
Case 1: If any $i \in F$ such that $(i, \epsilon_i', \beta_i, \sigma_i')$ is valid but $(\cdot, \beta_i, \sigma_i') \notin L_{\mathsf{fake}}$, then abort and output $\mathsf{binding\ fail}$.
Case 2: Suppose that all $i \in F$ such that $(i, \epsilon_i', \beta_i, \sigma_i')$ is valid and $(\cdot, \beta_i, \sigma_i') \in L_{\mathsf{fake}}$.

1. If $\mathsf{real} = \mathsf{no}$, abort and output $\mathsf{cut\ and\ choose\ fail}$.
2. Else, set variables $L_{\mathsf{fake}}, L_{\mathsf{real}}$ for the algorithm Sig.

Figure 5.15: Simulator for the puzzle promise protocol in the case that the tumbler is corrupt (Appendix F in [HAB+16])

close.

$\eth_1$: Instead of computing $c_i = H^{\mathsf{shk}}(\epsilon_i) \oplus \sigma_i$, the simulator chooses $c_i \xleftarrow{\$} \{0,1\}^s$ and stores the pair $(\epsilon_i, c_i \oplus \sigma_i)$ in $\mathcal{Q}_{H^{\mathsf{shk}}}$ in $\eth_1$. The both games $\eth_{0.5}$ and $\eth_1$ are statistically close because H^{shk} is unpredictable.

$\eth_2$: Instead of computing $z_{j_i} = \mathsf{RLWE.Enc}(\mathsf{pk}, \epsilon_{j_i})$, the simulator randomly chooses $z_{j_1}, d_2, \ldots, d_\mu$ from $\{0,1\}^\lambda$ and computes $z_{j_i} = \mathsf{RLWE.Enc}(\mathsf{pk}, d_i) + z_{j_{i-1}}$. Let us check for $\mathsf{RLWE\ failure}$. According to the RLWE assumption, the ciphertext z_{j_i} cannot be distinguished from a uniformly random value. The probability that the plaintext presented will match is $1/2^\lambda$. It is negligible. Note that $\eth_2$ computes neither ϵ_i nor σ_i for real messages m_i with $i \in R$.

$\eth_3$: Instead of the actual signature, the simulator sends the message

$$(\mathsf{sign\ request}, \mathsf{PK}_{\mathcal{T}}^{eph}, \{\mathsf{FakeFormat}||r_i\}_{i\in F}, \{m_j\}_{j\in R})$$

to $\mathcal{F}_{\mathsf{promise\ sign}}$. Then, the simulator S obtains the response $(\mathsf{promise}, \mathcal{B}^*, \mathsf{ans}, \{\mathsf{FkSign}\}_{i\in[\eta]})$ and uses $\sigma_i = \mathsf{FkSign}_i$. $\eth_2$ and $\eth_3$ is identical from the view of $\mathcal{B}^*$. $\eth_3$ is the simulator S in Fig. 5.14 itself.

We conclude that the transcript by the simulator S in Fig. 5.14 is indistinguishable from the one in the real world.

Remark 5.5. *The below lemma shows that $\mathcal{B}^*$ cannot forge a valid signature σ for a valid message $T_{\mathsf{cash}(\mathcal{T},\mathcal{B})}$.*

Lemma 5.6 (Lemma 3 in [HAB+16])**.** *If ECDSA is an existentially unforgeable signature scheme, then* $\Pr[E_{\mathsf{forge}}]$ *is negligible.*

Case that the tumbler is corrupt. The simulator S in Fig. 5.15 plays a role of the corrupt the tumbler $\mathcal{T}^*$. The difference between the real world and the ideal world is whether the simulator S in Fig. 5.15 stops. Let us confirm the probability that the simulator will stop for each event.

- Event $\mathsf{cut\ and\ choose\ fail}$:

$$\Pr[\mathsf{cut\ and\ choose\ fail}] = \frac{1}{\binom{\mu+\eta}{\eta}} + \frac{1}{2^{\lambda_1}}$$

- Event $\mathsf{binding\ fail}$:

$$\Pr[\mathsf{binding\ fail}] = \frac{1}{2^{\lambda_2}}$$

We conclude that the protocol in Fig. 5.12 securely realizes the functionality $\mathcal{F}_{\mathsf{promise\ sign}}$ in Fig. 5.13. □

5.4 Conclusion

Scalability and privacy protection are significant problems with blockchain. In this work, we have proposed an anonymous probabilistic payment to solve these simultaneously. Our proposal is not restricted to any particular cryptocurrency. We have mediated the tumbler of the payment channel hub between a payer and a payee. A cryptographic puzzle plays a role in controlling the intermediation and the execution of transactions. Masking the puzzle allows the payer and the payee to unlink their payments. Besides, we have introduced a novel fractional oblivious transfer based on the RLWE encryption. We have adopted it for the probabilistic payment. The proposed protocol realizes the ideal functionalities discussed in TumbleBit (NDSS 2017).

Chapter 6

Conclusion

Blockchain is an underlying technology that constitutes our digital society. Privacy protection is a significant theme in society. Homomorphic encryption is a promising technology for privacy protection. Confidentiality and unlinkability are challenging issues regarding privacy-preserving blockchain. High throughput is also a challenging issue regarding blockchain. A permissionless blockchain has a much lower throughput than permissioned blockchain. We hope that blockchain will be widely available in the real world. Privacy-preserving and high throughput blockchain are desirable. In this thesis, we have studied privacy-preserving blockchain with homomorphic encryption. First, we achieve a privacy-preserving and verifiable permissioned blockchain. Next, we realize a confidential and auditable permissionless blockchain. Finally, we propose an unlinkable and high throughput permissionless blockchain.

Traceability in permissioned blockchain. We have achieved privacy protection and high transparency in a permissioned blockchain. Meaningful traceability consists of three properties. These are trade privacy, preservation, and noninvolvement. This work is a proposal wherein both preservation and noninvolvement hold while protecting trade privacy. We have constructed a traceability model based on the three properties and encrypted it with the RLWE encryption. Moreover, we have encrypted the model with somewhat homomorphic encryption by using the ring isomorphism encoding. We have confirmed that the encrypted model is feasible. We have shown the protocol to verify that the plaintext is equal to zero using non-interactive zero-knowledge proof.

Confidential and auditable payments. We have constructed a confidential and auditable payment scheme. The proposed scheme allows a court or an authority to audit transactions while keeping the transaction information confidential. Every participant writes the ciphertexts of transaction information in a ledger. We have confirmed the concealment of the transaction information and the soundness of the scheme. The proposed scheme is secure in this sense. The court or the authority can forcibly reveal transaction information with a unique secret key. In this sense, the proposed scheme is auditable.

Anonymous probabilistic payment in payment hub. Scalability and privacy protection are significant problems. To solve these simultaneously, we have proposed an anonymous probabilistic payment. Our proposal is not restricted to any particular cryptocurrency. We have mediated the tumbler of the payment channel hub between a payer and a payee. A cryptographic puzzle plays a role in controlling the intermediation and execution of transactions. Masking the puzzle allows the payer and the payee to unlink their payments. Besides, we have introduced a novel fractional oblivious transfer based on the RLWE encryption. We have adopted it for the probabilistic payment. The proposed protocol realizes the ideal functionalities discussed in TumbleBit (NDSS 2017).

Bibliography

[AAUC18] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Comput. Surv.*, 51(4), July 2018.

[ABB+18] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, page 30. ACM, 2018.

[ABC20] Ghada Almashaqbeh, Allison Bishop, and Justin Cappos. Microcash: Practical concurrent processing of micropayments. In Joseph Bonneau and Nadia Heninger, editors, *Financial Cryptography and Data Security*, pages 227–244, Cham, 2020. Springer International Publishing.

[ACDCKK18] Elli Androulaki, Christian Cachin, Angelo De Caro, and Eleftherios Kokoris-Kogias. Channels: Horizontal scaling and confidentiality on permissioned blockchains. In Javier Lopez, Jianying Zhou, and Miguel Soriano, editors, *Computer Security*, pages 111–131, Cham, 2018. Springer International Publishing.

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange—a new hope. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 327–343, Austin, TX, August 2016. USENIX Association.

[BAZB20] Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards privacy in a smart contract world. In Joseph Bonneau

and Nadia Heninger, editors, *Financial Cryptography and Data Security*, pages 423–443, Cham, 2020. Springer International Publishing.

[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)*, pages 315–334. IEEE, 2018.

[BCK+14] Fabrice Benhamouda, Jan Camenisch, Stephan Krenn, Vadim Lyubashevsky, and Gregory Neven. Better zero-knowledge proofs for lattice encryption and their application to group signatures. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, pages 551–572, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[BCNS15] J. W. Bos, C. Costello, M. Naehrig, and D. Stebila. Post-quantum key exchange for the tls protocol from the ring learning with errors problem. In *2015 IEEE Symposium on Security and Privacy*, pages 553–570, May 2015.

[BD18] Zvika Brakerski and Nico Döttling. Two-message statistically sender-private ot from lwe. In Amos Beimel and Stefan Dziembowski, editors, *Theory of Cryptography*, pages 370–390, Cham, 2018. Springer International Publishing.

[BDLN16] Carsten Baum, Ivan Damgård, Kasper Green Larsen, and Michael Nielsen. How to prove knowledge of small secrets. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 478–498, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[BG13] Techane Bosona and Girma Gebresenbet. Food traceability as an integral part of logistics management in food and agricultural supply chain. *Food control*, 33(1):32–48, 2013.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In Joe Kilian, editor, *Theory of Cryptography*, pages 325–341, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):13, 2014.

[BKS19] Elette Boyle, Lisa Kohl, and Peter Scholl. Homomorphic secret sharing from lattices without fhe. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019*, pages 3–33, Cham, 2019. Springer International Publishing.

[BM90] Mihir Bellare and Silvio Micali. Non-interactive oblivious transfer and applications. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 547–557, New York, NY, 1990. Springer New York.

[BNM+14] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A. Kroll, and Edward W. Felten. Mixcoin: Anonymity for bitcoin with accountable mixes. In Nicolas Christin and Reihaneh Safavi-Naini, editors, *Financial Cryptography and Data Security*, pages 486–504, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[BR99] Mihir Bellare and Ronald L Rivest. Translucent cryptography—an alternative to key escrow, and its implementation via fractional oblivious transfer. *Journal of cryptology*, 12(2):117–139, 1999.

[BSBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. *IACR Cryptology ePrint Archive*, 2018:46, 2018.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, pages 505–524, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[Can01] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145. IEEE, Oct 2001.

[CDG+18] D. B. Cousins, G. Di Crescenzo, K. D. Gür, K. King, Y. Polyakov, K. Rohloff, G. W. Ryan, and E. Savas. Implementing conjunction obfuscation under entropic ring lwe. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 354–371, 2018.

[CGL+16] Alessandro Chiesa, Matthew Green, Jingcheng Liu, Peihan Miao, Ian Miers, and Pratyush Mishra. Decentralized anonymous micropayments. Cryptology ePrint Archive, Report 2016/1033, 2016. `https://eprint.iacr.org/2016/1033`.

[CGL+17] Alessandro Chiesa, Matthew Green, Jingcheng Liu, Peihan Miao, Ian Miers, and Pratyush Mishra. Decentralized anonymous micropayments. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, pages 609–642, Cham, 2017. Springer International Publishing.

[CLP17] Hao Chen, Kim Laine, and Rachel Player. Simple encrypted arithmetic library - seal v2.1. In Michael Brenner, Kurt Rohloff, Joseph Bonneau, Andrew Miller, Peter Y.A. Ryan, Vanessa Teague, Andrea Bracciali, Massimiliano Sala, Federico Pintore, and Markus Jakobsson, editors, *Financial Cryptography and Data Security*, pages 3–18, Cham, 2017. Springer International Publishing.

[CLPX18] Hao Chen, Kim Laine, Rachel Player, and Yuhou Xia. High-precision arithmetic in homomorphic encryption. In Nigel P. Smart, editor, *Topics in Cryptology – CT-RSA 2018*, pages 116–136, Cham, 2018. Springer International Publishing.

[DFKP13] George Danezis, Cedric Fournet, Markulf Kohlweiss, and Bryan Parno. Pinocchio coin: building zerocoin from a succinct pairing-based proof system. In *Proceedings of the First ACM workshop on Language support for privacy-enhancing technologies*, pages 27–30. ACM, 2013.

[DLP14] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over ntru lattices. In Palash Sarkar and Tetsu Iwata,

editors, *Advances in Cryptology – ASIACRYPT 2014*, pages 22–41, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[DPW+16] Johnny Dilley, Andrew Poelstra, Jonathan Wilkins, Marta Piekarska, Ben Gorlick, and Mark Friedenbach. Strong federations: An interoperable blockchain solution to centralized third-party risks. *arXiv preprint arXiv:1612.05491*, 2016.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO' 86*, pages 186–194, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012:144, 2012.

[GC15] Matthias Geihs and Daniel Cabarcas. Efficient integer encoding for homomorphic encryption via ring isomorphisms. In Diego F. Aranha and Alfred Menezes, editors, *Progress in Cryptology - LATINCRYPT 2014*, pages 48–63, Cham, 2015. Springer International Publishing.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC ' 09, pages 169–178, New York, NY, USA, 2009. Association for Computing Machinery.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, pages 626–645, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[GKKT16] Johannes Göbel, Holger Paul Keeler, Anthony E Krzesinski, and Peter G Taylor. Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay. *Performance Evaluation*, 104:23–41, 2016.

[GM17] Matthew Green and Ian Miers. Bolt: Anonymous payment channels for decentralized currencies. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS ' 17, page 473–489, New York, NY, USA, 2017. Association for Computing Machinery.

[GM18] Nicholas Genise and Daniele Micciancio. Faster gaussian sampling for trapdoor lattices with arbitrary modulus. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 174–203, Cham, 2018. Springer International Publishing.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

[HAB+16] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. Tumblebit: An untrusted bitcoin-compatible anonymous payment hub. 2016. `https://eprint.iacr.org/2016/575`.

[HAB+17] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. Tumblebit: An untrusted bitcoin-compatible anonymous payment hub. In *Network and Distributed System Security Symposium*, 2017.

[HBG16] Ethan Heilman, Foteini Baldimtsi, and Sharon Goldberg. Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions. In Jeremy Clark, Sarah Meiklejohn, Peter Y.A. Ryan, Dan Wallach, Michael Brenner, and Kurt Rohloff, editors, *Financial Cryptography and Data Security*, pages 43–60, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[Hir18] Toufic Hirbli. Palm oil traceability: Blockchain meets supply chain. Master's thesis, Department of Supply Chain Management, Massachusetts Institute of Technology, 2018.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Ntru: A ring-based public key cryptosystem. In Joe P. Buhler, editor, *Algorithmic Number*

Theory, pages 267–288, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[HS01] Jeffrey Hoffstein and Joseph Silverman. Optimizations for ntru. *Public-Key Cryptography and Computational Number Theory, Warsaw*, pages 77–88, 2001.

[HS14] Shai Halevi and Victor Shoup. Helib. *Retrieved from HELib: https://github. com. shaih/HElib*, 2014.

[Kam18] Reshma Kamath. Food traceability on blockchain: Walmart's pork and mango pilots with ibm. *The Journal of the British Blockchain Association*, 1(1):3712, 2018.

[LATV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, STOC ' 12, page 1219–1234, New York, NY, USA, 2012. Association for Computing Machinery.

[LH19] Momeng Liu and Yupu Hu. Universally composable oblivious transfer from ideal lattice. *Front. Comput. Sci.*, 13(4):879–906, August 2019.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *Journal of the ACM (JACM)*, 60(6):43, 2013.

[Lyu09] Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, pages 598–616, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, pages 738–755, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[Max13a] Gregory Maxwell. Coinjoin: Bitcoin privacy for the real world. *URL: https://bitcointalk.org/index.php*, 2013.

[Max13b] Gregory Maxwell. Coinswap: Transaction graph disjoint trustless trading. *CoinSwap: Transactiongraphdisjointtrustlesstrading (October 2013)*, 2013.

[Max15] Greg Maxwell. Confidential transactions. *URL: https://people. xiph. org/˜ greg/confidential_values. txt (Accessed 09/05/2016)*, 2015.

[MGGR13] Ian Miers, Christina Garman, Matthew Green, and Aviel D Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *2013 IEEE Symposium on Security and Privacy*, pages 397–411. IEEE, 2013.

[MO19] Tatsuo Mitani and Akira Otsuka. Traceability in permissioned blockchain. In *2019 2nd IEEE International Conference on Blockchain*, pages 286–293. IEEE, 2019.

[MO20a] Tatsuo Mitani and Akira Otsuka. Anonymous probabilistic payment in payment hub. Cryptology ePrint Archive, Report 2020/748, 2020. `https://eprint.iacr.org/2020/748`.

[MO20b] Tatsuo Mitani and Akira Otsuka. Confidential and auditable payments. 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 466–480, Cham, 2020. Springer International Publishing.

[MO20c] Tatsuo Mitani and Akira Otsuka. Traceability in permissioned blockchain. *IEEE Access*, 8:21573–21588, 2020.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, pages 700–718, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM Journal on Computing*, 37(1):267–302, 2007.

[MSRL+19] Pedro Moreno-Sanchez, Randomrun, Duc V. Le, Sarang Noether, Brandon Goodell, and Aniket Kate. Dlsag: Non-interactive refund transactions for interoperable payment channels in monero. Technical report, 2019. `https://eprint.iacr.org/2019/595`.

[MU17] Michael Mitzenmacher and Eli Upfal. *Probability and computing: randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press, 2017.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Retrieved from Website: https://bitcoin.org/bitcoin.pdf*, 2008.

[NKMS16] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 305–320. IEEE, 2016.

[NLV11] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, CCSW '11, pages 113–124, New York, NY, USA, 2011. ACM.

[Noe15] Shen Noether. Ring signature confidential transactions for monero. 2015. `https://eprint.iacr.org/2015/1098`.

[PBF+19] Andrew Poelstra, Adam Back, Mark Friedenbach, Gregory Maxwell, and Pieter Wuille. Confidential assets. In Aviv Zohar, Ittay Eyal, Vanessa Teague, Jeremy Clark, Andrea Bracciali, Federico Pintore, and Massimiliano Sala, editors, *Financial Cryptography and Data Security*, pages 43–63, Berlin, Heidelberg, 2019. Springer Berlin Heidelberg.

[PD16] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments, 2016.

[Ped92] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryp-*

tology — CRYPTO '91, pages 129–140, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[Pei14] Chris Peikert. Lattice cryptography for the internet. In Michele Mosca, editor, *Post-Quantum Cryptography*, pages 197–219, Cham, 2014. Springer International Publishing.

[PR17] Omer Paneth and Guy N. Rothblum. On zero-testable homomorphic encryption and publicly verifiable non-interactive arguments. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography*, pages 283–315, Cham, 2017. Springer International Publishing.

[Ps15] Rafael Pass and abhi shelat. Micropayments for decentralized currencies. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS ' 15, page 207–218, New York, NY, USA, 2015. Association for Computing Machinery.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), September 2009.

[Riv97] Ronald L. Rivest. Electronic lottery tickets as micropayments. In Rafael Hirschfeld, editor, *Financial Cryptography*, pages 307–314, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[RMSK14] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. Coinshuffle: Practical decentralized coin mixing for bitcoin. In Mirosław Kutyłowski and Jaideep Vaidya, editors, *Computer Security - ESORICS 2014*, pages 345–364, Cham, 2014. Springer International Publishing.

[RR18] Kurt Rohloff and Gerard Ryan. The palisade lattice cryptography library. *Retrieved from Website: https://git.njit.edu/palisade/PALISADE*, 2018.

[SALY17] Shi-Feng Sun, Man Ho Au, Joseph K. Liu, and Tsz Hon Yuen. Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In Simon N. Foley, Dieter Gollmann, and Einar Snekkenes, editors, *Computer Security – ESORICS 2017*, pages 456–474, Cham, 2017. Springer International Publishing.

[SCG+14] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE, 2014.

[TMSB18] Vinesh Thiruchelvam, Alexandre Shaka Mughisha, Maryam Shahpasand, and Mervat Bamiah. Blockchain-based technology in the coffee supply chain trade: Case of burundi coffee. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 10(3-2):121–125, 2018.

[WG18] Karl Wüst and Arthur Gervais. Do you need a blockchain? In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 45–54. IEEE, 2018.

[Whe97] David Wheeler. Transactions using bets. In Mark Lomas, editor, *Security Protocols*, pages 89–92, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[Woo14] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014.

[Wüs16] Karl Wüst. Security of blockchain technologies. Master's thesis, Department of Computer Science, ETH Zürich, 2016.

[YSL+20] Tsz Hon Yuen, Shi-Feng Sun, Joseph K. Liu, Man Ho Au, Muhammed F. Esgin, Qingzhao Zhang, and Dawu Gu. Ringct 3.0 for blockchain confidential transaction: Shorter size and stronger security. In Joseph Bonneau and Nadia Heninger, editors, *Financial Cryptography and Data Security*, pages 464–483, Cham, 2020. Springer International Publishing.

[ZXL19] Rui Zhang, Rui Xue, and Ling Liu. Security and privacy on blockchain. *ACM Computing Surveys (CSUR)*, 52(3), July 2019.

[ZZD+15] Jiang Zhang, Zhenfeng Zhang, Jintai Ding, Michael Snook, and Özgür Dagdelen. Authenticated key exchange from ideal lattices. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, pages 719–751, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.