博士論文

分散台帳向けセキュリティ・プライバシ技術の研究
(Security and Privacy Enhancing Technologies for Blockchain Systems)

長沼　健

Acknowledgement

I would like to show my highest appreciation to my supervisor, Professor Hiroshi Kori, and Professor Noboru Kunihiro, for their carefully considered feedback, valuable and constructive suggestions, and enthusiastic encouragement. I would also like to express my gratitude to Kunihiro-lab members, whose comments made an enormous contribution to my work. I would also like to thank my colleagues at Hitachi, Ltd., for their continuous support. Finally, I wish to thank my family for their moral support and warm encouragement.

Contents

Chapter 1

Introduction

The application of IT technology to the financial field called Fintech is progressing, and many companies that have not traditionally been in the financial industry have launched services such as electronic cash issuance and QR code settlement. Also, in such a trend, users' needs for technology and platform for performing settlement and contract conclusion securely and inexpensively are increasing. Blockchain or Distributed Ledger Technology, a generalized concept, has recently attracted a great deal of attention as a technology for realizing these services. Blockchain is a core technology used in the Bitcoin system proposed by Satoshi Nakamoto [69]. It enables digital cash, which is called bitcoin, transferring in the P2P network without Trusted Third Parties (TTPs) such as banks. As a result, by reducing the transaction fees paid to the intermediary, the payment system is operated at low fees.

The distributed ledger technology typified by Bitcoin roughly includes a transaction generation unit that generates a transaction, including remittance or contract information, and a consensus unit that confirms the validity of the transaction and records it in the distributed ledger [85]. When a user generates a transaction, a secret key, which is secret information known only to the user, is used to attach a digital signature based on the public key cryptosystem. Furthermore, the consensus unit verifies the attached digital signature, confirms the consistency with other transactions, and then stores with a data structure that is difficult to falsification [69, 61, 81].

Thus, in the blockchain, the user's secret key and digital cash or other assets are closely linked, and management of the secret key is a crucial security issue. For example, if the user loses the secret key, then he/she cannot generate a transaction at all, and the asset is substantially lost [63]. In fact, in the case of Bitcoin, it has been reported [75] that the secret key for bitcoin, which is $17-23\%$ of the all issued amount, has already been lost, making it immovable assets. Also, if a malicious attacker steals a secret key, improper remittance by impersonation occurs [1, 76]. As another problem,

in the blockchain, since the ledger data is shared by multiple nodes on the P2P network, there is a privacy problem in which sensitive information such as "who sent money to whom" is leaked. In this thesis, We propose the technologies to solve the security and privacy problems of the above and aim at the realization of a blockchain infrastructure that allows users to enjoy services with peace of mind.

Accurately, we describe a solution using the zero-knowledge proof for the privacy problem in which ledger data is shared by multiple nodes, and propose new zero-knowledge proof schemes. More precisely, the user's privacy can be preserved by encrypting sensitive data in a transaction. However, since it is encrypted, another problem occurs that the consensus unit cannot verify the validity of the transaction. Cryptographically, this problem can be solved with zero-knowledge proof technologies. Zero-knowledge proof is a general term for the technology that allows the validator can check that concealed data satisfies certain propositions. By using this technique, the user can prove the validity of the transaction to the consensus unit while keeping the sensitive information secret. In Chapter 2, we propose quantum computer resistant zk-SNARK type zero-knowledge proof schemes that support arithmetic circuits for the use case in consortium/permissioned blockchains such as Hyperledger Fabric [3] or Quorum [7]. In 2018, Gennaro et al. [52] proposed a designed-verifier type zk-SNARK based on SSPs (Square Span Programs) with quantum computer attack resistance. However, this scheme only supports propositions expressed by Boolean circuits. Therefore, the construction of zk-SNARK, which has quantum computer resistance and supports arithmetic circuits, has been an open problem. Also, many existing zk-SNARK type zero-knowledge proofs are not suitable for use in public/decentralized blockchains such as Bitcoin because they require system parameter setup by a TTP or a central institution. In Chapter 3, we describe a general method of constructing a zero-knowledge proof that does not require setup by TTPs from vector commitment schemes. Our method can be applied to the typical vector commitment schemes: Pedersen commitment [72] based on the discrete logarithm problem and the Ajtai commitment [13] based on the lattice problem. As a result, trust-less and efficient zero-knowledge proofs based on the discrete logarithm problem and the lattice problem can be constructed. These were also open problems in this area. Also, in Chapter 4, we propose a solution to the problem of asset loss due to the loss/leakage of the secret key described above using a digital signature scheme based on biometric information. In this digital signature scheme, the secret key of the user is generated each time from the biometric information at the time of the signature generation, so there is no need to store the secret key. As a result, the risk of loss/leakage is greatly reduced. Finally, in Chapter 5, we describe a protocol that uses the channel function of Hyperledger Fabric [3] to execute netting settlement while keeping the ledger information secret.

1.1 Our Contributions

We summarize our contributions as follows:

- We propose novel zero-knowledge proof schemes to solve the privacy problem of ledger data, especially zk-SNARKs for arithmetic circuits with quantum computer resistance.
- We propose a general method that can configure zero-knowledge proofs without a TTP from vector commitment schemes.
- We propose a solution to the problem of secret key loss for blockchain users using biometrics authentication technology.
- We propose a secure multi-party protocol for executing netting settlement on the permissioned/consortium blockchains.

1.2 Organization of Thesis

This thesis is organized as follows. In Chapter 2, we propose novel zk-SNARKs for arithmetic circuits with quantum computer resistance. Furthermore, in Chapter 3, we propose a general method that can configure zero-knowledge proofs without a TTP from vector commitment schemes. In Chapter 4, we propose a solution to the problem of secret key loss for blockchain users using biometrics authentication technology. Finally, in Chapter 5, we propose a secure multi-party protocol for executing netting settlement on Hyperledger Fabric [3].

Chapter 2

Post-Quantum zk-SNARKs from QAPs

2.1 Summary

In recent years, zero-knowledge proof and *zero-knowledge succinct non-interactive argument of knowledge (zk-SNARK)* are drawing significant attention as privacy-enhancing technologies in various domains, especially cryptocurrency and blockchain industries and verifiable computations. A zero-knowledge proof is a (non) interactive proof protocol that proves that the prover has correct knowledge without revealing the knowledge to the verifier. In several cryptocurrency systems (e.g., Zcash [11], Ethereum [2], Quorum [7], Pinocchio coin [40]), zero-knowledge proofs are used for concealing the transaction amounts and anonymizing both senders and receivers.

In these systems, the zk-SNARK protocol from Pinocchio [71], which was originally proposed for verifiable computations and the libsnark library [4, 23], was used. These systems assume the difficulty of the discrete logarithm problem of elliptic curves as the base of security. Therefore, these systems are not resistant to quantum attacks. Some zero-knowledge proof schemes, such as lattice-based schemes by Genaro et al. [52], Baum et al. [16], Libert et al. [64], Peikert and Shiehian [73], Nitulescu [70], provide resistance against quantum attacks. In addition, for reducing the proof data, we must use a succinct and non-interactive proof system because there are multiple proof verifiers when considering the use for permissionless/permissioned blockchain systems. Therefore, a zero-knowledge proof scheme of the zk-SNARK [25] type is desirable.

Recently, Gennaro et al. proposed a post-quantum designated verifier type zk-SNARK that is based on the learning with errors (LWE) encryption and *square span programs (SSPs)*. However, their scheme is a zk-SNARK that targeted Boolean circuits. Therefore, in Gennaro et al. [52], the construction of a designated verifier type zk-SNARK method for *quadratic arith-*

Table 2.1: Efficiency comparison: The circuit size = 2^{16} and CRS while using PRNG for its generation. Our experimental environment comprises an Intel Core i7-9700K CPU @ 3.60 GHz with a single thread. The term $|CRS|$ column represents the size of CRS: Common Reference String. The memory column represents the memory requirements for Setup and Prover algorithms.

	Circuit	PQ	Proof size	|CRS|	Setup	Prover	Verifier	Memory	Assumption
Proposal 1	QAP	✓	1083 KB	30.5 MB	98.4 s	90.0 s	2.2 ms	33.4 GB	PKE, PDH
Proposal 2	QAP	✓	405 KB	15.4 MB	49.7 s	50.7 s	0.8 ms	16.8 GB	Linear-only
Nitulescu [70]	SAP	✓	270 KB	23.8 MB	76.9 s	100.9 s	0.6 ms	26.0 GB	Linear-only
GMNO [52]	SSP	✓	640 KB	8.63 MB	98.8 s	142.9 s	1.5 ms	9.4 GB	PKE, PDH
PHGR [71]	QAP	-	288 B	6.50 MB	2.8 s	0.6 s	4.1 ms	9.0 MB	PKE, PDH

metic program (QAP)-characterized arithmetic circuits is described as an open problem. Recently, Nitulescu [70] proposed a post-quantum designated verifier zk-SNARK for arithmetic circuits by using *square arithmetic programs (SAPs)*. Notably, SAP is another expression method for arithmetic circuits and can be considered as a special case of QAP (see [56, 57]).

In this paper, we give other answers to this problem to propose two post-quantum designated verifier zk-SNARK schemes for QAP-characterized arithmetic circuits. Furthermore, we implement our proposed schemes and all the known LWE-based zk-SNARK schemes and evaluate their performances.

Related works: Lattice-based (zk-) SNARKs.

In recent years, several designated verifier type SNARK schemes have been proposed on the basis of assumptions of the lattice-based cryptography (Boneh et al. [28, 29], Gennaro et al. [52], Nitulescu [70]). The schemes by Boneh et al. [28, 29] were based on linear probabilistically checkable proofs and used the assumption of the linear-only encryption for security proof; i.e., an attacker can generate only the ciphertexts of the affine combinations of given LWE ciphertexts. The authors described a general transformation procedure to zero-knowledge proofs; however, it is not clear whether it is applicable in the lattice-based setting. However, the scheme by Gennaro et al. [52] is a zk-SNARK that is based on SSP and LWE cryptosystems and can be proved to be secure by employing weaker and more standard assumptions that do not use the linear-only encryption assumption. In addition, their scheme is characterized by its zero-knowledge property. However, their scheme targeted Boolean circuits only, and it is difficult to apply immediately to the existing blockchain systems because the libsnark library [4], which is the standard zero-knowledge proof library in the blockchain area, targeted QAP-characterized arithmetic circuits. To use the scheme by Gennaro et al. [52] in a blockchain system, one must convert the existing

arithmetic circuit to a Boolean circuit, then transform it to a rank-1 constraint system, and further generate an SSP for the given R1CS. Recently, Nitulescu [70] proposed an LWE-based zk-SNARK for SAP-characterized arithmetic circuits. The zero-knowledge proof of her scheme comprised only two LWE ciphertexts, and it is somewhat similar to our second proposed scheme. Conversely, our second scheme is a generalization of her scheme because QAP is a generalization of SAP. Furthermore, these schemes require the linear-only encryption, which is a stronger assumption than those of our first scheme.

Our contributions.

In this paper, we propose two designated verifier type zk-SNARKs that are based on the LWE encryption and QAPs. Therefore, these schemes seem to be secure against quantum attacks. Furthermore, we implemented our proposed schemes, the scheme by Gennaro et al. [52], and the scheme by Nitulescu [70] and reported the experimental results of these schemes. We summarize our contributions as follows.

Proposal 1. Pinocchio-like LWE-based zk-SNARK using QAPs.

We construct the designated verifier type zk-SNARK for arithmetic circuits characterized by QAPs, by assuming the q-PKE and q-PDH assumptions for the LWE cryptosystem. Notably, the first proposed scheme does not require the linear-only encryption assumption [28, 29, 70], which is a stronger assumption than the q-PKE and q-PDH assumptions [1]. This scheme uses the data format by Pinocchio [71, 23] to the maximum possible extent; therefore, it is compatible with the existing permissioned blockchain systems and is easy to implement in the libsnark [4, 23] or other systems using the Pinocchio protocol. Another advantage of this scheme is that it is appropriate for the multi-thread CPU architecture. In addition, our experimental results show that this scheme offers almost the same performance as that of the second scheme in a multi-thread CPU environment (see Section 2.9).

Proposal 2. More efficient LWE-based zk-SNARK using QAPs.

In the above-mentioned first proposed scheme, the size of the zero-knowledge proof is equal to that of eight LWE ciphertexts. The size of the scheme by Gennaro et al. [52] is as equal to that of five LWE ciphertexts. We refer to the zk-SNARK configuration technique by Groth [56] and assume the linear-only encryption used in Boneh et al. [28, 29]. We construct a zk-SNARK using a QAP that has three LWE ciphertexts as the zero-knowledge proof. In addition, our experimental results show that the Setup and Prover (proof generation) algorithms of this scheme are the fastest among the other post-quantum zk-SNARKs (see Table 2.1). The setup and the proof generation algorithms are the bottlenecks of the zk-SNARK system; especially, the

[1] We can prove the security of this scheme by using the q-PKE and q-PDH assumptions, which are weaker and concrete assumptions than the linear-only assumption. However, the data size of the zero-knowledge proof is greater than those of the other schemes because we cannot take the advantage of the linear-only assumption.

proof-generation algorithm is more important in practical use. Therefore, this scheme seems to be the most efficient one. Our experimental results show that the processing time of this scheme is approximately three times faster than that of the scheme by Gennaro et al. [2] or two times faster than the one proposed by Nitulescu (see Table 2.1). The scheme by Nitulescu, which is based on the SAP expression requires twice as many multiplication circuits as those required by QAP expression. Consequently, this scheme is slower than our QAP-based one.

Secure and appropriate operations for permissioned BC.

We propose a system architecture for operating the proposed schemes on permissioned blockchain such as Quorum [7], Hyperledger [3]. Furthermore, we propose a simple multi-party protocol that securely generates and shares the public parameters CRS. This protocol is secure on the assumption that at least one node is honest and can make the centralized setup decentralized and prevent attacks like those described in M. Campanelli et al. [34]. Moreover, since our multi-party protocols are based on LWE encryption, quantum attack resistance is achieved.

Implementation and optimization.

We implemented our proposed schemes in the libsnark library [4, 23] and optimized them for performing QAP operations; e.g., we optimized the processing of the sparse-matrix expressions in QAP, Montgomery reduction, and loop operation for the LWE encryption). In addition, we implemented the schemes by Gennaro et al. [52], Nitulescu [70], and Pinocchio [71, 23], respectively (see Table 2.1). The experimental results show that the processing performances of the first proposed scheme is almost the same as those of the existing schemes by Gennaro et al. or Nitulescu, and that the second proposed scheme is two or three times faster than their schemes. Notably, the scheme by Gennaro et al. targets Boolean circuits, thus we cannot directly compare their efficiencies.

Because our proposed schemes are of one or two orders of magnitude slower than the original scheme by Pinocchio [71, 23], which is a pre-quantum scheme, both our schemes must be further optimized for use with real systems (see Section 2.9 for the multi-thread optimization).

Technical challenges.

Our first proposed scheme employs the noise-smudging technique by Boneh et al. [27]. Therefore, our constructions are conceptually similar to those of Gennaro et al. [52]. However, their construction, as well as the security proof, is specified for Boolean circuits, and, therefore, it is difficult to be applied to the case of arithmetic circuits. Therefore, we give another type of security proof with the same security assumptions, i.e., the q-PKE and

[2]The scheme by Gennaro et al. [52] targets Boolean circuits, and our schemes target arithmetic circuits. Therefore, we cannot directly compare their efficiencies in terms of circuit sizes. However, compared with Boolean circuits, QAPs are often more efficient in expressing computations from real code.

q-PDH assumptions, to overcome the obstacles due to arithmetic circuits.

The second proposed scheme is based on the linear-only assumption as in the scheme by Nitulescu [70]; therefore, the construction and the security proof of the second proposed scheme are similar to those of the scheme by Nitulescu [70]. Because QAP is a generalization of SAP, our construction and security proof can be regarded as a generalization of the scheme by Nitulescu [70]. If we take appropriate parameters in our arguments, we can construct the zk-SNARK scheme by Nitulescu.

Limitation of application.

Our proposed schemes and all known post-quantum zk-SNARKs are however, *designated verifier* type zero-knowledge proof protocols and require some secret information in the verification algorithm. Therefore, these schemes cannot be immediately applied to public-type blockchain systems. Currently, we can apply these schemes to permissioned-type blockchains or verifiable computation, which enable a computer to offload the computation of some functions, to other untrusted web-servers while maintaining verifiable results. Therefore, the permissioned blockchain and verifiable computation might be appropriate applications of our designated verifier schemes. The construction of LWE based post-quantum zk-SNARKs for a public verifier type remains an open problem.

2.2 Preliminaries

Notation. In this paper, we denote the set of real numbers by $\mathbf{R}$, set of integers by $\mathbf{Z}$, set of natural numbers by $\mathbf{N}$, and set of integers modulo q by $\mathbf{Z}_q$. We also denote $\{0, 1, ..., N\} \subset \mathbf{Z}$ by $[0, N]$. Let $\lambda \in \mathbf{N}$ be the computational security parameter and $\kappa \in \mathbf{N}$ the statistical parameter. A function $f(x)$ on $\mathbf{R}$ is negligible in λ if $f(\lambda) = o(\lambda^{-c})$ for every fixed constant $c \in \mathbf{R}_{>0}$, and we denote it by $\mathrm{negl}(\lambda)$. In addition, the probability is overwhelming in λ if it is equal to $1 - \mathrm{negl}(\lambda)$.

In the security proofs of this study, we assume that all the adversaries are probabilistic Turing machines that run in time $\mathrm{poly}(\lambda)$ (PPT), and we denote an adversary by $\mathcal{A}$. We also denote when $\mathcal{A}$ interacts with an oracle $\mathcal{O}$ by $\mathcal{A}^{\mathcal{O}}$. For two PPT machines, $\mathcal{A}$ and $\chi_{\mathcal{A}}$, by $(\mathcal{A}||\chi_{\mathcal{A}})(x)$ we denote the execution of $\mathcal{A}$ followed by that of $\chi_{\mathcal{A}}$ on the same input x.

2.2.1 Quadratic Arithmetic Programs

Here, we define QAPs according to Gennaro et al. [51] and Parno et al. [71]. Let $\mathbf{Z}_p$ be the finite field of order p.

Definition 1. (QAPs)
A QAP $\mathcal{Q} = (\mathcal{A}, \mathcal{B}, \mathcal{C}, t(x))$ over $\mathbf{Z}_p$ is a tuple that comprises $3(m+1)$ polynomials $\mathcal{A} = \{a_i(x)\}_{i=0}^{m}$, $\mathcal{B} = \{b_i(x)\}_{i=0}^{m}$, $\mathcal{C} = \{c_i(x)\}_{i=0}^{m}$, and a target poly-

nomial $t(x)$ such that $\deg(t(x)) \geq \max(\deg(a_i(x)),\ \deg(b_i(x)), \deg(c_i(x)))$ for all $i \in [0, m]$. Let C be an arithmetic circuit over $\mathbf{Z}_p$ with n inputs and n' outputs. Notably, $\mathcal{Q}$ verifies C if and only if for each inputs/outputs pair $(d_1, ..., d_{n+n'}) \in \mathbf{Z}_p^{n+n'}$ of C, there exist $(d_{n+n'+1}, ..., d_m) \in \mathbf{Z}_p^{m-n-n'}$ such that $p(x)$ satisfies the following:

$$p(x) := \Big(a_0(x) + \sum_{i=1}^{m} d_i a_i(x)\Big) \cdot \Big(b_0(x) + \sum_{i=1}^{m} d_i b_i(x)\Big) - \Big(c_0(x) + \sum_{i=1}^{m} d_i c_i(x)\Big),$$

then $p(x)$ is divisible by the target polynomial $t(x)$. Conversely, $(d_1, ..., d_{n+n'}) \in \mathbf{Z}_p^{n+n'}$ is an inputs/outputs pair of C if and only if there exist $(d_{n+n'+1}, ..., d_m) \in \mathbf{Z}_p^{m-n-n'}$ and $h(x)$ such that $p(x) = t(x)h(x)$. We refer to the number m as the size of $\mathcal{Q} = (\mathcal{A}, \mathcal{B}, \mathcal{C}, t(x))$ and $\deg t(x)$ as its degree.

Theorem 1. (Gennaro et al. [51], Ben-Sasson et al. [23])
Let C be a fan-in-2 type arithmetic circuit over $\mathbf{Z}_p$ with d multiplication gates. Accordingly, there exists a probabilistic and polynomial-time algorithm to generate a degree d QAP $\mathcal{Q} = (\mathcal{A}, \mathcal{B}, \mathcal{C}, t(x))$ such that $\mathcal{Q}$ verifies C.

We denote this algorithm by $\mathbf{QAP}(C) \rightarrow \mathcal{Q} = (\mathcal{A}, \mathcal{B}, \mathcal{C}, t(x))$, where C denotes an input arithmetic circuit.

2.2.2 Square Arithmetic Programs

In this section, we define Square Arithmetic Programs (SAPs) according to Groth [56] and Groth and Maller [57]. Groth and Maller showed that replacing each of the constraints in QAPs with 2 other constraints, it is possible to design a Square Arithmetic Program (SAP), which is a QAP in which the two sets of polynomials involved in the quadratic term are identical. Using this technique, we can reduce the number of proof elements (at the cost of the circuit with twice as many multiplication gates).

Definition 2. (Square Arithmetic Program: SAP)
A Square Arithmetic Program (SAP) $\mathcal{S} = (\mathcal{A}, \mathcal{C}, t(x))$ over $\mathbf{Z}_p$ is a couple consisting of $2(m+1)$ polynomials $\mathcal{A} = \{a_i(x)\}_{i=0}^m$, $\mathcal{C} = \{c_i(x)\}_{i=0}^m$ and a target polynomial $t(x)$ such that $\deg(t(x)) \geq \max(\deg(a_i(x)), \deg(c_i(x)))$ for all $i \in [0, m]$. Let C be an arithmetic circuit over $\mathbf{Z}_p$ that has n inputs and n' outputs. We say that $\mathcal{S}$ verifies C if and only if for each inputs/outputs pair of C $(d_1, ..., d_{n+n'}) \in \mathbf{Z}_p^{n+n'}$ there exist $(d_{n+n'+1}, ..., d_m) \in \mathbf{Z}_p^{m-n-n'}$ such that let $p(x)$ define the bellow:

$$p(x) := \Big(a_0(x) + \sum_{i=1}^{m} d_i \cdot a_i(x)\Big)^2 - \Big(c_0(x) + \sum_{i=1}^{m} d_i \cdot c_i(x)\Big),$$

then $p(x)$ is divisible by the target polynomial $t(x)$. In other words, $(d_1, ..., d_{n+n'}) \in \mathbf{Z}_p^{n+n'}$ is an inputs/outputs pair of C if and only if there exist $(d_{n+n'+1}, ..., d_m) \in \mathbf{Z}_p^{m-n-n'}$ and $h(x)$ such that $p(x) = t(x)h(x)$.
We call the number m as the size of $\mathcal{S} = (\mathcal{A}, \mathcal{C}, t(x))$ and $\deg t(x)$ as its degree.

Theorem 2. (Groth [56], Groth and Maller [57])
Let C be an fan-in-2 type arithmetic circuit over $\mathbf{Z}_p$ that has d multiplication gates. Then there exist a degree $2d$ SAP $\mathcal{S} = (\mathcal{A}, \mathcal{C}, t(x))$ such that $\mathcal{S}$ verifies C.

SAP generation algorithm (Groth [56], Groth and Maller [57])
There exists a probabilistic and polynomial-time algorithm for generating SAP. Actually, an SAP instance can be easily generated from a QAP instance of C. We denote this algorithm by **SAP**.

$$\mathbf{SAP}(C) \rightarrow \mathcal{S} = (\mathcal{A}, \mathcal{C}, t(x)),$$

where C is an input arithmetic circuit.

2.2.3 zk-SNARKs

Here, we define a designated-verifier zk-SNARK. Let $\mathcal{R}$ be an **NP**-relation, and $(u, w) \in \mathcal{R}$. In this paper, we refer to an element u as statement (or public part) and w as the witness (or secret part) of $\mathcal{R}$. In addition, we denote the **NP**-relation defined using an arithmetic circuit C by $\mathcal{R}_C$ and its language by $\mathcal{L}_C$. The statement u corresponds to the public wire's value of C, and the witness corresponds to the value of the secret wire.

Definition 3. (Designated verifier non-interactive proof system)
Let $\mathcal{R}$ be an **NP**-relation. A designated verifier non-interactive proof system for $\mathcal{R}$ is a tuple of three polynomial-time algorithms, namely, Π=($\mathbf{G}, \mathbf{P}, \mathbf{V}$), as follows:

- $(CRS, VRS) \leftarrow \mathbf{G}(1^\lambda, \mathcal{R})$ takes security parameter λ and the **NP**-relation $\mathcal{R}$, and then outputs a common reference string (or public parameter) CRS and a trapdoor for verification VRS.
- $\pi \leftarrow \mathbf{P}(CRS, u, w)$ takes CRS and statement u, witness w pair for $\mathcal{R}$, and outputs the proof of knowledge π.
- **bool** $\leftarrow \mathbf{V}(VRS, u, \pi)$ takes VRS, statement u, and proof of knowledge π, and then outputs **true** if the proof π was accepted, otherwise outputs **false**. In this study, we assume that the verification algorithm $\mathbf{V}$ is a deterministic polynomial-time one.

Remark. In the above-mentioned definition, the word *designated-verifier* comes from the verification algorithm $\mathbf{V}$ that requires the secret information VRS. If the verification algorithm $\mathbf{V}$ does not require any secret information, i.e. only requires CRS, we refer to it as a public verifier non-interactive proof system.

Definition 4. (Completeness)
A designated verifier non-interactive proof system $\Pi=(\mathbf{G}, \mathbf{P}, \mathbf{V})$ has completeness if the prover has a valid pair (u, w) for $\mathcal{R}$. Accordingly,

$$\Pr[\ \mathbf{V}(VRS, u, \pi) = \mathbf{true} \mid \pi \leftarrow \mathbf{P}(CRS, u, w)] = 1 - \mathrm{negl}(\lambda).$$

Roughly speaking, the completeness means that the prover who has the correctly knowledge can pass the verification procedure.

Definition 5. (Knowledge Soundness)
A designated verifier non-interactive proof system $\Pi=(\mathbf{G}, \mathbf{P}, \mathbf{V})$ has knowledge-soundness if for any PPT adversary $\mathcal{A}$ and its PPT extractor $\mathcal{E}xt_{\mathcal{A}}$, the following is satisfied:

$$\mathrm{Adv}^{ksnd}_{\Pi,\mathcal{R},\mathcal{A},\mathcal{E}xt_{\mathcal{A}}} := \Pr[(u, w) \notin \mathcal{L} \wedge \mathbf{V}(VRS, u, \pi)] = \mathrm{negl}(\lambda),$$

where $(\mathcal{A}||\mathcal{E}xt_{\mathcal{A}})(CRS) \rightarrow (u, w, \pi)$.

Roughly speaking, the knowledge-soundness means that any PPT prover who does not have a correctly knowledge for $\mathcal{R}$ can not pass the verification procedure. In other words, if there exists a prover who can pass the verification procedure then it can extract witness w for some statement u. We also note that in the designated verifier non-interactive proof system, the adversary $\mathcal{A}$ can not access to VRS.

Notably, in the designated verifier non-interactive proof system, the adversary $\mathcal{A}$ cannot access VRS.

Definition 6. (Statistical Zero-knowledge)
A designated verifier non-interactive proof system $\Pi=(\mathbf{G}, \mathbf{P}, \mathbf{V})$ has statistical zero-knowledge if there exist PPT simulators $\mathcal{S}_1$ and $\mathcal{S}_2$ such that for any PPT distinguisher $\mathcal{D}$, the following two probabilistic distributions are statistically difficult to distinguish:

$$\Pr[\mathcal{D}(\pi) = 1|\mathbf{G}(1^{\lambda}, \mathcal{R}) \rightarrow (CRS, VRS), \mathcal{D}(CRS) \rightarrow (u, w), \mathbf{P}(CRS, u, w) \rightarrow \pi] \approx$$

$$\Pr[\mathcal{D}(\pi) = 1|\mathcal{S}_1(1^{\lambda}, \mathcal{R}) \rightarrow (CRS, VRS, trap), \mathcal{D}(CRS) \rightarrow (u, w), \mathcal{S}_2(CRS, u, trap) \rightarrow \pi].$$

Roughly speaking, the zero-knowledge means that using trapdoor for Π, the simulator who does not have the witness w can generate a valid proof π for u. In other words, we cannot get any information about witness w from proof π.

Definition 7. (Succinctness)
A designated verifier non-interactive proof system $\Pi=(\mathbf{G}, \mathbf{P}, \mathbf{V})$ is succinct if the proof size of π is $O(\lambda)$ and the processing time of the verification algorithm $\mathbf{V}$ is $O_\lambda(|u|)$.

Definition 8. (Designated zk-SNARK)
If a designated verifier non-interactive proof system has completeness, knowledge-soundness, zero-knowledge, and succinctness properties, then we refer to it as the designated verifier zk-SNARK.

2.2.4 LWE encryption

Security of LWE

Let $n, q \in \mathbf{Z}$ be integers, and χ be a discrete probability distribution over $\mathbf{Z}_q$. An LWE oracle $\mathcal{O}_{\mathbf{s}}^n$ for a fixed secret vector $\mathbf{s} \in \mathbf{Z}_q^n$ outputs $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbf{Z}_q^n \times \mathbf{Z}_q$, where $\mathbf{a} \in_{\$} \mathbf{Z}_q^n$ and $e \leftarrow \chi$.
In addition, let $\mathcal{O}_R$ be the random sampling algorithm on $\mathbf{Z}_q^n \times \mathbf{Z}_q$.

Definition 9. (Decisional LWE assumption)
The decisional LWE assumption holds for parameters (n, q, χ) if for any PPT adversary $\mathcal{A}$, one has the following:

$$\mathrm{Adv}_{n,q,\chi}^{LWE} := \left| \Pr[\mathcal{A}^{\mathcal{O}_{\mathbf{s}}^n}(1^\lambda) = 1] - \Pr[\mathcal{A}^{\mathcal{O}_R}(1^\lambda) = 1] \right| = \mathrm{negl}(\lambda).$$

Discrete Gaussian distribution
For any $\sigma \in \mathbf{R}_{>0}$ and

$$S_\sigma := \sum_{k=-\infty}^{k=\infty} e^{-\pi k^2/2\sigma^2},$$

we define the discrete Gaussian distribution χ_σ over $\mathbf{Z}$ with mean 0 and parameter σ as follows:

$$\Pr[X = x] := \frac{e^{-\pi x^2/2\sigma^2}}{S_\sigma}, \quad x \in \mathbf{Z}.$$

Regev [74] showed that solving the decisional LWE problem is as difficult as solving some lattice problems in the worst case.

Theorem 3. (Hardness of the decisional LWE [74])
Let χ_σ be the discrete Gaussian distribution with a parameter $\sigma \in (2\sqrt{n}, q)$, and $q < 2^{\mathrm{poly}(n)}$. The decisional LWE problem for parameters (n, q, χ_σ) is at least as difficult to solve as solving $\mathbf{GapSVP}_{\tilde{\mathcal{O}}(nq/\sigma)}$ (see [74] for the definition of the gap SVP problem).

In this paper, we assume the decisional LWE assumption for parameters (n, q, χ_σ).

LWE symmetric-key encryption scheme

We define an LWE-based symmetric-key encryption scheme according to [31, 44]. Let $\Gamma := (q, n, p, \sigma)$ be a system parameter for the LWE encryption, and we assume that q is an odd number and that $p|q$. We take a representative of $\mathbf{Z}_q$ as $\{\frac{-(q-1)}{2}, ..., \frac{q-1}{2}\}$.

Definition 10. (LWE symmetric-key encryption scheme)
The LWE symmetric-key encryption scheme is a tuple that comprises PPT algorithms, namely, ($\mathbf{KeyGen}, \mathbf{Enc}, and \mathbf{Dec}$), as follows:

- $\mathbf{s} \leftarrow \mathbf{KeyGen}(1^\lambda)$ takes a security parameter λ, and outputs the symmetric-key $\mathbf{s} \in \mathbf{Z}_q^n$.
- $ct \leftarrow \mathbf{Enc}(\mathbf{s}, m)$ takes the symmetric-key $\mathbf{s}$ and message m, and outputs a ciphertext $ct = (\mathbf{a}, b) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e + \frac{q}{p}m)$, where $\mathbf{a} \xleftarrow{\$} \mathbf{Z}_q^n$ is a random vector and $e \leftarrow \chi_\sigma$ an error term.
- $m' \leftarrow \mathbf{Dec}(\mathbf{s}, ct)$ takes the symmetric-key $\mathbf{s}$ and ciphertext $ct = (\mathbf{a}, b)$, and outputs $m' = \lfloor \frac{p}{q}(b - \langle \mathbf{a}, \mathbf{s} \rangle) \rceil \bmod p$, where $\lfloor \cdot \rceil$ denotes the integer rounding function.

In this scheme, if the absolute value of the error term of ciphertext ct satisfies $|e| < q/2p$, then the ciphertext is correctly decoded, i.e., $m = m'$.

Somewhat affine homomorphic property

The above-described LWE symmetric-key encryption scheme has the somewhat additive homomorphic property. Conversely, let $m_1, m_2 \in \mathbf{Z}_p$ be two messages, and

$$\mathbf{Enc}(\mathbf{s}, m_1) = (\mathbf{a}_1, \langle \mathbf{a}_1, \mathbf{s} \rangle + e_1 + \frac{q}{p}m_1)), \ \mathbf{Enc}(\mathbf{s}, m_2) = (\mathbf{a}_2, \langle \mathbf{a}_2, \mathbf{s} \rangle + e_2 + \frac{q}{p}m_2))$$

be their ciphertexts, respectively. If the sum of the error terms satisfies $|e_1 + e_2| < q/2p$, then

$$(\mathbf{a}_1 + \mathbf{a}_2, b_1 + b_2) = (\mathbf{a}_1 + \mathbf{a}_2, \langle \mathbf{a}_1 + \mathbf{a}_2, \mathbf{s} \rangle + e_1 + e_2 + \frac{q}{p}(m_1 + m_2))$$

is a valid ciphertext of $m_1 + m_2$. In this paper, we denote the aforementioned ciphertext by $\mathbf{Enc}(\mathbf{s}, m_1 + m_2)$ or $\mathbf{Enc}(m_1 + m_2)$. Generally, let $[c_i] \in \mathbf{Z}_p^n$ be a coefficient vector and $\mathbf{Enc}(\mathbf{s}, m_i) = (\mathbf{a}_i, b_i)$ $(i = 1, 2, ..., l)$ the ciphertexts. If the error term satisfies $|\sum_{i=1}^{l} c_i e_i| < q/2p$, then

$$\mathbf{Enc}(\mathbf{s}, \sum c_i m_i) = (\sum_{i=1}^{l} c_i \mathbf{a}_i, \sum_{i=1}^{l} c_i b_i)$$

is a valid ciphertext of $\sum_{i=1}^{l} c_i m_i \in \mathbf{Z}_p$.

As described above, because the error term becomes large in the affine operations, the error term of the initial encoding must be sufficiently small with respect to $q/2p$ in consideration of the later affine operations. We provide a detailed estimation of error terms in Section 2.7. We provide, here, an upper bound of the size of the error term after performing the affine homomorphic operation (see Banaszczyk [15] lemma 2.4 for the proof).

Lemma 1. (Error-term evaluation for affine operation)
For any $\sigma, T > 0$, and $\mathbf{c} = (c_1, c_2, \ldots, c_n) \in \mathbf{Z}^n$:

$$\Pr\Big[\mathbf{e} \leftarrow \chi_\sigma^n \mid \langle \mathbf{e}, \mathbf{c} \rangle \geq T\sigma||\mathbf{c}||\Big] < 2\exp(-\pi T^2).$$

We refer to the aforementioned parameter, T, as *tail cut parameter*. This Lemma shows that error term $e \leftarrow \chi_\sigma$ is included in $[-T\sigma, T\sigma]$ overwhelming in T (take n=1). If we take the tail cut parameter $T = 8$, the right-hand side of the inequality is less than $\exp(-200)$. Hence, in this paper, we regard this value as negligible and assume that $e \in [-8\sigma, 8\sigma]$.

Convert to Public-key scheme

The above symmetric-key cryptosystem can be used as a public-key cryptosystem by publishing a set of ciphertexts $\{\mathbf{Enc}(\mathbf{s}, 0)\}$. Conversely, let

$$pk = \{\mathbf{Enc}(\mathbf{s}, 0)\}_{j=1}^{t} = \{(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e_j)\}_{j=1}^{t}$$

be a public key and $m \in \mathbf{Z}_p$ a plaintext. We take a random subset $T \subset [1, t]$ and calculate the following:

$$\mathbf{Enc}(\mathbf{s}, m) := \big(\mathbf{0}, \frac{q}{p} m\big) + \sum_{j \in T} \mathbf{Enc}(\mathbf{s}, 0)_j.$$

Noise Smudging

In Section 2.3, we proposed our designated verifier zk-SNARKs, in which the verifier holds the secret-key $\mathbf{s}$ of the LWE encryption scheme and decrypts the LWE ciphertext during verification. As seen above, in this LWE encryption scheme, the verifier knows not only the plaintext m but also the error term e. Therefore, some information regarding the coefficients of the affine homomorphic operation may be disclosed to the verifier. Consequently, it is not zero-knowledge proof. To solve this problem, we use the noise-smudging technique by Boneh et al. [27]. This is a technique to hide the value of the original error term by adding a larger error term.

Lemma 2. (Noise Smudging [27])
For any $A, Z > 0$, and a fixed value $e_1 \in [-A, A]$, $e_2 \xleftarrow{\$} [-Z, Z]$. The statistical distance between distributions $e_1 + e_2$ and e_2 is equal to A/Z. Particularly, if A is negligible in Z, both the distributions are statistically difficult to distinguish.

Using this technique, we can hide the original error term e_1 from the verifier by adding a large error term $(\mathbf{0}, e_2)$ to the original ciphertext $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e_1 + mq/p)$. We provide a detailed estimation of e_1, e_2 in Section 2.7. Throughout this study, we assume that the prover executes the discussed noise-smudging process upon the generation of LWE ciphertexts, meaning that the decryption procedure does not reveal information regarding the coefficients of the affine homomorphic operation.

2.3 Designated zk-SNARKs using QAPs

Here, we propose two designated verifier zk-SNARKs using QAPs. The first proposed scheme is based on the Pinocchio-like construction, which is easy to implement using the libsnark library [4, 23] or Pinocchio-based systems. Furthermore, this scheme does not require the linear-only assumption, and we can prove its security using standard q-PKE and q-PDH assumptions. The second scheme is based on the construction by Groth [56], wherein the zero-knowledge proof is three LWE ciphertexts. In addition, we experimentally show that its setup and proof-generation algorithms are the fastest among the all known post-quantum zk-SNARKs, e.g., Gennaro et al. [52] and Nitulescu [70].

2.3.1 Proposal 1. Pinocchio-like LWE-based zk-SNARK

Throughout this section, we fix an LWE encryption system parameter $\Gamma := (q, n, p, \sigma)$ and assume that p is a prime number. Furthermore, we assume the decisional LWE assumption (see Section 2.2.4 Definition 9) for the parameters in Γ. For ensuring simplicity, we denote by $\mathbf{Enc}(m)$ an LWE ciphertext, which is generated by public-key (see Section 2.2.4 convert to public-key scheme) for a plaintext $m \in \mathbf{Z}_p$. In addition, we take a reduction parameter $R > 0$ for our security proof (knowledge soundness) and publish it as a system parameter together with Γ. In the verification process, **false** is outputted if the error term of the LWE ciphertext is greater than R. Furthermore, in the rest of this paper, we assume that the generator of LWE ciphertexts uses the noise-smudging technique to hide the original error term. Let C be a fan-in-2 arithmetic circuit that has d multiplication gates, and let $\mathcal{R}_C$ be the **NP**-relation defined by C.

Let $\mathbf{QAP}(C) \to \mathcal{Q} = (\mathcal{A}, \mathcal{B}, \mathcal{C}, t(x))$ be the degree d QAP that verifies C, and $\mathcal{A} = \{a_i(x)\}_{i=0}^m$, $\mathcal{B} = \{b_i(x)\}_{i=0}^m$, $\mathcal{C} = \{c_i(x)\}_{i=0}^m$, where m denotes the

number of gates in C. We also denote the index set of public input/output wire of C by $I_{public} \subset [0, m]$ and that of the other wire by $I_{mid} \subset [0, m]$. Notably, I_{public} (resp. I_{mid}) correspond to the state (resp. witness) part of C.

As shown in the following, we construct a designated verifier type non-interactive proof system Π=($\mathbf{G}, \mathbf{P}, \mathbf{V}$) for circuit C.

- Construction of $\mathbf{G}$.

The verifier generates $\alpha,\ \beta_a,\ \beta_b,\ \beta_c,\ s \xleftarrow{\$} \mathbf{Z}_p,\ \ t(s) \neq 0$, and the following secret/public key pair of the LWE encryption scheme $sk = \mathbf{s} \in \mathbf{Z}_p^n,\ pk = \{\mathbf{Enc}(0)\}_j$. In addition, the verifier generates the following LWE ciphertexts CRS:

$$CRS := \Big(\mathbf{Enc}(s), ..., \mathbf{Enc}(s^d),\ \mathbf{Enc}(\alpha s), ..., \mathbf{Enc}(\alpha s^d),$$

$$\{\mathbf{Enc}(\beta_a \cdot a_i(x))\}_i, \{\mathbf{Enc}(\beta_b \cdot b_i(x))\}_i, \{\mathbf{Enc}(\beta_c \cdot c_i(x))\}_i,\ pk = \{\mathbf{Enc}(0)\}_j,$$

$$\mathbf{Enc}(t(s)),\ \mathbf{Enc}(\alpha \cdot t(s)),\ \mathbf{Enc}(\beta_a \cdot t(s)),\ \mathbf{Enc}(\beta_b \cdot t(s)),\ \mathbf{Enc}(\beta_c \cdot t(s))\Big).$$

The verifier defines VRS as follows: $VRS := \big(CRS,\ sk,\ \alpha,\ \beta_a,\ \beta_b,\ \beta_c,\ s\big)$. Finally, the verifier outputs $(CRS, VRS) \leftarrow \mathbf{G}(1^\lambda, \mathcal{R}_C)$.

After generating (CRS, VRS), the verifier sends CRS to the prover.

- Construction of $\mathbf{P}$.

The prover, who has a correct state/witness pair $(u, w) \in \mathcal{R}_C$ and CRS, calculates all the value of wires $(d_1, ..., d_m) \in \mathbf{Z}_p^m$. The prover takes random values $\gamma_a,\ \gamma_b,\ \gamma_c \xleftarrow{\$} \mathbf{Z}_p$ and calculates

$$p(x) = \Big(\gamma_a t(x) + \sum_{i=0}^{m} d_i a_i(x)\Big)\Big(\gamma_b t(x) + \sum_{i=0}^{m} d_i b_i(x)\Big) - \Big(\gamma_c t(x) + \sum_{i=0}^{m} d_i c_i(x)\Big),$$

and $h(x) = \frac{p(x)}{t(x)} \ \in \mathbf{Z}_p[x]$, by using $\mathcal{Q} = (\mathcal{A}, \mathcal{B}, \mathcal{C}, t(x))$.

After calculating $h(x)$, the prover calculates $H := \mathbf{Enc}(h(s))$ by using $\{\mathbf{Enc}(s), ..., \mathbf{Enc}(s^d)\}$ and its somewhat affine homomorphic property. In addition, the prover calculates

$$A_{mid}(x) := \gamma_a t(x) + \sum_{i \in I_{mid}} d_i a_i(x), \qquad B(x) := \gamma_b t(x) + b_0(x) + \sum_{i=1}^{m} d_i b_i(x),$$

$$C(x) := \gamma_c t(x) + c_0(x) + \sum_{i=1}^{m} d_i c_i(x),$$

and its LWE ciphertexts

$$A := \mathbf{Enc}(A_{mid}(s)),\ \hat{A} := \mathbf{Enc}(\alpha A_{mid}(s)),\ B := \mathbf{Enc}(B(s)),\ \hat{B} := \mathbf{Enc}(\alpha B(s)),$$

$$C := \mathbf{Enc}(C(s)),\ \hat{C} := \mathbf{Enc}(\alpha C(s)),\ D := \mathbf{Enc}(\beta_a A_{mid}(s) + \beta_b B(s) + \beta_c C(s)),$$

by using CRS and affine homomorphic evaluations. Finally, the prover outputs a proof value π as:

$$\pi := (A, B, C, D, \hat{A}, \hat{B}, \hat{C}, H) \leftarrow \mathbf{P}(CRS, u, w).$$

- Construction of $\mathbf{V}$.
The designated verifier, who has VRS, receives a proof π from the prover and decrypts π by using the decryption function $\mathbf{Dec}(\mathbf{s}, -)$: $\mathbf{Dec}(\mathbf{s}, \pi) = \big(a, b, c, d, \hat{a}, \hat{b}, \hat{c}, h\big)$. We denote by $\{e_i\}$ the set of all the error terms from this decryption process. The verifier calculates

$$a' := a + \sum_{i \in I_{public}} d_i \cdot a_i(s), \quad d' = d + \beta_a \times \Big(\sum_{i \in I_{public}} d_i \cdot a_i(s) \Big),$$

and checks the following four conditions:

$$\text{(Linear combinations)} \qquad \alpha \cdot a = \hat{a}, \quad \alpha \cdot b = \hat{b}, \quad \alpha \cdot c = \hat{c} \tag{2.1}$$

$$\text{(Same coefficients)} \qquad d' = \beta_a \cdot a' + \beta_b \cdot b + \beta_c \cdot c \tag{2.2}$$

$$\text{(Divisibility)} \qquad a' \cdot b - c = h \cdot t(s) \tag{2.3}$$

$$\text{(Error term)} \qquad |e_i| \leq R \quad \text{for all } i. \tag{2.4}$$

If proof π satisfies the aforementioned four conditions, then **true** is outputted, otherwise **false** is outputted.
Remark: Noise smudging. Using the noise-smudging technique, the prover can hide the original error term from the decryption process of the verifier, meaning that the above-mentioned LWE decryption procedure does not reveal the information regarding the coefficients of the affine homomorphic operations.

Theorem 4.
Let d be the degree of $\mathbf{QAP}(C)$. We assume the d-PKE, $2d$-PDH, and $3d+3$-PDH assumptions (see Section 2.4 for their definitions). The proposed scheme 1 is a designated verifier zk-SNARK with soundness error $1/p$. Conversely, this scheme holds completeness, knowledge soundness, (statistical) zero-knowledge, and succinctness.

We provide a proof of this theorem in Section 2.5.

2.3.2 Proposal 2. More efficient LWE-based zk-SNARK using QAPs

We propose another designated verifier zk-SNARK using QAPs. The zero-knowledge proof in this scheme comprises three LWE ciphertexts, and this scheme is a generalization of the scheme by Nitulescu [70] (taking $a_i(x) = b_i(x)$ and $\alpha = \beta$ in the below construction and security proof). We use the

same notations as used in Proposal 1. Let $\mathcal{Q} = (\mathcal{A}, \mathcal{B}, \mathcal{C}, t(x))$ be a QAP for C.
- Construction of $\mathbf{G}$.
The verifier generates random numbers as:

$$\alpha, \beta, \delta, s \xleftarrow{\$} \mathbf{Z}_p,\ (t(s) \neq 0), sk = \mathbf{s} \xleftarrow{\$} \mathbf{Z}_q^n,$$

and defines (CRS, VRS) in the following data set:

$$CRS := \Big(\mathbf{Enc}(\alpha), \mathbf{Enc}(\beta), \mathbf{Enc}(\delta), \{\mathbf{Enc}(s^i)\}_{i=0}^{d}, pk = \{\mathbf{Enc}(0)\}_j$$

$$\big\{\mathbf{Enc}\Big(\frac{\beta a_i(s) + \alpha b_i(s) + c_i(s)}{\delta}\Big)\big\}_{i \in I_{mid}}, \{\mathbf{Enc}\Big(\frac{s^i t(s)}{\delta}\Big)\}_{i=0}^{d}\Big),$$

and $VRS := (CRS, \alpha, \beta, \delta, s, sk)$.
Finally, the verifier outputs $(CRS,\ VRS) \leftarrow \mathbf{G}(1^\lambda, \mathcal{R}_C)$ and sends CRS to the prover.
- Construction of $\mathbf{P}$.
The prover, who has a correct state/witness pair $(u, w) \in \mathcal{R}_C$, calculates $(d_1, ..., d_m) \in \mathbf{Z}_p^m$ for the value of each wire. The prover also calculates $h(x)$ such that

$$\Big(\sum_{k=0}^{m} d_i \cdot a_i(x)\Big) \cdot \Big(\sum_{k=0}^{m} d_i \cdot b_i(x)\Big) = \Big(\sum_{k=0}^{m} d_i \cdot c_i(x)\Big) + h(x)t(x).$$

In addition, the prover takes random numbers $\gamma_a,\ \gamma_b \xleftarrow{\$} \mathbf{Z}_p$ and calculates

$$A(s) := \alpha + \sum_{i=0}^{m} d_i a_i(s) + \gamma_a \delta, \quad B(s) := \beta + \sum_{i=0}^{m} d_i b_i(s) + \gamma_b \delta,$$

$$C(s) := \frac{\sum_{i \in I_{mid}} d_i(\beta a_i(s) + \alpha b_i(s) + c_i(s))}{\delta} + \gamma_a A(s) + \gamma_b B(s) - \gamma_a \gamma_b \delta.$$

Using CRS, the prover can generate LWE ciphertexts (A, B, C) such that

$$A := \mathbf{Enc}(A(s)),\ B := \mathbf{Enc}(B(s)),\ C := \mathbf{Enc}(C(s)).$$

Finally, the prover outputs $\pi := (A, B, C) \leftarrow \mathbf{P}(CRS, u, w)$.
- Construction of $\mathbf{V}$.
The designated verifier, who has $VRS = \big(CRS,\ \alpha,\ \beta,\ \delta,\ s, sk\big)$, obtains $\pi = (A, B, C)$ from the prover. The verifier decrypts π by using $\mathbf{Dec}(sk, -)$ and obtains $\mathbf{Dec}(\mathbf{s}, \pi) = \big(a, b, c\big)$. After obtaining (a, b, c), the verifier checks the following equations:

$$a \cdot b = \alpha \cdot \beta + \sum_{i \in I_{public}} d_i\big(\beta a_i(s) + \alpha b_i(s) + c_i\big) + c \cdot \delta, \tag{2.5}$$

If the aforementioned equation holds, **V** outputs **true**, otherwise outputs **false**.

Our experimental results show that this the Setup/Proof-generation algorithms of this scheme are the fastest among those of the other schemes. However, the security proof of this scheme requires the linear-only encryption assumption (see Section 2.4 or [28, 29]). This assumption is stronger than the security assumptions in Proposal 1.

Theorem 5.
The second proposed designated verifier non-interactive proof system $\Pi = (\mathbf{G}, \mathbf{P}, \mathbf{V})$ is a zk-SNARK under the linear-only encryption assumption with soundness error $(4d + 2)/p$.

We provide a proof of this theorem in Section 2.5.

2.3.3 The shortest LWE-based zk-SNARK using SAP

We propose a designated verifier zk-SNARK using SAP. This scheme is a special case of proposed scheme 2, in which we are taking $a_i(x) = b_i(x)$ for all i. The zero-knowledge proof of this scheme consists of only two LWE ciphertexts. Furthermore, the construction of this scheme is very similar to the scheme proposed by Nitulescu [70], and the rest of this paper, we call this scheme as Nitulescu's scheme. We also use the same notations in the proposal 1 and 2.
Let $\mathcal{S} = (\mathcal{A}, \mathcal{C}, t(x))$, $\mathcal{A} = \{a_i(x)\}$, $\mathcal{C} = \{c_i(x)\}$ be a square arithmetic program for C.
- Construction of **G**.
The verifier generates random numbers as: $\alpha, \beta, \delta, s \xleftarrow{\$} \mathbf{Z}_p$, $(t(s) \neq 0)$, $sk = \mathbf{s} \xleftarrow{\$} \mathbf{Z}_q^n$, and define (CRS, VRS) as the following data set:

$$CRS := \Big(\mathbf{Enc}(\alpha), \mathbf{Enc}(\beta), \mathbf{Enc}(\delta), \{\mathbf{Enc}(s^i)\}_{i=0}^{d},$$

$$\big\{\mathbf{Enc}\Big(\frac{\beta a_i(s) + \alpha a_i(s) + c_i(s)}{\delta}\Big)\big\}_{i \in I_{mid}}, \{\mathbf{Enc}\Big(\frac{s^i t(s)}{\delta}\Big)\}_{i=0}^{d},\ pk = \{\mathbf{Enc}(0)\}_j\Big),$$

and $VRS := (CRS, \alpha, \beta, \delta, s, sk)$. Finally, the verifier outputs $(CRS,\ VRS) \leftarrow \mathbf{G}(1^\lambda, \mathcal{R}_C)$ and send CRS to the prover.
- Construction of **P**.
The prover, who has a correct state/witness pair $(u, w) \in \mathcal{R}_C$, calculates $(d_1, ..., d_m) \in \mathbf{Z}_p^m$ for each wire's value. The prover also calculates $h(x)$ such that

$$\Big(\sum_{k=0}^{m} d_i \cdot a_i(x)\Big)^2 = \Big(\sum_{k=0}^{m} d_i \cdot c_i(x)\Big) + h(x)t(x).$$

In addition, The prover takes random numbers $\gamma_a \xleftarrow{\$} \mathbf{Z}_p$ and defines

$$A(s) := \alpha + \sum_{i=0}^{m} d_i a_i(s) + \gamma_a \delta,$$

$$C(s) := \frac{\sum_{i \in I_{mid}} d_i(\beta a_i(s) + \alpha a_i(s) + c_i(s))}{\delta} + 2\gamma_a A(s) - \gamma_a^2 \delta.$$

Using CRS, the prover can generate LWE ciphertexts (A, C) such that

$$A := \mathbf{Enc}(A(s)),\ C := \mathbf{Enc}(C(s)).$$

Finally, the prover outputs $\pi := (A, C) \leftarrow \mathbf{P}(CRS, u, w)$.
- Construction of $\mathbf{V}$.
The designated verifier, who has $VRS = \big(CRS,\ \alpha,\ \beta,\ \delta,\ s, sk\big)$, obtains $\pi = (A, C)$ from the prover. The verifier decrypts π using $\mathbf{Dec}(sk, -)$ and gets $\mathbf{Dec}(\mathbf{s}, \pi) = \big(a, c\big)$. After getting $(a, c, \{e_a, e_c\})$, the verifier checks the following equations:

$$a^2 = \alpha \cdot \beta + \sum_{i \in I_{public}} d_i\big(\beta a_i(s) + \alpha a_i(s) + c_i\big) + c \cdot \delta, \tag{2.6}$$

$$|e_a|,\ |e_c| \leq R. \tag{2.7}$$

If the above equations hold, then $\mathbf{V}$ outputs **true**, the otherwise outputs **false**.

Theorem 6. The above designated verifier non-interactive proof system $\Pi = (\mathbf{G}, \mathbf{P}, \mathbf{V})$ is zk-SNARK under the Linear-Only Vector Encryption assumption (see Section 2.4 for the definition).

This theorem is a special case of Proposal 2.

2.4 Security Assumptions

We describe the security assumptions, i.e., q-PKE, q-PDH, and linear-only encryption assumptions, used in the security proof. These are also defined in [51, 39, 55, 26, 28, 29].

2.4.1 q-PKE and q-PDH assumption

q-PKE Game
$(sk, pk) \leftarrow \mathbf{KeyGen}(1^\lambda)$. $\alpha, s \xleftarrow{\$} \mathbf{Z}_p$. $\sigma \leftarrow \big(pk, \mathbf{Enc}(1), \ldots, \mathbf{Enc}(s^q), \mathbf{Enc}(\alpha), \ldots,\ \mathbf{Enc}(\alpha s^q)\big)$. $(\mathbf{Enc}(c), \mathbf{Enc}(c'); a_0, \ldots, a_q) \leftarrow (\mathcal{A} \mid\mid \mathcal{E}xt_{\mathcal{A}})(\sigma, aux)$. return $[\alpha * c = c'] \wedge [c \neq \sum_{i=0}^{q} a_i s^i]$.

An encryption scheme **Enc** satisfies the q-power knowledge of exponent (q-PKE) assumption if for any PPT attacker $\mathcal{A}$ and its extractor $\mathcal{E}xt_{\mathcal{A}}$, the following holds:

$$\mathbf{Adv}_{\mathcal{A},\mathcal{E}xt_{\mathcal{A}}}(\lambda) := Pr\big[q\text{−PKE Game} = 1\big] = \text{negl}(\lambda).$$

q-PDH Game
$(sk, pk) \leftarrow \mathbf{KeyGen}(1^\lambda)$. $s \xleftarrow{\$} \mathbf{Z}_p$. $\sigma \leftarrow \big(pk, \mathbf{Enc}(1), \ldots, \mathbf{Enc}(s^{q-1}),\ \mathbf{Enc}(s^{q+1}), \ldots, \mathbf{Enc}(s^{2q})\big)$. $c \leftarrow \mathcal{A}(\sigma)$. return $[\mathbf{Dec}(sk, c) = s^q]$.

An encryption scheme **Enc** satisfies the q-power Diffie–Hellman (q-PDH) assumption if for any PPT attacker $\mathcal{A}$, the following holds

$$\mathbf{Adv}_{\mathcal{A}}(\lambda) := Pr\big[q\text{−PDH Game} = 1\big] = \ negl(\lambda).$$

2.4.2 Linear-Only Encryption Assumption.

Here, we define the linear-only encryption assumption (see [26, 28, 29] for more information). Let $\Pi = (\mathbf{KeyGen}, \mathbf{Enc}, \mathbf{Dec})$ be a symmetric-key encryption scheme with affine homomorphic property, such as the LWE encryption scheme in Section 2.2.4. In addition, let λ is a security parameter and aux the all auxiliary input data. Let $\mathcal{C}$ be a challenger and $\mathcal{A}$ a PPT attacker, then there exists an efficient extractor $\mathcal{E}xt_{\mathcal{A}}$.

Linear-Only Encryption Game
$sk \leftarrow \mathbf{KeyGen}(1^\lambda)$. $(ct_1, ct_2, \ldots, ct_n) \leftarrow \mathcal{C}(sk, aux)$: generates ciphertexts s.t. $ct_i = \mathbf{Enc}(sk, m_i)$. $ct' \leftarrow \mathcal{A}(\{ct_i\}, aux)$. $(a_1, a_2, \ldots, a_n, b) \leftarrow \mathcal{E}xt_{\mathcal{A}}(\{ct_i\}, aux)$. return $[\mathbf{Dec}(sk, ct') \neq \sum_{i=1}^{n} a_i m_i + b]$.

A symmetric-key encryption scheme $\Pi = (\mathbf{KeyGen}, \mathbf{Enc}, \mathbf{Dec})$ satisfies the linear-only encryption assumption if for any PPT attacker $\mathcal{A}$ and its extractor $\mathcal{E}xt_{\mathcal{A}}$, the probability that the above game returns **true** is negligible.

2.5 Security proofs

2.5.1 Security proof for Proposal 1

Theorem 7.
Let d be the degree of $\mathbf{QAP}(C)$. We use the d-PKE, $2d$-PDH, and $3d + 3$-PDH assumptions (see Section 2.4 for their definitions). Proposal 1 is a designated verifier zk-SNARK with soundness error $1/p$. Conversely,

this scheme holds completeness, knowledge soundness, (statistical) zero-knowledge, and succinctness.

Proof of Succinctness and Completeness.

The completeness is trivial from the construction. In addition, proof π comprises eight LWE ciphertexts. Therefore, the size of $|\pi|$ is $O(\lambda)$, and the computational complexity of the verification process is $O_\lambda(|u|))$ on the basis of the construction $\square$.

Proof of Zero-Knowledge.

We construct PPT simulators $\mathcal{S}im_1, \mathcal{S}im_2$ in the following manner.
$\mathcal{S}im_1$ generates (CRS, VRS) by using $\Pi.\mathbf{G}$, and after generating

$$VRS = \big(CRS,\ sk,\ \alpha,\ \beta_a,\ \beta_b,\ \beta_c,\ s\big),$$

$\mathcal{S}im_1$ gives some parameters to $\mathcal{S}im_2$ as $trap := (\alpha,\ \beta_a,\ \beta_b,\ \beta_c,\ s)$.
In addition, $\mathcal{S}im_2$ can generate proof π by using $trap$ as follows:
$\mathcal{S}im_2$ generates $\delta_a, \delta_b, \delta_c \in_\$ \mathbf{Z}_p$, and

$$\begin{aligned} A &:= \mathbf{Enc}(\delta_a),\ \hat{A} := \mathbf{Enc}(\alpha\delta_a), \\ B &:= \mathbf{Enc}(\delta_b),\ \hat{B} := \mathbf{Enc}(\alpha\delta_b), \\ C &:= \mathbf{Enc}(\delta_c),\ \hat{C} := \mathbf{Enc}(\alpha\delta_c), \\ D &:= \mathbf{Enc}(\beta_a \cdot \delta_a + \beta_b \cdot \delta_b + \beta_c \cdot \delta_c). \end{aligned}$$

In addition, $\mathcal{S}im_2$ calculates

$$\tilde{a} := \delta_a + \sum_{i \in I_{public}} d_i \cdot a_i(s),$$

$$h := \frac{\tilde{a} \cdot \delta_b - \delta_c}{t(s)}, \quad H := \mathbf{Enc}(h),$$

and outputs π as:

$$\pi := (A, B, C, D, \hat{A}, \hat{B}, \hat{C}, H) \leftarrow \mathcal{S}im_2(CRS, u, trap).$$

We can assume that all the error terms of LWE ciphertexts are bounded by the reduction parameter R.
Because of the construction, it is easy to verify that

$$\mathbf{Dec}(\mathbf{s}, \pi) = \big(a, b, c, d, \hat{a}, \hat{b}, \hat{c}, h\big),$$

holds for the four conditions in the verification algorithm $\mathbf{V}$. In addition, these probability distributions are statistically difficult to distinguish from

the output of $\Pi.\mathbf{P}(CRS, u, w)$ with correct witness w. Notably, using the noise-smudging technique, the error terms in the decryption process do not reveal information regarding the affine homomorphic operation. Therefore, the outputs for the real and simulated proofs from any distinguisher $\mathcal{D}$ are the same value overwhelming in λ. □

Proof of Knowledge Soundness.

Let $\Pi = (\mathbf{G}, \mathbf{P}, and\mathbf{V})$ be our proposed non-interactive proof system for circuit C and **NP**-relation $\mathcal{R}_C$ defined by C. If there exists a PPT attacker and extractor $(\mathcal{A}_{snd}, \mathcal{E}xt_{\mathcal{A}_{snd}})$ for the knowledge soundness that has a non-negligible advantage, then we show that using $(\mathcal{A}_{snd}, \mathcal{E}xt_{\mathcal{A}_{snd}})$, there is a PPT attacker $\mathcal{A}_{PDH}$, for $2d$-PDH or $(3d+3)$-PDH problems, who has a non-negligible advantage. Let

$$\mathcal{Q} = (\mathcal{A}, \mathcal{B}, \mathcal{C}, t(x)),$$

$$\mathcal{A} = \{a_i(x)\}, \mathcal{B} = \{b_i(x)\}, \mathcal{C} = \{c_i(x)\}$$

be a QAP for C. Notably, $\mathcal{A}_{PDH}$ generates common reference strings / verifiable reference strings (CRS, VRS) (the details of the method to generate (CRS, VRS) are given in this proof). In addition, $\mathcal{A}_{PDH}$ inputs CRS to $(\mathcal{A}_{snd}, \mathcal{E}xt_{\mathcal{A}_{snd}})$, and obtains (u, w, π) from $(\mathcal{A}_{snd}, \mathcal{E}xt_{\mathcal{A}_{snd}})$.
Meanwhile, because $\mathcal{A}_{snd}$ has a non-negligible advantage, one has

$$(u, w) \notin \mathcal{L}_C \wedge \Pi.\mathbf{P}(VRS, u, \pi) = \mathbf{true}$$

with a non-negligible probability.
We denote $\pi = (A, B, C, D, \hat{A}, \hat{B}, \hat{C}, H)$ and $\mathrm{Dec}(\mathbf{s}, \pi) = (a, b, c, d, \hat{a}, \hat{b}, \hat{c}, h)$. Because the equation of linear combinations, we can get this equation

$$\alpha \cdot a = \hat{a}.$$

Therefore, using the d-PKE assumption for

$$\mathbf{Enc}(s), ..., \mathbf{Enc}(s^d), \quad \mathbf{Enc}(\alpha s), ..., \mathbf{Enc}(\alpha s^d),$$

there exist a PPT attacker and extractor $(\mathcal{A}_{PKE}, \mathcal{E}xt_{\mathcal{A}_{PKE}})$ such that if $(\phi_0, \phi_1, \ldots, \phi_d)$ is the output of $\mathcal{E}xt_{\mathcal{A}_{PKE}}$, then the following equations hold:

$$\mathrm{Enc}\big(\sum_{i=0}^{d} \phi_i s^i\big) = A, \quad \mathrm{Enc}\big(\sum_{i=0}^{d} \alpha\phi_i s^i\big) = \hat{A}.$$

We can apply the same arguments for $(b, \hat{b})$, $(c, \hat{c})$ and obtain $(\psi_0, \psi_1, \ldots, \psi_d)$, $(\omega_0, \omega_1, \ldots, \omega_d)$ from $\mathcal{E}xt_{\mathcal{A}_{PKE}}$ such that

$$\mathrm{Enc}\big(\sum_{i=0}^{d} \psi_i s^i\big) = B, \quad \mathrm{Enc}\big(\sum_{i=0}^{d} \alpha\psi_i s^i\big) = \hat{B}$$

$$\mathrm{Enc}\big(\sum_{i=0}^{d} \omega_i s^i\big) = C, \quad \mathrm{Enc}\big(\sum_{i=0}^{d} \alpha\omega_i s^i\big) = \hat{C}$$

In addition, we define

$$\phi_{mid}(x) := \sum_{i=0}^{d} \phi_i x^i, \quad \psi(x) := \sum_{i=0}^{d} \psi_i x^i, \quad \omega(x) := \sum_{i=0}^{d} \omega_i x^i,$$

$$\phi(x) := \phi_{mid}(x) + \sum_{k \in I_{public}} d_k a_k(x),$$

the following claim holds.

Claim

There exist $\gamma_a, \gamma_b, \gamma_c \in \mathbf{Z}_p$, and $\{d_k\}_{k \in I_{mid}}$ such that

$$\phi(x) = \gamma_a t(x) + \sum_{k \in I_{mid}} d_k a_k(x) + \sum_{k \in I_{public}} d_k a_k(x),$$

$$\psi(x) = \gamma_b t(x) + \sum_{k \in I_{mid}} d_k b_k(x) + \sum_{k \in I_{public}} d_k b_k(x),$$

$$\omega(x) = \gamma_c t(x) + \sum_{k \in I_{mid}} d_k c_k(x) + \sum_{k \in I_{public}} d_k c_k(x).$$

Because we show this claim at the end of this proof, we assume it here.
The adversary $(\mathcal{A}_{snd}, \mathcal{E}xt_{\mathcal{A}_{snd}})$ does not have a correct witness. If we define $w := \{d_k\}_{k \in I_{mid}}$, then $(u, w) \notin \mathcal{R}_{\mathcal{C}}$.
Because of the definition of QAP, $t(x)$ does not divide $p(x) := \phi(x)\psi(x) - \omega(x)$. We define $H(x) := p(x)/t(x)$, as according to the equation of divisibility, $H(s) = h$. Let $(x - r)$ be a polynomial that divides $t(x)$ but not $p(x)$, and let $T(x) := t(x)/(x - r) \in \mathbf{Z}_p[x]$. Because $t(x)$ and $p(x)$ have degrees at most d and $2d$, respectively, $\mathcal{A}_{snd}$ can use "the extended Euclidean algorithm for polynomials" to find polynomials $A(x)$ and $B(x)$ of degrees $2d - 1$ and $d - 1$, respectively, such that $A(x)t(x) + B(x)p(x) = T(x)$. Dividing the aforementioned equation by $t(x)$ on both the sides, we have $A(x) + B(x)H(x) = 1/(x - r)$. Therefore, we define

$$\Delta(x) := A(x)(x - r) + hB(x)(x - r) - 1 \neq 0,$$

and $\deg(\Delta(x)) = 2d$, and $\Delta(s) = 0$.
Let $L \neq 0$ be the leading term of polynomial $\Delta(x)$, then we have the following:

$$\begin{aligned} \Delta(x) &= Lx^{\delta} + (\text{lower term}) \\ \Longrightarrow s^{\delta} &= -L^{-1}(\text{lower term})(s), \\ \Longleftrightarrow s^{2d} &= -L^{-1}s^{2d-\delta}(\text{lower term})(s). \end{aligned}$$

Therefore,

$$\mathrm{Enc}(s^{2d}) = \mathrm{Enc}\big(- L^{-1}s^{2d-\delta}(\text{lower term})(s)\big).$$

The aforementioned equation asserts that LWE ciphertext $\mathrm{Enc}(s^{2d})$ can be expressed as an affine combination of lower degree terms $\mathrm{Enc}(s^i),\ (i < 2d)$. This is a contradiction to the $2d$-PDH assumption.

Proof of the Claim.
We define the following:

$$\tilde{\phi}(x) := \phi(x) - \sum_{k \in I_{public}} d_k a_k(x),$$

$$\tilde{\psi}(x) := \psi(x) - \sum_{k \in I_{public}} d_k b_k(x),$$

$$\tilde{\omega}(x) := \omega(x) - \sum_{k \in I_{public}} d_k c_k(x),$$

then $\max(\deg\tilde{\phi}(x), \deg\tilde{\psi}(x), \deg\tilde{\omega}(x)) \leq d$. It is sufficient to prove the claim we show the degree shift polynomial $F(x)$ as:

$$F(x) := \theta_a\tilde{\phi}(x) + \theta_b x^{d+1}\tilde{\psi}(x) + x^{2d+2}\tilde{\omega}(x), \quad \deg(F) \leq 3d + 2$$

is included in the vector space on $\mathbf{Z}_p$ as follows:

$$V := \mathrm{Span}_{\mathbf{Z}_p}\Big\langle t(x),\ x^{d+1}t(x),\ x^{2d+2}t(x),$$

$$\zeta_i(x) = (\theta_a a_i(x) + \theta_b x^{d+1}b_i(x) + \theta_c x^{2d+2}c_i(x))_{i \in I_m id}\Big\rangle.$$

However, if $F(x)$ is not included in the vector space, since Gennaro et al. [51] Lemma 10, there exists a degree $3d + 3$ polynomial $U(x)$ such that the x^{3d+3}'s coefficient of $U(x) \times F(x)$ is not 0 with an overwhelming probability (that is equal to $1-1/p$, where $1/p$ is the soundness error of this scheme). We briefly describe the proof of this lemma here. We take a random polynomial $U(x)$ such that all the $x^{(3d+3)}$'s coefficients of

$$U(x)t(x),\ U(x)x^{d+1}t(x),\ U(x)x^{2d+2}t(x),\ U(x)\zeta_i(x)$$

are equal to 0. Thereby meaning that the reverse coefficients vector of $U(x)$ is included in the orthogonal complement space of V. Using $F(x) \notin V$, we can take such $U(x)$. In addition, the x^{3d+3}'s coefficient of $U(x) \times F(x)$ is the inner product value of the reverse coefficients vector of $U(x)$ and coefficients vector of $F(x)$. Both the vectors are not included in V, and the inner product value is a random element of $\mathbf{Z}_p$. Therefore, this is not 0 with the probability of $1 - 1/|\mathbf{Z}_p|$. This is a proof sketch of the above-mentioned lemma.
We can write

$$U(s) \times F(s) = U(s)\theta_a\tilde{\phi}(s) + U(s)\theta_b s^{d+1}\tilde{\psi}(s) + U(s)\theta_c s^{2d+2}\tilde{\omega}(s)$$

$$= U(s)\theta_a\big(\phi(s) - \sum_{public} d_k a(s)\big) + U(s)\theta_b s^{d+1}\big(\psi(s) - \sum_{public} d_k b(s)\big)$$

$$+U(s)\theta_c s^{2d+2}\big(\omega(s) - \sum_{public} d_k c(s)\big).$$

In advance, when $\mathcal{A}_{PDH}$ generates (CRS, VRS), we take $\theta_a, \theta_b, and \theta_c$ that satisfy the following:

$$U(s)\theta_a = \beta_a, \quad U(s)\theta_b s^{d+1} = \beta_b, \quad U(s)\theta_c s^{2d+2} = \beta_c.$$

Then, according to the same coefficient equation in the verification procedure, one has the following:

$$U(s) \times F(s) = d' - \beta_a \sum_{public} d_k a(s) - \beta_b \sum_{public} d_k b(s) - \beta_c \sum_{public} d_k c(s)$$

$$= d' - \sum_{public} d_k(\beta_a a(s)) - \sum_{public} d_k(\beta_b b(s)) - \sum_{public} d_k(\beta_c c(s)) \in \mathbf{Z}_p. \quad (2.8)$$

The ciphertext of this value can be generated using π, which is the output of adversary $\mathcal{A}_{snd}$, and CRS by using the affine homomorphic property. However,

$$U(s) \times F(s) = (\text{higher terms}) + a \cdot s^{3d+3}$$

$$+(\text{lower terms}), \quad a \in \mathbf{Z}_p. \quad (2.9)$$

Therefore, we can generate ciphertext $\mathrm{Enc}(s^{3d+3})$ using $\{\mathrm{Enc}(s^i)\}_{i \neq 3d+3}$, π, and CRS. This is a contradiction to the $(3d+3)$-PDH assumption. □
Remark: Reduction parameter. According to the proof of the above-mentioned theorem, in the construction of attacker $\mathcal{A}_{PDH}$, we need further affine homomorphic operations for π, which is the output of soundness attacker $\mathcal{A}_{snd}$. Therefore, if the error terms of the LWE ciphertexts of zero-knowledge proof π are considerably noisy, we cannot execute other operations. Therefore, we take a reduction parameter R, which is detailed in Section 2.7.

2.5.2 Security Proof for Proposal 2

The zero-knowledge proof size in this scheme is smaller than that of Proposal 1, and our experimental results show that the Setup/Proof-generation algorithms of this scheme are the fastest among those of the other schemes. However, the security proof of this scheme requires the linear-only encryption assumption (see Section 2.4 or [28, 29]). This assumption is stronger than the security assumptions in Proposal 1.

Theorem 8.
The second proposed designated verifier non-interactive proof system $\Pi = (\mathbf{G}, \mathbf{P}, \mathbf{V})$ is a zk-SNARK under the linear-only encryption assumption with soundness error $(4d + 2)/p$.

Proof of Completeness and Succinctness.

The completeness and succinctness are trivial from the construction. □

Proof of Zero-knowledge.

We construct PPT simulators, namely, Sim_1 and Sim_2, in the following manner.
Sim_1 generates CRS and VRS in the construction $\mathbf{G}$, and it takes a trapdoor for Sim_2 as $trap := (\alpha, \beta, \delta, s)$, similar to that in the proof of the previous theorem. Sim_2 outputs $\pi = (A, B, C)$ for given CRS and state u the following method. Sim_2 generates random values $a, b \in_{\$} \mathbf{Z}_p$, and it calculates

$$C(s) := \frac{ab - \alpha\beta - \sum_{i \in I_{public}} d_i(\beta a_i(s) + \alpha b_i(s) + c_i(s))}{\delta}.$$

Using CRS, Sim_2 can generate LWE ciphertexts $\pi = (A, B, C)$ with noise smudging such that

$$A := \mathbf{Enc}(a),\ B := \mathbf{Enc}(b),\ C = \mathbf{Enc}(C(s)).$$

It is easy to verify that $\pi = (A, B, C)$ satisfies the equation (see 2.5) in the verification procedure and that these distributions are statistically difficult to distinguish from the output of $\Pi.\mathbf{P}(CRS, u, w)$ with correct witness w. □

Proof of Knowledge Soundness.

We show that if there exists a PPT knowledge soundness adversary $\mathcal{A}_{snd}$

and its extractor $\mathcal{E}xt_{\mathcal{A}_{snd}}$ who has a non-negligible advantage, then we can extract a witness. Let a set of LWE ciphertexts $\pi = (A, B, C)$ be an output from $\mathcal{A}_{snd}$ that satisfies the equation (see Section 3.2 (2.5)). Using the linear-only encryption assumption, we have

$$A(\alpha, \beta, \delta, s) = A_\alpha \alpha + A_\beta \beta + A_\delta \delta + A(s)$$

$$+ \sum_{i \in I_{mid}} A_i \frac{\beta a_i(s) + \alpha b_i(s) + c_i(s)}{\delta} + A_h(s) \frac{t(s)}{\delta},$$

where $A_\alpha, A_\beta, A_\delta, A_i \in \mathbf{Z}_p$, and $A(s), A_h(s)$ are the polynomials of degree d, and we can ignore the coefficients of $\mathbf{Enc}(0)$. Similarly, we can write $B(\alpha, \beta, \delta, s)$ and $C(\alpha, \beta, \delta, s)$.
Schwartz–Zippel lemma says that the equation (2.5) holds when viewing A, B, and C as formal polynomials indeterminates α, β, δ, s with probability $(1 - \frac{4d+2}{p})$.

Lemma 3. (Schwartz-Zippel)
Let p be a prime, and let $f \in \mathbf{Z}_p[x_1, ..., x_n]$ be a multivariate polynomial of total degree d that is not identically zero. Then,

$$\Pr[(\alpha_1, \ldots, \alpha_n) \leftarrow \mathbf{Z}_p^n \mid f(\alpha_1, \ldots, \alpha_n) = 0] \leq d/p.$$

See Boneh et al. [29] for the proof of this lemma. Furthermore, we assume that soundness error $\frac{4d+2}{p}$ is negligible (this value is negligible in practical use cases and real-world parameters; see Section 2.7).

By comparing coefficient α^2 of the equation (2.5), $A_\alpha B_\alpha = 0$. Therefore, $A_\alpha = 0$ or $B_\alpha = 0$. Without loss of generality, we can take $B_\alpha = 0$. Furthermore, by comparing coefficient $\alpha\beta$, we have $1 = A_\alpha B_\beta + A_\beta B_\alpha = A_\alpha B_\beta$. This means $AB = AB_\beta \times A_\alpha B$; therefore, we can also assume that $A_\alpha = B_\beta = 1$. By comparing the coefficient of β^2, we have $A_\beta B_\beta = A_\beta = 0$. Next, comparing the coefficient of $1/\delta^2$, we have

$$\Big(A_h(s)t(s) + \sum_{i \in I_{mid}} A_i(\beta a_i(s) + \alpha b_i(s) + c_i(s))\Big) \times$$

$$\Big(B_h(s)t(s) + \sum_{i \in I_{mid}} B_i(\beta a_i(s) + \alpha b_i(s) + c_i(s))\Big) = 0.$$

Without loss of generality, we also assume

$$\Big(A_h(s)t(s) + \sum_{i \in I_{mid}} A_i(\beta a_i(s) + \alpha b_i(s) + c_i(s))\Big) = 0.$$

Next, we have $B_h(s)t(s) + \sum_{i \in I_{mid}} B_i(\beta a_i(s) + \alpha b_i(s) + c_i(s)) = 0$ because

$$\alpha \times \frac{B_h(s)t(s) + \sum_{i \in I_{mid}} B_i(\beta a_i(s) + \alpha b_i(s) + c_i(s))}{\delta} = 0.$$

Therefore, we have

$$A(\alpha, \beta, \delta, s) = \alpha + A_\delta \delta + A(s),\ B(\alpha, \beta, \delta, s) = \beta + B_\delta \delta + B(s).$$

Since the verification equation (2.5), that involve α (resp. β) give us

$$\alpha B(s) = \sum_{i \in I_{public}} d_i \alpha b_i(s) + \sum_{i \in I_{mid}} C_i \alpha b_i(s),$$

$$\beta A(s) = \sum_{i \in I_{public}} d_i \beta a_i(s) + \sum_{i \in I_{mid}} C_i \beta a_i(s).$$

If we define $C_i = d_i$ for $i \in I_{mid}$, then we have

$$A(s) = \sum_{i=0}^{m} d_i a_i(s), \quad B(s) = \sum_{i=0}^{m} d_i b_i(s).$$

Finally, we check at the coefficients of s^i,

$$\Big(\sum_{i=0}^{m} d_i a_i(s)\Big) \times \Big(\sum_{i=0}^{m} d_i b_i(s)\Big) = \sum_{i=0}^{m} d_i c_i(s) + C_h(s)t(s).$$

From the aforementioned equation, it is evident that $C_i = d_i$ for $i \in I_{mid}$ are the correct witnesses of statement $\{d_i\}$ for $i \in I_{pubic}$. This is a contradiction of the assumption that $\mathcal{A}_{\text{snd}}$ does not have a correct witness. □

2.6 zk-SNARKs on permissioned blockchains

In this section, we describe the system configuration when applying the proposed schemes from the previous section to permissioned blockchains for example, Quorum [7] and Hyperledger Fabric [3]. Furthermore, we propose a simple multi-party protocol that securely generates CRS. In this protocol, we assume that at least one management node is honest. The blockchain user can convince that the CRS is correctly generated by random numbers. Therefore, it is possible to prevent attacks where malicious management nodes install trapdoors when CRS is generated such as Campanelli et al. [34].

2.6.1 Designated verifier zk-SNARK on permissioned blockchain

In blockchain systems, zero-knowledge proofs are generally used when a user sends a transaction that contains private information. More specifically, the user encrypts the data payload to be concealed in the transaction, generates an encrypted transaction, and sends it to the blockchain system. At

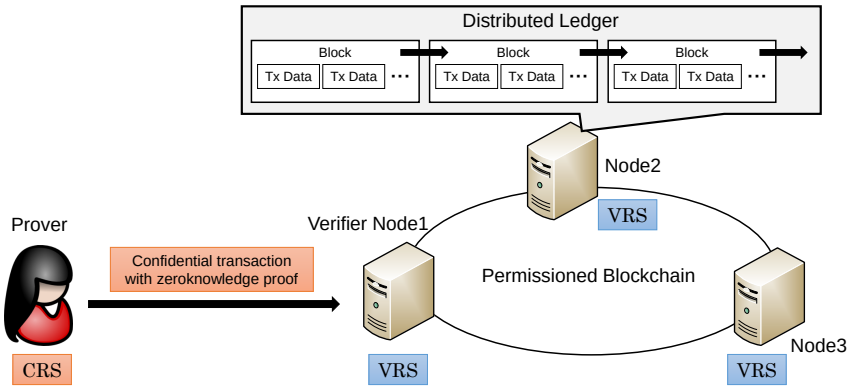

Figure 2.1: Designated verifier zk-SNARK on permissioned blockchain

that time, a zero-knowledge proof is also attached to the transaction to prove that the encrypted part follows the rules of the protocol. For example, in Zcash protocol [11] or Monero protocol [5], the user encrypts the remittance amount to conceal it from others. In these protocols (using the Unspent Transaction Output (UTXO) transaction model), the sum of the input amount and the sum of the output amount match, and the transfer amount must be a non-negative integer value. The transaction sender must prove it to the miner nodes and attach a zero-knowledge proof for it to the encrypted transaction. In other words, in the context of zero-knowledge proofs, the prover is a blockchain user that generates an encrypted transaction, and the verifier is a node that approves the transaction. In the permissionless blockchain system such as Bitcoin [69], the transaction approval is performed by unspecified nodes in the network. Therefore, VRS can not include secret information. Since VRS that have to be shared between these nodes, when using the designated-verifier type zero-knowledge proof. Therefore, it is unrealistic. On the other hand, in permissioned blockchains, since a specific management nodes $Node_1, Node_2, \ldots, Node_n$ perform transaction approval, the secure generation and sharing of VRS is possible. In the next subsection, we propose a secure method for our LWE-based designated verifier zk-SNARKs.

In this section, we propose a simple method for generating and sharing CRS and VRS of proposed schemes among management nodes $Node_1, \ldots, Node_n$ in the permissioned blockchain. First, it is assumed that there is a private blockchain whose participants are only $Node_1, \ldots, Node_n$, for generating CRS and VRS. This blockchain is used as a bulletin board that is difficult to forge for sharing information among the management nodes. Hereinafter, in this section, transmission and reception of information among the management nodes are performed using this private blockchain (bulletin board). As a first step, we show a secure and random generation al-

gorithm for an element $s \in \mathbf{Z}_q$ among management nodes $Node_1, \ldots, Node_n$. This is a simple three-phase commitment protocol, and it is a well-known sharing protocol using a bulletin board and a commitment scheme (for example, ElGamal cryptographic version see Hans Delfs and K. Helmut [41] chap. 4.4.6). We simply changed their commitment scheme to LWE encryption to achieve the quantum attack resistance. After executing Algorithm 1, all management nodes can share a random element $s \in \mathbf{Z}_q$. It is easy

Input: a modulus $q \in \mathbf{Z}$, and system parameters of LWE encryption scheme.
Output: a random number $s \in \mathbf{Z}_q$. , and all nodes $Node_i$ share it.
1: Generate $\mathbf{k}_i \in \mathbf{Z}_p^n$ as an LWE symmetric key.
2: Generate a random number $s_i \in \mathbf{Z}_q$, and an LWE ciphertext $c_i = \mathrm{Enc}(\mathbf{k}_i, s_i)$.
3: Write c_i to the private blockchain as the commitment value.
4: After all $\{c_i\}$ have been written, write the secret key $\mathbf{k}_i$ to the blockchain.
5: Decrypt all $\{c_i\}$, and calculates $s := \sum s_i$.
6: Write s to the private blockchain.
7: If the all written values s are the same one, then accept to s.
8: else Go to Step1.

Algorithm 1: Simple Secure Multi-party Sampling from $\mathbf{Z}_q$

to verify that Algorithm 1 outputs a random element $s \in \mathbf{Z}_q$, if at least one node is honest. However, when a management node aborts in the middle of the algorithm, then this procedure stops. Thus each management node is a single point of failure only during the generation of (CRS, VRS). Note that if q is a prime number then $\mathbf{Z}_q$ is the finite field of order q. Therefore, we can randomly sample the elements of the finite field. We denote this sampling algorithm by $\mathbf{RandZ}_q() \to s \in \mathbf{Z}_q$. This algorithm is used as a subroutine to generate a (CRS, VRS) pair.

Next, we propose a multi-party sampling algorithm for the error term of LWE ciphertexts in CRS according to the discrete Gaussian distribution (see Section 2.2.4). If at least one node is honest, then we can show this algorithm is correctly distributed according to the discrete Gaussian distribution. This algorithm is based on the rejection sampling method proposed by Gentry et al. [53] and Ducas and Nguyen [45]. Let $\mathbf{FP}_m$ be the set of floating point numbers with m-bit mantissa. In other words, an element $a \in \mathbf{FP}_m$ consists of three pairs $a = (s, e, v)$, $s \in \{1, -1\}$ is the sign, $e \in \mathbf{Z}_t$ is exponent, $v \in \mathbf{Z}_{2^m-1}$ is the mantissa, and $a = s \times 2^{e-m} \times v \in \mathbf{R}$. If we set the parameter of the algorithm $\mathbf{RandZ}_q()$ as $q = 2$, t, and $2^m - 1$, then we can construct a multi-party sampling algorithm for $\mathbf{FP}_m$. We denote this multi-party algorithm by $\mathbf{RandFP}_m() \to a \in \mathbf{FP}_m$. Furthermore, if we take

$s = 1$ and $e = 0$, then we can construct a sampling algorithm for floating point number from the interval $[0, 1)$. We denote this multi-party algorithm by $\mathbf{NRandFP}_m() \to a \in [0, 1)$.

Input: a standard deviation $\sigma \in \mathbf{FP}_m$, and a tail cut parameter $\tau \in \mathbf{FP}_m$.
Output: $x \in \mathbf{Z}$, with distribution statistically close to χ_σ on $\mathbf{Z}$.
1: **Calculating** $h := -\pi/\sigma^2 \in \mathbf{FP}_m$.
2: **Sampling** $x \leftarrow \mathbf{Z}_{2\tau\sigma} = [-\tau\sigma, \tau\sigma]$: using $\mathbf{RnadZ}_{2\tau\sigma}()$ among $Node_1 \sim Node_n$.
3: **Calculating** $p := \exp(h \cdot x^2)$
4: **Sampling** $r \leftarrow [0, 1)$ using $\mathbf{NRnadFP}_m()$ among $Node_1 \sim Node_n$.
5: **If** $r < p$ **then** return x.
6: **else** Go to Step2.

Algorithm 2: Rejection Sampling for Discrete Gaussian on $\mathbf{Z}$

The Algorithm 2 outputs statistical close to the discrete Gaussian distribution χ_σ (see [53, 45]), and we take the tail cut parameter $T = 8$ and $\tau\sigma \in \mathbf{Z}$ for $\tau \geq T$. At this time, as shown in Section 2.2.4, $\Pr\big[|x| > \tau\sigma| \ \ x \leftarrow \chi_\sigma\big] < e^{-200}$. We assume this probability is a negligible value; as a result, the all outputs of χ_σ are included in the interval $[-\tau\sigma, \tau\sigma]$ overwhelming τ. We denote this sampling algorithm by $\mathbf{GSamp}(\sigma) \to e \in \mathbf{Z}$. Using Algorithm 2, we can construct a securely and randomly sampling algorithm for (CRS, VRS) among the management nodes $Node_1, Node_2, \ldots, Node_n$.

Input: $\mathcal{Q} = (\mathcal{A}, \mathcal{B}, \mathcal{C}, t(x))$, $\{\texttt{Plain Text}\}$, and LWE parameters $\Gamma = (q, n, p, \sigma)$.
Output: (CRS, VRS) for $\mathcal{Q}$.
1: Generate $VRS = (\alpha,\ \beta_a,\ \beta_b,\ \beta_c,\ s,\ \mathbf{sk}) \leftarrow \mathbf{RandZ}_q()^5 \times \mathbf{RandZ}_q()^n$
2: While $\{\texttt{Plain Text}\} \neq \phi$ Do
3: $m \in \{\texttt{Plain Text}\}$
4: generate $e \leftarrow \mathbf{GSamp}(\sigma)$ and $\mathbf{a} \leftarrow \mathbf{RandZ}_q^n$
5: $ct := (\mathbf{a}, \langle \mathbf{a}, \mathbf{sk} \rangle + e + \frac{q}{p}m)$
6: $CRS := CRS \cup \{ct\}$ and $\{\texttt{Plain Text}\} := \{\texttt{Plain Text}\} \setminus \{m\}$.
7: End while return (CRS, VRS).

Algorithm 3: Multi-party Sampling of CRS and VRS

Let C be a public arithmetic circuit and $\mathcal{Q} = (\mathcal{A}, \mathcal{B}, \mathcal{C}, t(x))$ (resp. $\mathcal{S} = (\mathcal{A}, \mathcal{C}, t(x))$) be a its QAP (resp. SAP) expression. We denote the plaintext set of ciphertexts in CRS by $\{\texttt{Plain Text}\}$. By using Algorithm 3, all management nodes can securely share the same (CRS, VRS) for our proposed scheme 1, 2, and 3 and publish VRS to the blockchain users.

2.7 Concrete parameters

Here, we estimate various parameters of our proposed schemes. For the ease of comparison with other works, each parameter was referenced to the values used in other papers as much as possible.

First, our zero-knowledge proof of Proposal 1 comprises eight (resp. Proposal 2 comprises three) LWE ciphertexts, if λ is a security parameter, then $|\pi| = 8 \times |ct|$ (resp. $|\pi| = 3 \times |ct|$), where $|ct|$ denotes the length of the LWE ciphertext with security parameter λ.

Next, we evaluate the size of LWE ciphertexts for practical use-cases. First, we take the size of the plaintext space as $p = 2^{32} - 5$, and the statistical parameter $\kappa = 32$ as in [52, 44, 38].

Subsequently, we evaluate parameters q and σ in the LWE encryption scheme on the basis of the error-term evaluation. As mentioned in the previous sections, we must consider the following two conditions.

- The prover generates LWE ciphertexts as proof π from CRS via affine homomorphic operations and noise smudging, following which all the error terms are bounded by reduction parameter R.
- There exists a margin for the affine operations in the knowledge-soundness proof after generating proof π.

We must take the parameters that all error terms that occur in these two operations are bounded by $\frac{q}{2p}$. Furthermore, we only evaluate the error terms for proposed scheme 1 because it has higher error growth than that of proposed scheme 2. If we take appropriate parameters (q, σ) for proposed scheme 1, the error terms that occurred in scheme 2 are bounded by $\frac{q}{2p}$.

We assume that tail cut parameter $T = 8$. Because Lemma 1, all the error terms in $\pi = (A, B, C, D, \hat{A}, \hat{B}, \hat{C}, H)$ (before performing the noise-smudging process) are bounded by

$$\sigma \times 8p\sqrt{3m + 5}.$$

In addition, we execute the noise-smudging process by using statistical parameter $\kappa = 32$, and then we set $A/Z = 2^{-32}$ using Lemma 2. Therefore, the reduction parameter R is

$$|\text{error terms of } \pi| < 2^{32} \times \sigma \times 8p\sqrt{3m + 5} = R \ \ \big(\leq \frac{q}{2p} \big).$$

Furthermore, we also execute the affine homomorphic operations to prove the knowledge soundness (see Section 2.5) of these LWE ciphertexts, whose errors are bounded by R. The highest complexity affine homomorphic operation is given by equations (2.8) and (2.9) that constitute the $(3d+3)$-PDH attacker. The size of the error term in the right-hand side of equation (2.8) is $R + \sigma \times 8p\sqrt{3m}$ from Lemma 1. Furthermore, when equation (2.9) is

solved for s^{3d+3}, the polynomial $U(x)F(x)$, whose degree is $6d+5$, grows error terms. Therefore, again using Lemma 1, the conditional for the error term is

$$p \cdot \left(R + (\sigma \times 8p\sqrt{3m}) + (\sigma \times 8p\sqrt{6d+5})\right)$$
$$= 8p^2\sigma(2^{32}\sqrt{3m+5} + \sqrt{3m} + \sqrt{6d+5}) \leq \frac{q}{2p}.$$

This is a relation between moduli (p, q) when statistical parameter $\kappa = 32$. For example, if we apply $m = 2^{22}, d = 2^{15}$, and $p = 2^{32}$, the size and degree are sufficient for many practical use-cases such as Zcash (see [11, 20, 71]). We can obtain the following:

$$\sigma \times 2^{144} \leq q.$$

This is the condition for (σ, q). Next, we evaluate the bit length of our CRS. The CRS in the proposed scheme 1 (resp. 2) comprises $2(d+1)+3m+5 = 3m+2d+7$ (resp. $m+2d+5$) LWE ciphertexts and pk. Therefore, we have

$$|CRS| = (3m + 2d + 7) \times (n + 1) \times \log q + |pk|,$$

$$(\text{resp. } (m + 2d + 5) \times (n + 1) \times \log q + |pk|).$$

If we take parameters $m = 2^{22} and d = 2^{15}$, then

$$\approx 2^{23.58} \times (n + 1) \times \log q + |pk|, \ (\text{resp. } 2^{22} \times (n + 1) \times \log q + |pk|).$$

Furthermore, assuming the random oracle model and using PRNG,

$$|CRS| = \lambda + (2^{23.58} + \#pk) \times \log q, \ (\text{resp. } \lambda + (2^{22} + \#pk) \times \log q),$$

and proof size $|\pi| = 8 \times (n+1)\log q$ (resp. $3 \times (n+1)\log q$). We list our experimental results in Table 2.1 using the security parameters recommended in [52]. This security parameter is known as a secure parameter against LWE encryption attacks, such as lattice reduction attacks [65] and Gröbner basis attack [14].

2.8 Evaluation

2.8.1 Summary of implementation

We implemented our proposed schemes and the one by Nitulescu [70], which are zk-SNARKs for arithmetic circuits, by using the libsnark library [4] and library GMP [9]. Subsequently, we evaluated their performances. We also implemented the schemes by Gennaro et al. [52] and Pinocchio [71, 23] (see Table 2.1) but do not provide the detailed comparison here because the scheme by Gennaro et al. targets Boolean circuits and that of Pinocchio is a pre-quantum scheme. Therefore, we cannot directly compare them.

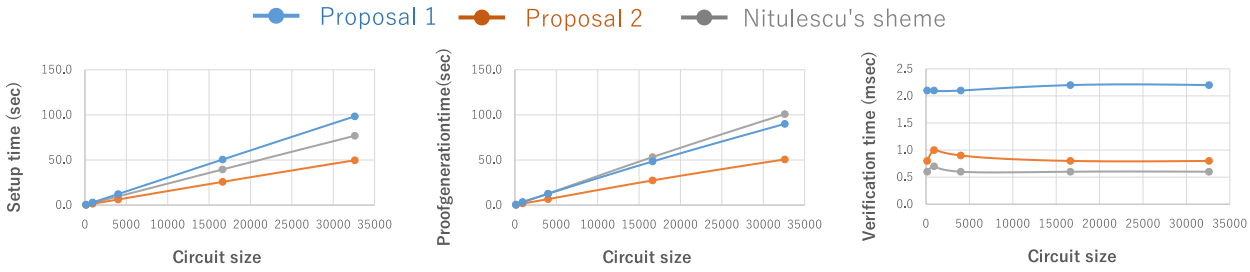

Figure 2.2: Experimental results: setup, proof-generation, and verification algorithms by using PRNG for CRS generation.

In addition, we chose the prime $p = 2^{32} - 5$, modulus $q = 2^{736}$, and dimension of lattice $n = 1470$, as in Gennaro et al. [52].

We employed a PRNG for generating $|CRS|$. An LWE ciphertext $(\mathbf{a}, b) \in \mathbf{Z}_q^n \times \mathbf{Z}_q$ has the bit length of $(n+1) \times \log q$. If we apply a practical parameter of LWE, the size of CRS is significantly large. In practical use cases, we generate a λ-bit random number r, and replace all the $\mathbf{a}$ in CRS by the output of $\mathbf{a} \leftarrow \mathrm{PRNG}(r)$. Using this technique, we can significantly reduce the size of CRS and prove that the LWE encryption is secure when using the random oracle model [49]. Furthermore, we used AES-256 in the counter mode as a PRNG for taking advantage of AES-NI instructions.

We performed the benchmarks of our protocols on a single thread of an Intel Core i7-9700K CPU @ 3.60 GHz. Our experimental results are depicted in Fig. 2.2. We implemented circuits with sizes $2^7, 2^{11}, 2^{13}, 2^{15}$, and 2^{16}. As depicted in Fig. 2.2, the processing times of our setup and proof-generation algorithms increase linearly, and those of the verification algorithms are almost small constants. In addition, our results show that Proposal 2 is the most efficient scheme because its setup and proof-generation algorithms, which are the bottleneck of zk-SNARKs, are the fastest among all the other schemes. For example, when the circuit size is 2^{16}, 49.7 s are taken to generate a CRS and 50.7 s to generate a proof (see Table 2.3 in Section 2.9.2 for more details). These processing times are two or three times faster than those of the scheme by Gennaro et al. [52] and approximately one or two orders of magnitude slower than those of the scheme by Pinocchio [4, 71, 23] (see Table 2.1). Therefore, we must further optimize our proposals to use them in real systems. For example, Proposal 2 takes approximately 500 s and 300 GB memory to generate a proof for the HMAC-SHA-256 circuit. We provide additional experimental data and optimization techniques in Section 2.9. For example, Proposal 1 is appropriate for a multi-thread CPU. Accordingly, we implemented it in an 8-thread CPU environment. Consequently, this scheme is eight times faster than the original one, and its performance is almost the same as that of Proposal 2 (see Fig. 2.4).

2.8.2 Discussion

In this paper, we have described the construction of zero-knowledge proof with quantum computer resistance. In many current blockchain systems, not only zero-knowledge proofs but also digital signature schemes are based on the discrete logarithm problem. Therefore, the question arises as to whether it is meaningful to make the quantum computer resistant only to the zero-knowledge proof method. In the blockchain system, security by a digital signature and privacy protection of ledger information should be considered separately. The ledger information remains even after the blockchain system is no longer used, thus giving quantum computer resistance has a specific meaning.

2.9 Additional experimental data and optimization

Here, we report the additional experimental data and some optimization techniques.

Methodology of our experiment: We implemented our proposed schemes and the one by Nitulescu by using the libsnark library [4]. More precisely, we implemented the zero-knowledge proof for the Bubble sort algorithm of N items using it. We set the number of items as $N = 3, 7, 14, 28,$ and 39 and each circuit size as $|C| = 132, 924, 4004, 16632,$ and 32604. Our experimental environment comprises an Intel Core i7-9700K CPU @ 3.60 GHz with a single thread.

2.9.1 Optimization techniques

We developed the following simple but effective optimization techniques for our implementation:

1. **Loop-optimization for the LWE encryption on the libsnark** In the setup algorithm, we generated many LWE ciphertexts for CRS and, therefore, used pseudo random numbers from AES-NI instructions. Using our LWE encryption scheme for the libsnark library, we converted the pseudo random numbers to LWE ciphertexts. This conversion included multiple loop operations, and we refactored this procedure and reduced many multiplications between large integers.

2. **The 0-fragmentation technique for a sparse expression** A QAP/SAP expression is often very sparse. For instance, Ben-Sasson et al. [23] reported that the percentage of zero polynomials in a QAP instance $\mathcal{Q} = (\mathcal{A}, \mathcal{B}, and \mathcal{C})$ is typically approximately $48\%, 85\%, 29\%$, respectively. Therefore, we set the 0-flag for each polynomial and skipped the procedure in the proof-generation algorithm. Because of

this technique, we could reduce many additions and multiplications of large integers.

The above-mentioned technique 1 reduced the setup time by approximately 40% compared with the original one, and technique 2 also reduced the proof-generation time by approximately 30% compared with the original one.

Experimental results of proposed schemes without PRNG

We implemented all the schemes in the libsnark library [4], and we did not use a pseudo random number generator for generating $|CRS|$. Therefore, these implementations required considerable memory; for example, Proposal 1 required 33 GB memory for the circuit size of 32,604 (see the right-hand side of Fig. 2.3 for more details). However, we think this memory size is somewhat impractical. Of course, these proof-generation algorithms of this scheme are faster than those of other schemes, as there is no PRNG procedure involved in this scheme. In addition, our experimental results show that the setup time, proof-generation time, and size of $|CRS|$ linearly increased according to the circuit size, with the other parameters being (almost) constants.

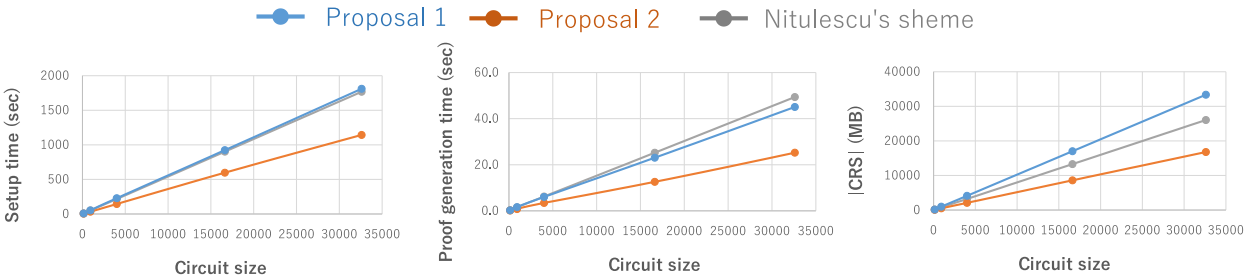

Figure 2.3: Experimental results of the proposed algorithms without PRNG.

Experimental results of the multi-thread optimization

We improved the setup and proof-generation algorithms by using a multi-thread CPU. Generally, the setup and proof-generation algorithms are the bottlenecks in zk-SNARK. We can compute these procedures in parallel. For example, the proof-generation algorithm in proposed scheme 1 generates 8 LWE ciphertexts, and it can be calculated in parallel using 8 threads. Therefore, we can radically reduce the setup and proof-generation times. However, this method depends on the CPU environment, and, therefore, we

report it as reference results. As depicted in Figure 2.4 (see Table 2.4 in Section 2.9.2 for more details), the setup and proof-generation algorithms are improved using the multi-thread architecture. The processing time of Proposal 1 is almost the same as that of Proposal 2; they look the almost same line in Fig. 2.4 because Proposal 1 is more appropriate for the multi-thread architecture than other schemes. Precisely, the zero-knowledge proof of Proposal 1 comprises 8 LWE ciphertexts, and we can compute it in parallel. Therefore, we allocated one thread to each ciphertext. However, the proof of Proposal 2 (resp. the scheme by Nitulescu) comprises 3 (resp. 2) LWE ciphertexts, and the effect of the multi-thread operation is limited. We detail the experimental results in Section 2.9.2.

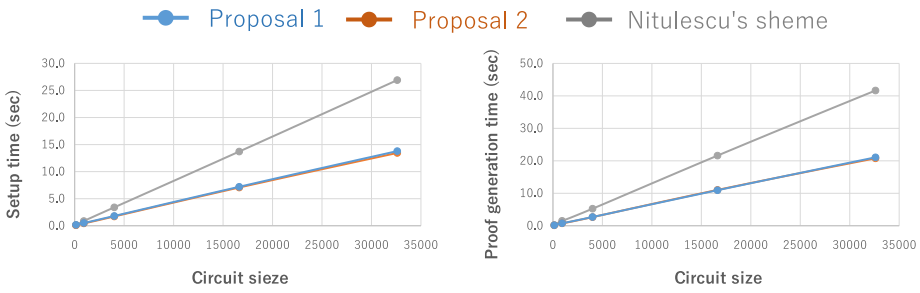

Figure 2.4: Experimental results of the proposed algorithm with PRNG and multi-thread optimization.

2.9.2 More detailed data

For completeness, we report the additional experimental data in the next page. We implemented our proposed schemes and the one by Nitulescu [70] in the libsnark library [4], by using the library GMP [9] and evaluated their performances. In Table 2.3, we show the experimental results of these schemes with PRNG for CRS generation. Considering the trade-off between time and memory, we consider this architecture the best practice. In Table 2.4, we show the experimental results of these schemes with PRNG and multi-thread optimization. The processing performance of the first scheme is the almost same as that of the second scheme. Notably, these schemes require a multi-thread CPU environment.

Table 2.2: Experimental results of proposed schemes without PRNG.

	Circ. size	Setup	Proof gen.	Verification	$\|CRS\|$	$\|VRS\|$	$\|\pi\|$
Proposal 1	132	7.7 s	0.2 s	2.4 msec	144,9 MB	135.7 KB	1,082.7 KB
	924	51.34 s	1.6 s	2.5 msec	970.7 MB	136.1 KB	1,082.7 KB
	4,004	228.4 s	6.0 s	2.5 msec	4,118.6 MB	136.6 KB	1,082.7 KB
	16,632	923.7 s	23.1 s	2.5 msec	17,024.5 MB	137.7 KB	1,082.7 KB
	32,604	1,810.8 s	45.1 s	2.5 msec	33,357.4 MB	139.6 KB	1,082.7 KB
Proposal 2	132	4.9 s	0.1 s	1.0 msec	72.1 MB	135.5 KB	406.0 KB
	924	32.9 s	0.8 s	1.0 msec	493.0 MB	135.8 KB	406.0 KB
	4,004	144.1 s	3.4 s	1.0 msec	2,077.8 MB	136.2 KB	406.0 KB
	16,632	597.4 s	12.6 s	1.0 msec	8,594.9 MB	137.1 KB	406.0 KB
	32,604	1,143.5 s	25.2 s	1.1 msec	16,791.3 MB	137.8 KB	406.0 KB
Proposal 3	132	7.7 s	0.2 s	0.7 msec	111.1 MB	135.5 KB	270.7 KB
	924	51.9 s	1.5 s	0.7 msec	766.7 MB	135.7 KB	270.7 KB
	4,004	216.2 s	6.2 s	0.8 msec	3,214.3 MB	136.2 KB	270.7 KB
	16,632	901.9 s	25.2 s	0.7 msec	13,293.0 MB	137.1 KB	270.7 KB
	32,604	1,767.6 s	49.4 s	0.7 msec	26,043.2 MB	138.5 KB	270.7 KB

Table 2.3: Experimental results of the proposed schemes with PRNG for CRS generation.

	Circ. size	Setup	Proof gen.	Verification	$\|CRS\|$	$\|VRS\|$	$\|\pi\|$
Proposal 1	132	0.4 s	0.5 s	2.4 ms	0.1 MB	135.7 KB	1,082.7 KB
	924	2.9 s	3.3 s	2.5 ms	0.9 MB	136.1 KB	1,082.7 KB
	4,004	12.2 s	12.5 s	2.5 ms	3.8 MB	136.6 KB	1,082.7 KB
	16,632	50.5 s	48.3 s	2.5 ms	15.6 MB	137.7 KB	1,082.7 KB
	32,604	98.4 s	90.0 s	2.5 ms	30.5 MB	139.6 KB	1,082.7 KB
Proposal 2	132	0.2 s	0.2 s	1.0 ms	0.07 MB	135.5 KB	406.0 KB
	924	1.5 s	1.8 s	1.0 ms	0.5 MB	135.8 KB	406.0 KB
	4,004	6.2 s	6.6 s	1.0 ms	1.9 MB	136.2 KB	406.0 KB
	16,632	25.8 s	27.3 s	1.0 ms	7.9 MB	137.1 KB	406.0 KB
	32,604	49.7 s	50.7 s	1.1 ms	15.4 MB	137.8 KB	406.0 KB
Nitulescu [70]	132	0.3 s	0.5 s	0.7 ms	0.1 MB	135.5 KB	270.7 KB
	924	2.3 s	3.1 s	0.7 ms	0.7 MB	135.7 KB	270.7 KB
	4,004	9.5 s	12.7 s	0.8 ms	2.9 MB	136.2 KB	270.7 KB
	16,632	39.5 s	53.2 s	0.7 ms	12.2 MB	137.1 KB	270.7 KB
	32,604	76.9 s	100.9 s	0.7 ms	23.8 MB	138.5 KB	270.7 KB

Table 2.4: Experimental results of the proposed schemes with PRNG and multi-thread optimization.

	Circ. size	Setup	Proof gen.	Verification	$\|CRS\|$	$\|VRS\|$	$\|\pi\|$
Proposal 1	132	0.2 s	0.2 s	2.4 ms	0.1 MB	135.7 KB	1,082.7 KB
	924	0.5 s	0.7 s	2.5 ms	0.9 MB	136.1 KB	1,082.7 KB
	4,004	1.8 s	2.7 s	2.5 ms	3.8 MB	136.6 KB	1,082.7 KB
	16,632	7.2 s	10.9 s	2.5 ms	15.6 MB	137.7 KB	1,082.7 KB
	32,604	13.8 s	21.1 s	2.5 ms	30.5 MB	139.6 KB	1,082.7 KB
Proposal 2	132	0.1 s	0.1 s	1.0 ms	0.07 MB	135.5 KB	406.0 KB
	924	0.4 s	0.7 s	1.0 ms	0.5 MB	135.8 KB	406.0 KB
	4,004	1.7 s	2.7 s	1.0 ms	1.9 MB	136.2 KB	406.0 KB
	16,632	7.1 s	11.0 s	1.0 ms	7.9 MB	137.1 KB	406.0 KB
	32,604	13.5 s	20.8 s	1.1 ms	15.4 MB	137.8 KB	406.0 KB
Nitulescu [70]	132	0.2 s	0.2 s	0.7 ms	0.1 MB	135.5 KB	270.7 KB
	924	0.9 s	1.5 s	0.7 ms	0.7 MB	135.7 KB	270.7 KB
	4,004	3.4 s	5.3 s	0.8 ms	2.9 MB	136.2 KB	270.7 KB
	16,632	13.7 s	21.6 s	0.7 ms	12.2 MB	137.1 KB	270.7 KB
	32,604	26.9 s	41.7 s	0.7 ms	23.8 MB	138.5 KB	270.7 KB

Chapter 3

Generic Construction of Polynomial Commitment Scheme and its Application for ZKPs without Trusted Setup

3.1 Summary

In recent years, *zero-knowledge proof (ZKP)* protocols [17, 54, 62] have drawn significant attention as privacy-enhancing technologies in various domains, especially the cryptocurrency and blockchain industries. In response to this situation, various zero-knowledge proof schemes based on various cryptographic security assumptions have been proposed [51, 71, 21, 56, 32, 83, 66, 33, 84, 60, 48, 47]. In fact, there are many constructions that achieve different trade-offs between proof size, proof time, and verification time, but also under different trust models as well as cryptographic assumptions. Although the pairing based zk-SNARKs [51, 71, 21, 56] are the most commonly adopted ZKP protocols in practice, which have been deployed in real systems such as the ZCash [20, 11] cryptocurrency, they require a trusted setup to generate the system parameters called *structured reference string (SRS)*, and the security is broken when the secret information is leaked to an attacker.

To address this problem, many ZKP protocols based on different primitives have been proposed to remove the trusted setup [32, 37, 19, 22, 33, 79, 84, 60]. A proof system is called *transparent* if it does not require any trusted setup. Bünz et al. [32] and Wahby et al. [83] constructed a transparent ZKP based on the discrete logarithm assumption. Recently, Bünz

et al. [33] also constructed a transparent ZKP based on ideal class groups with the adaptive root assumption, and Zhang et al. [84] and Kattis et al. [60] independently proposed transparent schemes based on the Fast Reed Solomon codewords [18, 24].

On the other hand, another research thread [21, 23, 66, 48, 36] has produced proof systems that remove trust from the circuit preprocessing phase called *universal* setup. The word universal means that SRS supports any circuit up to a given size bound. Because the circuit preprocessing algorithm, called the indexing, is public and deterministic, anyone can check the validity of the instance. However, these schemes require a trusted setup to generate SRS.

In this paper, we present a generic construction method of transparent *polynomial commitment schemes*, which have a small proof size, from *vector commitment schemes*. Our method can apply to Pedersen (resp. Ajtai) commitments, which is based on the discrete logarithm problem (resp. the lattice problem called the *Short Integer Solution: SIS*). Some works mentioned above showed that if we can construct a transparent polynomial commitment scheme, then universal and transparent ZKPs can be constructed based on it. Consequently, we obtain universal and transparent ZKPs from the discrete logarithm and lattice assumption.

3.1.1 Our contributions

Following the recent works [66, 36, 48, 33], universal ZKPs can be built from polynomial commitment schemes, and whenever the underlying polynomial commitment scheme is transparent, ZKP is also transparent. The main technical contributions of our work are proposing a generic construction of transparent polynomial commitment schemes from transparent vector commitments and applying it to Pedersen and Ajtai commitments. Our concrete contributions are:

Polynomial commitments from vector commitments

We propose a generic construction method of polynomial commitment scheme from vector commitment. More precisely, our theorem is as follow:
Main theorem. (informal) If a transparent vector commitment scheme satisfies *additive*, *key-concatenative*, and *key-convolution* property, furthermore, we assume it has an extractable zk-SNARK protocol that proves the committed vector is the $\mathbf{0}$-vector; then, we can construct a transparent polynomial commitment scheme based on it.

New universal ZKPs without trusted setup

We can apply our method to Pedersen and Ajtai vector commitment schemes, and thus, we obtain universal and transparent ZKPs. In this paper,

we show examples of applying our schemes to Marlin protocol [36].

Our proposed schemes are based on the standard cryptographic assumptions: the discrete logarithm or lattice assumptions. To the best of our knowledge, the second proposed scheme is the first lattice-based universal and transparent ZKP protocol. Furthermore, both of our proposals achieve the smallest proof size $O(\log C)$ and prover time $O(C)$ among known transparent and universal ZKP protocols for a circuit with C gates (see Table 3.1). On the other hand, our schemes require $O(C)$ verification time that is bigger than the others, which gives an interesting trade-off.

Although the verification time is linear, our verification algorithm is suitable for the batch verification of multiple proofs. In fact, the batch-verification time for verifying 100 proofs is about twice as long as one proof. Therefore, the time of verifying is significantly reduced in practice. This property is essential for use in blockchain and cryptocurrencies, such as ZCash [11], because the verifier verifies many zero-knowledge proofs within transactions.

Implementation and R1CS adapter for the libsnark

We implemented our two polynomial commitment schemes based on the Pedersen and Ajtai commitments. Our experiments show that the Pedersen (resp. Ajtai) based scheme takes 115.4 (resp. 311.7) seconds to generate a proof for degree $d = 2^{20}$ (resp. 2^{15}) polynomials and 13.3 (resp. 65.5) seconds to verify it. Also, the proof size is 1.4 KB (resp. 26.4 MB). Furthermore, the time of batch-verifying 100 proofs is 27.9 (resp. 137.6) seconds; therefore, each proof takes 279 msec (resp. 1376 msec).

Also, we implemented a universal and transparent Marlin type protocol [36] using our polynomial commitment schemes in C++. We generated a zero-knowledge proof of SHA-256 circuit by this scheme. Then, the proof size is 26.3 KB, and this size is less than half of Hyrax's proof for SHA-256; it is about 60 KB (see Section 8, [83]). Thus, to the best of our knowledge, our proposal is a universal and transparent ZKP with the shortest proof size.

Furthermore, we have developed an adapter to use the libsnark library [4], which is a standard zero-knowledge library in the zk-SNARK or blockchain area. This adapter converts the libsnark's R1CS and gadgets to our ZKPs. We believe this software will be useful to ZKP-developers and plan to release an open source our system as soon.

3.1.2 Related works

Transparent polynomial commitment scheme.
Wahby et al. [83] constructed a transparent polynomial commitment scheme and ZKP, called Hyrax, for multi-linear polynomial using the matrix com-

Table 3.1: Transparent ZKPs from polynomial commitments. C is the size of the circuit with depth D, and n is witness size.

	Proof size	Prover	Verifier	PQ	Primitive/Assumptions
Proposal 1	$O(\log C)$	$O(C)$	$O(C)$	-	Discrete Log
Proposal 2	$O(\log C)$	$O(C)$	$O(C)$	✓	Lattice/SIS
Hyrax [83]	$O(\sqrt{n} + D \log C)$	$O(C \log C)$	$O(\sqrt{n} + D \log C)$	-	Discrete Log
Super Sonic [33]	$O(\log C)$	$O(C \log C)$	$O(\log C)$	-	Class gp/Adaptive Root
Virgo [84]	$O(\log^2 n + D \log C)$	$O(C + n \log n)$	$O(\log^2 n + D \log C)$	✓	Reed Solomon Code

mitment scheme proposed by Bootle et al. [30] and efficient inner product argument technique by Bünz et al. [32]. The proof and commitment size of this scheme is about $O(\sqrt{d})$ for degree d polynomials. More precisely, The proof size and verifier time of Hyrax [83] are $O(\sqrt{n} + D \log C)$, and the prover time is $O(C \log C)$, where C is the size of the circuit with depth D and n is witness size. This scheme is based on the standard discrete logarithm assumption. Recently, Bünz et al. [33] have constructed a transparent commitment scheme over ideal class groups of imaginary quadratic fields under the *Adaptive Root Assumption*. This method is efficient because the proof size and verification time are both $O(\log C)$. However, their security assumptions are not standard, so further discussion is needed. More recently, Zhang et al. [84] and Kattis et al. [60] independently proposed transparent commitment schemes based on the Reed Solomon codewords [18, 24]. The proof size and verification time are $O(\log^2 n + D \log C)$, and the prover time is $O(C + n \log n)$.

3.1.3 Our techniques

We describe the intuitive construction of our polynomial commitment scheme for the case of Pedersen commitments. Let $f(X) = a_1 + a_2 X \cdots + a_n X^{n-1}$ be a polynomial over $\mathbf{F}_p$, and $\mathbf{a} = (a_1, \ldots, a_n)^T$ be the coefficient vector. At first, the prover commits $Com(f; r) := h^r(\prod_{i=1}^n g_i^{a_i})$ to the verifier, and the verifier sends a random point $q \in \mathbf{F}_p$. Next, the prover replies $f(q) = \langle \mathbf{a}, \mathbf{q} \rangle$, where $\mathbf{q} = (1, q, \ldots, q^{n-1})^T$, and wants to prove that $f(q)$ is the evaluated value of $Com(f)$ at q. If the prover sends $\pi = (\mathbf{a}, r)$, then the verifier can check it easy, but the proof size is $O(n)$, and all coefficients information is revealed. Therefore, the prover sends other two commitments C_1, C_2 and the verifier replies a random element $s \in \mathbf{F}_p$. Using C_1, C_2, and s, they update $Com(f; r)$ to $Com(f'; r')$, where $\deg f' = (\deg f)/2$. Also, the prover updates $\mathbf{a}$ to $\mathbf{a}'$, where $\mathbf{a}'$ has the half dimension $n/2$ and corresponds to f'. The prover and verifier recursively execute this procedure $\log f$ times, $Com(f; r)$ becomes 1-dimensional commitment $Com(\tilde{f}; \tilde{r}) = h^{\tilde{r}} \tilde{g}^{\tilde{a}}$ (this recursive technique is inspired by similar protocols from [30, 32, 83]). The prover eventually sends $\tilde{a}$ and a zero-knowledge proof of the discrete log-

arithm $\tilde{r}$ as a proof for the original statement. Note that our polynomial commitment scheme reveals $f(q) = \langle \mathbf{a}, \mathbf{q} \rangle$ and $\tilde{a}$ (the ordinary polynomial commitment scheme such as [59] reveals only $f(q)$). Therefore, we must hide this information in the zero-knowledge protocol layer (see Section 3.5.1).

3.2 Preliminary

Notation. Notation. In this paper, we denote the set of real numbers by $\mathbf{R}$, the set of integers by $\mathbf{Z}$, the set of natural numbers by $\mathbf{N}$, the set of integers modulo q by $\mathbf{Z}_q$, and the set of finite field of order p by $\mathbf{F}_p$. We also denote $\{0, 1, ..., N\} \subset \mathbf{Z}$ by $[0, N]$. Let $\lambda \in \mathbf{N}$ be the computational security parameter. A function $g(x)$ on $\mathbf{R}$ is negligible in λ if $g(\lambda) = o(\lambda^{-c})$ for every fixed constant $c \in \mathbf{R}_{>0}$, and we denote it by $\mathrm{negl}(\lambda)$. In the security proofs of this paper, we assume that all the adversaries are probabilistic Turing machines that run in time $\mathrm{poly}(\lambda)$ (PPT), and we denote an adversary by $\mathcal{A}$. Furthermore, we will drop the security parameter λ from the notation when it is contextually clear. In the interactive proof systems in this paper, we assume all verifiers are honest. In other words, all verifiers follow the protocol honestly.

3.2.1 Zero-knowledge proof: ZKP

Here, we define a zero-knowledge proof protocol. Let $\mathcal{R}$ be an **NP**-relation, and $(u, w) \in \mathcal{R}$. In this paper, we refer to an element u as statement (or public part) and w as the witness (or secret part) of $\mathcal{R}$. In addition, we denote the **NP**-relation defined using an arithmetic circuit C by $\mathcal{R}_C$ and its language by $\mathcal{L}_C$. The statement u corresponds to the public wire's value of C, and the witness corresponds to the value of the secret wire.
Definition. (Public verifier non-interactive proof system)
Let $\mathcal{R}$ be an **NP**-relation. A public verifier non-interactive proof system for $\mathcal{R}$ is a tuple of three polynomial-time algorithms, namely, $\Pi = (\mathbf{G}, \mathbf{P}, \mathbf{V})$, as follows:

- $(pp) \leftarrow \mathbf{G}(1^\lambda, \mathcal{R})$ takes security parameter λ and the **NP**-relation $\mathcal{R}$, and then outputs a public parameter pp.
- $\pi \leftarrow \mathbf{P}(pp, u, w)$ takes pp and statement u, witness w pair for $\mathcal{R}$, and outputs the proof of knowledge π.
- $\mathbf{bool} \leftarrow \mathbf{V}(pp, u, \pi)$ takes pp, statement u, and proof of knowledge π, and then outputs **true** if the proof π was accepted, otherwise outputs **false**. In this paper, we assume that the verification algorithm $\mathbf{V}$ is a deterministic polynomial-time one.

Definition. (Completeness)
A public verifier non-interactive proof system Π=$(\mathbf{G}, \mathbf{P}, \mathbf{V})$ has completeness if the prover has a valid pair (u, w) for $\mathcal{R}$. Accordingly, $pp \leftarrow \mathbf{G}(1^\lambda, \mathcal{R})$,

$$\Pr[\ \mathbf{V}(pp, u, \pi) = \mathbf{true} \mid \pi \leftarrow \mathbf{P}(p, u, w)] = 1$$

Definition. (Extractable Knowledge Soundness)
A designated verifier non-interactive proof system Π=$(\mathbf{G}, \mathbf{P}, \mathbf{V})$ has extractable knowledge-soundness if for any PPT adversary $\mathcal{A}$ there exits a PPT extractor $\mathcal{E}_\mathcal{A}$ such that

$$\mathrm{Adv}^{ksnd}_{\mathcal{A},\mathcal{E}xt_\mathcal{A}} := \Pr[(u, w) \notin \mathcal{L} \wedge \mathbf{V}(pp, u, \pi) = 1] = \mathrm{negl}(\lambda),$$

where $\mathcal{A}(pp) \rightarrow (u, \pi)$ and $\mathcal{E}xt_\mathcal{A}(pp, u, \pi) \rightarrow w$.
Definition. (Statistical Zero-knowledge)
A public verifier non-interactive proof system Π=$(\mathbf{G}, \mathbf{P}, \mathbf{V})$ has statistical zero-knowledge if there exist PPT simulators $\mathcal{S}_1$ and $\mathcal{S}_2$ such that for any PPT distinguisher $\mathcal{D}$, the following two probabilistic distributions are statistically difficult to distinguish:

$$\Pr\Big[\mathcal{D}(\pi) = 1 | \mathbf{G}(1^\lambda, \mathcal{R}) \rightarrow (CRS, VRS),$$

$$\mathcal{D}(CRS) \rightarrow (u, w), \mathbf{P}(CRS, u, w) \rightarrow \pi\Big] \approx$$

$$\Pr\Big[\mathcal{D}(\pi) = 1 | \mathcal{S}_1(1^\lambda, \mathcal{R}) \rightarrow (CRS, VRS, trap),$$

$$\mathcal{D}(CRS) \rightarrow (u, w), \mathcal{S}_2(CRS, u, trap) \rightarrow \pi\Big].$$

Definition. (Succinctness)
A public verifier non-interactive proof system Π=$(\mathbf{G}, \mathbf{P}, \mathbf{V})$ is succinct if the proof size of π is $O(\lambda)$ and the processing time of the verification algorithm $\mathbf{V}$ is $O(|u|)$, where $|u|$ is the size of statement.
Definition. (Succinct non-interactive zero-knowledge proof: zk-SNARK)
If a public verifier non-interactive proof system has completeness, extractable knowledge-soundness, and succinctness, then we refer to it as the public verifier succinct non-interactive zero-knowledge proof or zk-SNARK.

3.2.2 Polynomial commitments scheme

In this section, we define the polynomial commitment scheme by Kate et al. [59] and Chiesa et al. [36]. In addition, we now extend the syntax to polynomial commitment schemes. The following definition generalizes that of Kate et al. [59] and Chiesa et al. [36] to evaluate the inner product with

the coefficient vector $\mathbf{a} = (a_1, \ldots, a_n)$ of $f(X) = a_1 + a_2X + \cdots + a_nX^{n-1}$ can be evaluated.

More precisely, in the polynomial commitment scheme, the prover first commits the polynomial $f(X) = a_1 + a_2X + \cdots + a_nX^{n-1}$ and receives a random point $q \in \mathbf{F}_p$ from the verifier. After that, the prover sends the evaluation value $f(q)$ at the point q and the proof π indicating that to the verifier. At this time, if the vector $\mathbf{q}$ is defined as $\mathbf{q} := (1, q, ..., q^{n-1})$, it can be expressed as $f(q) = \langle \mathbf{a}, \mathbf{q} \rangle$ in the form of the inner product. Consequently, in our definition, the evaluation value is expressed in the form of the inner product with the coefficient vector. In fact, in the polynomial commitment scheme constructed in the next section, the prover sends **two inner product values** $\langle \mathbf{a}, \mathbf{q} \rangle$ and $\langle \mathbf{a}, \mathbf{v} \rangle$ to prove the evaluation value $f(q)$ at the point q, where $\mathbf{v}$ is a random vector generated by Fiat-Shamir heuristic [46, 35].

Definition 11. (Polynomial commitment scheme)
Let $\mathbf{F}_p$ be a finite field of order p. A polynomial commitment scheme over $\mathbf{F}_p$ is a tuple of polynomial time algorithms

$$\mathcal{PC} = (Setup, Trim, Commit, Open, Verify)$$

with the following syntax.

- $Setup(1^\lambda) \to pp$. On input a security parameter λ, $Setup$ outputs a public parameter pp.
- $Trim(pp, \mathbf{d}, M) \to ck$. Given oracle access to public parameters pp, and on input a security parameter λ, polynomial degree bounds $\mathbf{d} = \{d_i\}_{i=1}^n$, and a maximum degree $M \in \mathbf{N}$ ($d_i \leq M$), $Trim$ outputs a commitment key ck.
- $Commit(ck, \mathbf{f}, \mathbf{d}; \mathbf{r}) \to \mathbf{c}$. On input ck, univariate polynomials $\mathbf{f} = \{f_i(X)\}_{i=1}^n$ over $\mathbf{F}_p$, and degree bounds $\mathbf{d} = \{d_i\}_{i=1}^n$ with $\deg(f_i) \leq d_i \leq M$, $Commit$ outputs commitments $\mathbf{c} = \{c_i\}_{i=1}^n$ to the polynomials $\mathbf{f} = \{f_i(X)\}_{i=1}^n$. The randomness $\mathbf{r} = \{r_i\}_{i=1}^n$ are used if the commitments $\mathbf{c} = \{c_i\}_{i=1}^n$ are hiding.
- $Open(ck, \mathbf{f}, \mathbf{d}, Q, \mathbf{r}) \to \pi$. On input ck, univariate polynomials $\mathbf{f} = \{f_i(X)\}_{i=1}^n$, degree bounds $\mathbf{d} = \{d_i\}_{i=1}^n$, and a query set $\{Q_i\}_{i=1}^n \subset \mathbf{F}_p^{d_i}$, $Open$ outputs a proof π. The randomness $\mathbf{r}$ is equal the one previously used in $Commit$ algorithm.
- $Verify(pp, ck, \mathbf{c}, \mathbf{d}, Q, \mathbf{v}, \pi) \to \{\mathbf{True}, \mathbf{False}\}$. On input the public parameter pp, the commitment key ck, a commitment $\mathbf{c} = \{c_i\}_{i=1}^n$, the degree bounds $\mathbf{d} = \{d_i\}_{i=1}^n$, query sets $\{Q_i\}_{i=1}^n = \{\mathbf{q}_1^{(i)}, \ldots, \mathbf{q}_{m(i)}^{(i)}\}$,

alleged evaluations $\mathbf{v}_i = (v_1^{(i)}, \ldots, v_{m(i)}^{(i)})$, and a proof π, $Verify$ outputs **True** if and only if the proof π attests that, for every $\mathbf{q}_j^{(i)} \in Q_i$, the polynomials $f_i(X)$ committed in c_i has degree at most d_i and $v_j^{(i)} = \langle \mathbf{a}_i, \mathbf{q}_j^{(i)} \rangle$ for all i, j, where $\mathbf{a}_i$ is the coefficient vector of $f_i(X)$.

A polynomial commitment scheme $\mathcal{PC}$ must satisfy the completeness, extractability, and hiding properties defined below.

Definition 12. (Completeness)
For every maximum degree $M \in \mathbf{N}$, polynomial degree bounds $\mathbf{d}$, and PPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{l|c} & Setup(1^\lambda) \to pp \\ & \mathcal{A}(ck) \to (\mathbf{f}, \mathbf{d}, Q, \mathbf{r}) \\ & Trim(pp, \mathbf{d}, M) \to ck \\ Verify(pp, ck, \mathbf{c}, \mathbf{d}, Q, \mathbf{v}, \pi) = \textbf{True} & \deg(f_i(X)) \le d_i \le M \\ & Commit(ck, \mathbf{f}, \mathbf{d}; \mathbf{r}) \to \mathbf{c} \\ & \langle \mathbf{a}_i, Q_i \rangle \to \mathbf{v}_i \\ & Open(ck, \mathbf{f}, \mathbf{d}, Q, \mathbf{r}) \to \pi \end{array}\right] = 1,$$

where $\langle \mathbf{a}_i, Q_i \rangle = \{\langle \mathbf{a}_i, \mathbf{q}^{(i)} \rangle\}_{\mathbf{q}^{(i)} \in Q_i}$. Roughly speaking, the completeness means that everyone who has the correct knowledge for the polynomials $\mathbf{f}$ and randomness $\mathbf{r}$ can pass the verification procedure.

Definition 13. (Extractability)
For every maximum degree $M \in \mathbf{N}$, polynomial degree bounds $\mathbf{d}$, and PPT adversary $\mathcal{A}$, and its sub-attacker $\mathcal{A}'$ there exits a PPT extractor $\mathcal{E}_\mathcal{A}$ such that for every public-coin challenger $\mathcal{C}$ and its query sampler $\mathcal{Q}$ the probability below is negligibly close to 1 as a function of λ:

$$\Pr\left[\begin{array}{l|c} & Setup(1^\lambda) \to pp \\ & \mathcal{C}(pp) \to \text{state} \\ Verify(pp, ck, \mathbf{c}, \mathbf{d}, Q, \mathbf{v}, \pi) = \textbf{True} & Trim(pp, \mathbf{d}, M) \to ck \\ \Downarrow & \mathcal{A}(pp, \text{state}) \to \mathbf{c} \\ \deg(f_i) \le d_i \le M & \mathcal{Q}(ck, \text{state}) \to Q \\ Commit(ck, \mathbf{f}, \mathbf{d}; \mathbf{r}) \to \mathbf{c} & \mathcal{A}'(\mathbf{c}, \mathbf{d}, Q, \text{state}) \to (\mathbf{v}, \pi) \\ & \mathcal{E}_\mathcal{A}(ck, \mathbf{c}, \mathbf{d}, Q, \mathbf{v}, \pi) \to (\mathbf{f}, \mathbf{r}) \end{array}\right].$$

Roughly speaking, if there exists an attacker who can pass the verification procedure, then it can extract a valid polynomial set for the commitment $\mathbf{c}$.

Definition 14. (Computational hiding)
There exists a polynomial time simulator with a trapdoor $\mathcal{S} = (Setup, Commit, Open)$ such that for every maximum degree M and a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, the probabilities that $b = 1$ the two experiments in Table 3.2 are identical.

Table 3.2: Hiding game

$Real(1^\lambda, M, \mathcal{A})$
1: $pp \leftarrow Setup(\lambda)$.
2: $(\mathbf{f}, \mathbf{d}) \leftarrow \mathcal{A}_1(pp)$.
3: $ck \leftarrow Trim(pp, \mathbf{d}, M)$.
4: Sample randomness $\mathbf{r}$, and $\mathbf{c} \leftarrow Commit(ck, \mathbf{f}, \mathbf{d}; \mathbf{r})$.
5: $(Q, state) \leftarrow \mathcal{A}_2(pp.\mathbf{c})$.
6: $\pi \leftarrow Open(ck, \mathbf{f}, \mathbf{d}, Q; \mathbf{r})$.
7: $b \leftarrow \mathcal{A}_3(state, \pi)$.
$Ideal(1^\lambda, M, \mathcal{A})$
1: $(pp, trap) \leftarrow \mathcal{S}.Setup(\lambda, M)$.
2: $(\mathbf{f}, \mathbf{d}) \leftarrow \mathcal{A}_1(pp)$.
3: $ck \leftarrow Trim(pp, \mathbf{d}, M)$.
4: Sample simulated randomness $\mathbf{r}$, and $\mathbf{c} \leftarrow \mathcal{S}.Commit(trap, \mathbf{d}; \mathbf{r})$.
5: $(Q, state) \leftarrow \mathcal{A}_2(pp.\mathbf{c})$.
6: $\pi \leftarrow \mathcal{S}.Open(trap, (\langle \mathbf{a}_i, Q_i \rangle)_i, \mathbf{d}, Q; \mathbf{r})$.
7: $b \leftarrow \mathcal{A}_3(state, \pi)$.

Roughly speaking, the hiding property means that using trapdoor, the simulator who does not have the polynomials $\mathbf{f}$ can generate a valid proof π. In other words, A PPT attacker cannot get any information about polynomials $\mathbf{f}$ from the proof π except the inner product values $\{\langle \mathbf{a}_i, Q_i \rangle\}_{i=1}^n$.

3.2.3 Fiat-Shamir heuristic

All verifiers in this paper are *public-coin* verifier, as all the honest verifier's messages are random elements from $\mathbf{Z}_p^*$. The Fiat-Shamir transform [46, 35] compiles public-coin interactive proofs that have an honest verifier into non-interactive proofs. This transformation is secure under the random oracle model. In practice, all random challenges are replaced by hashes of the transcript up to that point. For example, the prover generates a commitment Com, and the next random challenge is $s := Hash(Com, pp)$, where pp is the public parameters. When the verifier generates multiple random challenge, in the interactive proof protocol, we call the Fiat-Shamir converted random number sequence *Fiat-Shamir sequence*.

3.2.4 Discrete log relation

Here, we define the Discrete log relation assumption.
Definition. (Discrete log relation)
Let p be a prime and $\mathbf{G}$ be a discrete group of order p. For all PPT adversaries $\mathcal{A}$ and for all $n \geq 2$ the following probability is negligible as a function

of λ:

$$\Pr\left[\begin{array}{c|c} \mathbf{a} \neq \mathbf{0} \wedge & \mathbf{G} \leftarrow Setup(1^\lambda). \\ \prod_{i=1}^{n} g_i^{a_i} = 1 & g_1, g_2, \ldots, g_n \xleftarrow{\$} \mathbf{G}. \\ & \mathbf{a} = (a_1, \ldots, a_n) \in \mathbf{Z}_p^n \leftarrow \mathcal{A}(\mathbf{G}, g_1, \ldots, g_n). \end{array}\right] \approx 0.$$

Note that if $n = 2$, this is the ordinary discrete log assumption because $g_1^{a_1} g_2^{a_2} = 1 \Leftrightarrow g_1^{a_1/a_2} = g_2$, where $a_1/a_2 \in \mathbf{Z}_p$.

3.2.5 Short integer solution

We define the short integer solution assumption according to [12, 13].
Definition. (Short Integer Solution: SIS)
Let $q \in \mathbf{Z}$ be an integer and $B(q)$ be a bound parameter. For all PPT adversaries $\mathcal{A}$ the following probability is negligible as a function of λ:

$$\Pr\left[\begin{array}{c|c} \mathbf{x} \neq \mathbf{0} \wedge & \\ \mathbf{A}\mathbf{x} = \mathbf{0} \wedge & Setup(1^\lambda, B(q)) \rightarrow \mathbf{A} \in \mathbf{Z}_q^{n \times m}. \\ ||\mathbf{x}||_2 \leq B(q) & \mathcal{A}(\mathbf{A}) \rightarrow \mathbf{x} \in \mathbf{Z}^m. \end{array}\right] \approx 0.$$

3.2.6 A General Forking Lemma

We refer to the forking lemma of [30, 32] that is used in the proof of theorem 10. Suppose that we have a $(2\mu + 1)$-move public-coin argument with μ challenge, $x_1, \ldots, x_\mu$. Let $n_i \geq 1,\ i = 1, \ldots, \mu$. Consider $\prod_{i=1}^{\mu} n_i$ accepting transcripts with challenges in the following tree format. The tree has depth μ and $\prod_{i=1}^{\mu} n_i$ leaves. The root of the tree is labeled with the statement. Each node of depth $i < \mu$ has exactly n_i children, each labeled with a distinct value of the ith challenge x_i.

Theorem 9. (General forking lemma)
Let $\mathcal{P}, \mathcal{V}$ be a $(2\mu + 1)$-move public-coin interactive protocol. Let χ be be a witness extraction algorithm that succeeds with probability $1 - \mu(\lambda)$ for some negligible function $\mu(\lambda)$ in extracting a witness from an $(n_1, \ldots, n_\mu)$-tree of accepting transcripts in probabilistic polynomial time. Assume that $\prod_{i=1}^{\mu} n_i$ is bounded above by a polynomial in the security parameter λ. Then $\mathcal{P}, \mathcal{V}$ has witness-extend emulation.

In our situation (in the proof of theorem 10), all $n_i = 4$ and $\mu = l (= \log(\deg f(x)))$, thus,

$$\prod_{i=1}^{\mu} n_i = 2^{2l} = (\deg f(x))^2.$$

This is bounded by a polynomial in the security parameter λ.

3.3 Vector commitment schemes

In this section, we define our vector commitment scheme and some properties that must be satisfied. Our examples, Pedersen and Ajtai vector commitments, satisfy these properties.

3.3.1 Vector commitment scheme

Definition 15. (Vector commitment scheme)
Let $\mathbf{F}_p$ be a finite field order p. A vector commitment scheme over $\mathbf{F}_p$ is a tuple of polynomial time algorithms

$$\mathcal{VC} = (VC.Setup, VC.KeyGen, VC.Commit)$$

with the following syntax.

- $VC.Setup(1^\lambda) \to pp$. On input a security parameter λ, $Setup$ outputs a public parameter pp.
- $VC.KeyGen(pp, n, t) \to k$. On input the public parameter pp, a dimension of vectors n, and a dimension of randomness vector part t, $KeyGen$ outputs a commitment key k for the n-dimensional vectors and the t-dimensional randomness vectors. In this paper, we assume the commitment key space is an abelian group and denote the scalar multiplication $\alpha \in \mathbf{Z}$ of k by αk.
- $VC.Commit(pp, k, \mathbf{a}; \mathbf{r}) \to Com_k(\mathbf{a}; \mathbf{r})$. On input the public parameter pp, the commitment key k, a vector $\mathbf{a} \in \mathbf{F}_p^n$, and a random vector $\mathbf{r} \in \mathbf{F}_p^t$, $Commit$ outputs a commitment value $Com_k(\mathbf{a}; \mathbf{r})$. The randomness $\mathbf{r} \in \mathbf{F}_p^t$ is used if the commitment Com is hiding.

Definition 16. (Computational hiding for vector commitments)
We say that a vector commitment scheme $\mathcal{VC}$ is computational hiding if for all PPT adversaries $\mathcal{A}$ and random-coin challengers $\mathcal{C}$, the probability below is negligibly close to $\frac{1}{2}$ as a function of λ:

$$\Pr\left[\, b = b' \;\middle|\; \begin{array}{c} VC.Setup(1^\lambda) \to pp. \\ VC.KeyGen(pp, n, t) \to k. \\ \mathcal{A}(pp, k) \to (\mathbf{a}_0, \mathbf{a}_1). \\ \mathcal{C} \text{ generates } b \in^\$ \{0,1\}, \mathbf{r} \in^\$ \mathbf{F}_p^t \\ \text{and } c = Com_k(\mathbf{a}_b; \mathbf{r}). \\ \mathcal{A}(c) \to b'. \end{array} \right] \approx \frac{1}{2}.$$

Roughly speaking, the hiding property means that a commitment does not reveal the committed value.

Definition 17. (Computational binding for vector commitments)
We say that a vector commitment scheme $\mathcal{VC}$ is computational binding if for all PPT adversaries $\mathcal{A}$ the probability below is negligibly close to 0 as a function of λ:

$$\Pr\left[\begin{array}{c|c} \mathbf{a}_0 \neq \mathbf{a}_1 \wedge & VC.Setup(1^\lambda) \rightarrow pp. \\ c_0 = c_1 & VC.KeyGen(pp, n, t) \rightarrow k. \\ & \mathcal{A}(pp, k) \rightarrow (\mathbf{a}_0, \mathbf{a}_1; \mathbf{r}_0, \mathbf{r}_1). \\ & c_0 = Com_k(\mathbf{a}_0; \mathbf{r}_0),\ c_1 = Com_k(\mathbf{a}_1; \mathbf{r}_1). \end{array}\right] \approx 0.$$

Roughly speaking, the biding property means that a commitment can only be opened to one value.

Furthermore, we assume that the vector commitment scheme has the following three operations and zero-knowledge property: (somewhat) additive homomorphic, (somewhat) key-concatenative, key-convolution, and zk-SNARK for the $\mathbf{0}$ vector.

Definition 18. (Additive homomorphic)
Let $\mathbf{a}_1,\ \mathbf{a}_2 \in \mathbf{F}_p$ be two vectors, and there exists + operation over the commitment value space such that

$$Com_k(\mathbf{a}_1; r_1) + Com_k(\mathbf{a}_2; r_2) = Com_k(\mathbf{a}_1 + \mathbf{a}_2; r_1 + r_2).$$

Also, in the case where the above operation + holds only a finite number of times, it is called somewhat additive homomorphic.

Furthermore, this operation is assumed to be compatible with scalar multiplication in the commit key-space. In other words, let $\alpha \in \mathbf{Z}$ be an integer, the following equation holds:

$$Com_{\alpha k}(\mathbf{a}; \mathbf{r}) = Com_k(\alpha\mathbf{a}; \alpha\mathbf{r}).$$

Definition 19. (Key-concatenative and Key-convolution)
Let k, k' be two commitment keys with the same public parameter pp, and there exist $||$ operations on the commitment value space and the commitment keys k, k' such that

$$Com_k(\mathbf{a}; r)||Com_{k'}(\mathbf{b}; r') = Com_{k||k'}Com(\mathbf{a}||\mathbf{b}; r + r'),$$

where $\mathbf{a}||\mathbf{b} \in \mathbf{F}_p^{2n}$ is the concatenate of two vectors. Also, in the case where the above operation $||$ holds only a finite number of times, it is called somewhat key-concatenative.

Furthermore, when two commitment keys k, k' have the same pp and randomness part for $\mathbf{r}$, there exists $*$ convolution operation between k and k' such that

$$Com_k(\mathbf{a}; \mathbf{r})||Com_{k'}(\mathbf{a}; \mathbf{r}') = Com_{k*k'}(\mathbf{a}; \mathbf{r} + \mathbf{r}'),$$

for the same committed vector $\mathbf{a}$.

Definition 20. (zk-SNARK for the **0** vector)
If there exists a *succinct non-interactive zero-knowledge proof: zk-SNARK* algorithm for the **0** vector with the extractability, then we say $\mathcal{VC}$ has the zk-SNARK for the **0** vector property. More precisely, there exists a PPT algorithm $ZKP^0_{ext}(pp, k, Com_k(\mathbf{0}; \mathbf{r}), \mathbf{r}) \to \pi^0$. On input the public parameter pp, the commitment key k, a commitment of the **0** vector $Com_k(\mathbf{0}; \mathbf{r})$, and its randomness $\mathbf{r}$, ZKP^0_{ext} outputs a succinct non-interactive zeroknowledge proof π^0 such that π^0 convinces the verifier that the committed vector of $Com_k(\mathbf{0}; \mathbf{r})$ is $\mathbf{0} = (0, 0, \ldots, 0) \in \mathbf{F}_p^n$ vector. Furthermore, we assume this zero-knowledge proof system satisfies the extractability. In other words, a PPT attacker $\mathcal{A}$ who can generate a valid proof $(Com_k(\mathbf{0}; \mathbf{r}), \pi^0)$, then there exists an extractor $\mathcal{E}_A$ who can extract a randomness $\mathbf{r}'$ such that $Com_k(\mathbf{0}; \mathbf{r}) = Com_k(\mathbf{0}; \mathbf{r}')$. Note that we relaxed the definition of the extractability from the ordinary one. In the case of Pedersen commitments, we can extract the same randomness $\mathbf{r}$; however, Ajtai commitments, we can only extract a vector $\mathbf{r}'$ such that $Com_k(\mathbf{0}; \mathbf{r}) = Com_k(\mathbf{0}; \mathbf{r}')$ and $||\mathbf{r}'|| \leq 2B$, where B is the original error bound (see [16]). In the case of this weak extractability, our soundness proof holds so that we employ this definition.

Here, we present two examples of vector commitments that satisfy the properties mentioned above.

3.3.2 Example 1: Pedersen vector commitment

Let $\mathbf{G}$ be a discrete group order p with the discrete log relation assumption (see definition 3.2.4).

- $VC.Setup(1^\lambda) \to pp$. *Setup* outputs a public parameter which includes a seed of PRNG on $\mathbf{G}$ for the key generation algorithm.
- $VC.KeyGen(pp, n, t) \to \mathbf{g} = (g_1, g_2, \ldots, g_n) \in \mathbf{G}^n, \mathbf{h} = (h_1 \ldots, h_t) \in \mathbf{G}^t$. $KeyGen$ generates random elements $\mathbf{g} \in \mathbf{G}^n$ and $\mathbf{h} \in \mathbf{G}^t$ as a commitment key $k = (\mathbf{g}; \mathbf{h})$. We call $\mathbf{h}$ randomness part.
- $VC.Commit(pp, k, \mathbf{a}; \mathbf{r}) \to Com_k(\mathbf{a}; \mathbf{r}) = \Big(\prod_{i=1}^n g_i^{a_i}\Big)\Big(\prod_{i=1}^t h_i^{r_i}\Big)$, where $\mathbf{a} = (a_1, \ldots, a_n), \mathbf{r} = (r_1, \ldots, r_t)$. We simply denote it by $\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{r}}$.

Note that this commitment scheme does not require a trusted setup because the commitment key $k = (\mathbf{g}; \mathbf{h})$ is the output of a PRNG (or random oracle on $\mathbf{G}$). This commitment scheme has the hiding and binding properties (see [72, 42] for more details).

Let $\mathbf{a}_1 = (a_1^{(1)}, a_2^{(1)}, \ldots, a_n^{(1)})^T, \mathbf{a}_2 = (a_1^{(2)}, a_2^{(2)}, \ldots, a_n^{(2)})^T$ be two vectors, we define

$$Com_k(\mathbf{a}_1; \mathbf{r}_1) + Com_k(\mathbf{a}_2; \mathbf{r}_2) := \Big(\prod_{i=1}^n g_i^{a_i^{(1)}}\Big)\mathbf{h}^{\mathbf{r}_1}\Big(\prod_{i=1}^n g_i^{a_i^{(2)}}\Big)\mathbf{h}^{\mathbf{r}_2}$$

$$\Big(\prod_{i=1}^{n} g_i^{a_i^{(1)}+a_i^{(2)}}\Big)\mathbf{h}^{\mathbf{r}_1+\mathbf{r}_2} = Com_k(\mathbf{a}_1+\mathbf{a}_1;\mathbf{r}_1+\mathbf{r}_2).$$

Therefore, this scheme has the additive homomorphic property. Also, let $k = (g_1,\ldots,g_n;\mathbf{h}), k' = (g'_1,\ldots,g'_n;\mathbf{h})$ be two commitment keys that have the same randomness part $\mathbf{h}$. We define

$$k||k' := (g_1,\ldots,g_n,g'_1,\ldots,g'_n;\mathbf{h}), \quad \text{and,}$$

$$Com_k(\mathbf{a};\mathbf{r})||Com_{k'}(\mathbf{b};\mathbf{r}') := \Big(\prod_{i=1}^{n} g_i^{a_i}\Big)\mathbf{h}^{\mathbf{r}}\Big(\prod_{i=1}^{n}(g'_i)^{b_i}\Big)(\mathbf{h})^{\mathbf{r}'}$$

$$= \Big(\prod_{i=1}^{n} g_i^{a_i}\Big)\Big(\prod_{i=1}^{n}(g'_i)^{b_i}\Big)\mathbf{h}^{\mathbf{r}+\mathbf{r}'} = Com_{k||k'}(\mathbf{a}||\mathbf{b};\mathbf{r}+\mathbf{r'}).$$

Therefore, this scheme has the key-concatenative property. Furthermore, we define

$$k * k' := \big((g_1g'_1),(g_2g'_2),\ldots,(g_ng'_n);\mathbf{h}\big), \quad \textit{then,}$$

$$Com_k(\mathbf{a};\mathbf{r})||Com_{k'}(\mathbf{a};\mathbf{r}') = \Big(\prod_{i=1}^{n} g_i^{a_i}\Big)\Big(\prod_{i=1}^{n}(g'_i)^{a_i}\Big)\mathbf{h}^{\mathbf{r}+\mathbf{r}'}$$

$$= \Big(\prod_{i=1}^{n}(g_ig'_i)^{a_i}\Big)\mathbf{h}^{\mathbf{r}+\mathbf{r}'} = Com_{k*k'}(\mathbf{a};\mathbf{r}+\mathbf{r}').$$

Therefore, this scheme also has the key-convolution property. Let $Com_k(\mathbf{0};\mathbf{r}) = \mathbf{g^0h^r} = \mathbf{h^r}$ be a commitment of the $\mathbf{0}$ vector. Since the binding property, it is a zk-SNARK for the $\mathbf{0}$ vector that the ordinary Schnorr's non-interactive zero-knowledge proof for discrete log such that "I know the discrete logarithm $\mathbf{r}$" (we refer to [77, 78, 80, 42, 83] for the construction). Furthermore, the proof size of this protocol consists of $(t+1)$ $\mathbf{F}_p$ elements.

3.3.3 Example2: Ajtai commitment

Let q be an integer and $\mathbf{A}$ be a a fixed, randomly-chosen matrix in $\mathbf{Z}_q^{m\times n}$. Ajtai's seminal works [12, 13] showed that it is as hard to find a vector $\mathbf{s}\in\mathbf{Z}^n$ with some small bounded norm $||\mathbf{s}|| \le B$ such that $\mathbf{As} = 0$. This problem is called the Short Integer Solution (SIS) problem (see Section 3.2.5) and its hardness increases as m or q increase and the bound B decreases, however, the hardness of SIS is essentially unaffected by n as soon as n is large enough. The Ajtai commitment scheme is defined as follows. We assume $p << q$.

- $VC.Setup(1^\lambda) \to pp$. *Setup* outputs a public parameter which includes a seed of PRNG for the key generation algorithm.

- $VC.KeyGen(pp, n, t) \rightarrow k = (\mathbf{A}_1, \mathbf{A}_2)$. $KeyGen$ outputs uniformly-random matrices $\mathbf{A}_1 \in \mathbf{Z}_q^{m \times n}$ and $\mathbf{A}_2 \in \mathbf{Z}_q^{m \times t}$ as commitment key. For the security, in the rest of this paper, we set the randomness part $t = 2m \log_p q$.
- $VC.Commit(pp, k, \mathbf{a}; \mathbf{r}) \rightarrow Com_k(\mathbf{a}; \mathbf{r}) := \mathbf{A}_1\mathbf{a} + \mathbf{A}_2\mathbf{r} \in \mathbf{Z}_q^m$, where $\mathbf{a} \in \mathbf{F}_p^n$ and $\mathbf{r} \in \mathbf{F}_p^t$ have small norms.

Note that the Ajtai commitment scheme does not require a trusted setup because the commitment key is the output of a PRNG (or random oracle). This commitment scheme has the computational hiding and binding properties (see [13, 16]). We define

$$Com_k(\mathbf{a}_1; \mathbf{r}_1) + Com_k(\mathbf{a}_2; \mathbf{r}_2) := \mathbf{A}_1(\mathbf{a}_1 + \mathbf{a}_2) + \mathbf{A}_2(\mathbf{r}_1 + \mathbf{r}_2),$$

then this commitment scheme has the somewhat additive homomorphic property, while $\mathbf{a}_1 + \mathbf{a}_2$ and $\mathbf{r}_1 + \mathbf{r}_2$ are also small vectors. Also, let $k = (\mathbf{A}_1, \mathbf{A}_2), k' = (\mathbf{A}'_1, \mathbf{A}_2)$ be two commitment keys that have the same randomness part $\mathbf{A}_2$. We define the concatenate operation as the ordinary matrix concatenation:

$$k||k' := ([\mathbf{A}_1|\mathbf{A}'_1], \mathbf{A}_2),$$

where $\mathbf{A}_1$ and $\mathbf{A}'_1$ have the same matrix size. Since

$$Com_k(\mathbf{a}; \mathbf{r})||Com_{k'}(\mathbf{b}; \mathbf{r}') := \Big(\mathbf{A}_1\mathbf{a} + \mathbf{A}_2\mathbf{r}\Big) + \Big(\mathbf{A}'_1\mathbf{b} + \mathbf{A}_2\mathbf{r}'\Big)$$

$$= [\mathbf{A}_1|\mathbf{A}'_1] \left[\frac{\mathbf{a}}{\mathbf{b}}\right] + \mathbf{A}_2(\mathbf{r} + \mathbf{r}') = Com_{k||k'}(\mathbf{a}||\mathbf{b}; \mathbf{r} + \mathbf{r}') \in \mathbf{Z}_q^m,$$

this commitment scheme has somewhat key-concatenate property, while $\left[\frac{\mathbf{a}}{\mathbf{b}}\right]$ and $\mathbf{r} + \mathbf{r}'$ are also small vectors. Furthermore, we define $k * k' := \big(\mathbf{A}_1 + \mathbf{A}'_1, \mathbf{A}_2\big)$, then

$$Com_k(\mathbf{a}; \mathbf{r})||Com_{k'}(\mathbf{a}; \mathbf{r}') = \Big(\mathbf{A}_1\mathbf{a} + \mathbf{A}_2\mathbf{r}\Big) + \Big(\mathbf{A}'_1\mathbf{a} + \mathbf{A}_2\mathbf{r}'\Big)$$

$$= (\mathbf{A}_1 + \mathbf{A}_2)\mathbf{a} + \mathbf{A}_2(\mathbf{r} + \mathbf{r}') = Com_{k*k'}(\mathbf{a}; \mathbf{r} + \mathbf{r}').$$

Therefore, this commitment scheme also has the key-convolution property. We discuss more concrete parameter settings for this somewhat additive and key-concatenative properties in Section 3.6.

Furthermore, let $Com_k(\mathbf{0}; \mathbf{r}) = \mathbf{A}_2\mathbf{r}$ be a commitment of the $\mathbf{0}$ vector. We refer to [16] for constructing a succinct non-interactive zero-knowledge proof for $\mathbf{r}$ with the extractability, and the binding property shows that it is a zk-SNARK for $\mathbf{0}$ vector. Their extractor outputs another $\mathbf{r'}$ such that $Com_k(\mathbf{0}; \mathbf{r}) = Com_k(\mathbf{0}; \mathbf{r'}) \wedge ||\mathbf{r'}|| \leq 2B$. Even in this case, our soundness proof works, but it requires taking a large bound B in advance. In fact, we must take $(\deg f)B$ as the bound.

3.4 Polynomial commitments from vector commitments

In this section, we show a general method to construct a polynomial commitment scheme with $O(\log(\deg f))$ proof size from a vector commitment. At first, we construct a basic protocol that has $O(\deg f)$ proof size. This protocol reveals all information about $f(X)$ and we modify it to reduce the proof size and the leakage information using a convolution trick that inspired by similar protocols from [30, 32, 83].

As already shown in [36], a single-bound, single-query, and single-polynomial commitment scheme can be used as a black box to construct a multi-degree bound, multi-query, and multi-polynomial commitment scheme. Therefore, we construct here a single bound, query, and polynomial commitment scheme.

In this section, we assume a vector commitment scheme $\mathcal{VC} = (VC.Setup, VC.KeyGen, VC.Commit)$ satisfies additive homomorphic, key-concatenative, key-convolution, and zk-SNARK for the $\mathbf{0}$ vector properties, and we denote the proof generation algorithm of the zk-SNARK by ZKP^{0}_{ext}.

3.4.1 (In-secure and in-efficient) Basic protocol

The evaluation value of $f(X)$ at the point q can be expressed as the inner product form

$$f(q) = a_1 + a_2 q + a_3 q^2 \cdots + a_n q^{n-1} = \langle \mathbf{a}, \mathbf{q} \rangle$$

with the vector $\mathbf{q} := (1, q, q^2, \ldots, q^{n-1})^T$ and coefficient vector $\mathbf{a} = (a_1, a_2, \ldots, a_n)^T$ of the polynomial $f(X)$. By expressed this form, an efficient inner product argument can be applied.

- **Setup**.
  Execute the Setup algorithm of the vector commitment scheme $VC.Setup(1^\lambda) \to pp$ and output pp as the result of $Setup(1^\lambda)$.

- **Trim**.
  Execute the key generation algorithm of the vector commitment scheme $VC.KeyGen(pp, M, t) \to k$, and output k as the result of $Trim$.

- **Commit**.
  The prover calculates $VC.Commit(pp, k, \mathbf{a}; \mathbf{r}) \to Com_k(\mathbf{a}; \mathbf{r})$ for the coefficient vector $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ of $f(X) \in \mathbf{F}_p[x]$, and outputs $Com_k(\mathbf{a}; \mathbf{r})$ as the result of

  $$Commit(k, f(X), M; \mathbf{r}) \to Com_k(\mathbf{a}; \mathbf{r}).$$

The prover sends $c = Com_k(\mathbf{a}; \mathbf{r})$ to the verifier.

- **Open**.
  The prover sets $Open(k, f(X), M, \{q\}, \mathbf{r}) \rightarrow \pi := (\mathbf{a}, \mathbf{r}, \langle \mathbf{a}, \mathbf{q} \rangle)$ and sends it to the verifier.

- **Verify**.
  The verifier obtains $\pi = (\mathbf{a}, \mathbf{r}, u)$ from the prover and checks

$$VC.Commit(pp, k, \mathbf{a}; \mathbf{r}) = c,$$

  and $u = \langle \mathbf{a}, \mathbf{q} \rangle$. The verifier outputs **True** if all verifications are successful, and outputs **False** otherwise.

Discussion: Proof size and information leakage
The above protocol has a $O(n)$ proof size because the proof $\pi = (\mathbf{a}, \langle \mathbf{a}, \mathbf{q} \rangle, u)$ includes an n-dimensional vector $\mathbf{a} = (a_1, a_2, \ldots, a_n) \in \mathbf{F}_p^n$. In addition, the hiding property is not satisfied because all information about the coefficient vector $\mathbf{a}$ of the polynomial $f(X)$ is leaked. We are going to reduce the proof size and the coefficient information as follows.

3.4.2 Convolution-trick

Here, we present a convolution-trick that is inspired by [30, 32, 83]. We assume $n = 2n'$, ($n' \in \mathbf{N}$), and denote

$$\mathbf{a} = (a_1, a_2, \ldots, a_{n'}, a_{n'+1}, \ldots, a_n)^T = (\mathbf{a}^L, \mathbf{a}^R)^T,$$

$$\mathbf{q} = (1, q, q^2, \ldots, q^{n'-1}, q^{n'}, \ldots, q^{n-1})^T = (\mathbf{q}^L, \mathbf{q}^R)^T,$$

where $\mathbf{a}^L, \mathbf{a}^R, \mathbf{q}^L, \mathbf{q}^R \in \mathbf{F}_p^{n'}$. Note that $\mathbf{a} = \mathbf{a}^L || \mathbf{a}^R, \mathbf{q} = \mathbf{q}^L || \mathbf{q}^R$. Also, let $k = k^L || k^R$ be a commitment key, and $\tilde{k}$ be another commitment key for 1-dimensional vectors; we call this $\tilde{k}$-part the inner product part. More precisely, we define $VC.KeyGen(pp, 1, 0) \rightarrow \tilde{k}$ so that $\tilde{k}$-part does not have the randomness. Thus, in the rest of this paper, we omit the randomness part of $\tilde{k}$. We define

$$C := Com_k(\mathbf{a}; \mathbf{r}) \big|\big| Com_{\tilde{k}}(\langle \mathbf{a}, \mathbf{q} \rangle),$$

$$C_1 := Com_k(\mathbf{a}^R || \mathbf{a}^L; \mathbf{r}_1) \big|\big| Com_{\tilde{k}}(\langle \mathbf{a}^L, \mathbf{q}^R \rangle + \langle \mathbf{a}^R, \mathbf{q}^L \rangle),$$

$$C_2 := Com_k((\mathbf{0} || \mathbf{a}^R); \mathbf{r}_2) \big|\big| Com_{\tilde{k}}(\langle \mathbf{a}^R, \mathbf{q}^R \rangle).$$

Furthermore, the verifier generates a random element $s \in \mathbf{F}_p$. At this time, the following lemma holds.

Lemma 4.
We define

$$\mathbf{a}' := \mathbf{a}^L + s \times \mathbf{a}^R, \quad \mathbf{q}' := \mathbf{q}^L + s \times \mathbf{q}^R, \quad k' := k^L * (s \times k^R),$$

then the following equation holds:

$$C + s \times C_1 + (s^2 - 1) \times C_2 = Com_{k'}(\mathbf{a}'; \mathbf{r}') \big|\big| Com_{\tilde{k}}(\langle \mathbf{a}', \mathbf{q}' \rangle),$$

where $\mathbf{r}' = \mathbf{r} + s\mathbf{r}_1 + (s^2 - 1)\mathbf{r}_2$.

Proof. The left-hand side of the equation is equal to

$$Com_k\Big(\mathbf{a} + (s\mathbf{a}^R || s\mathbf{a}^L) + (s^2 - 1)\mathbf{0} || \mathbf{a}^R; \mathbf{r}'\Big)$$

$$|| Com_{\tilde{k}}\Big(\langle \mathbf{a}, \mathbf{q} \rangle + s\langle \mathbf{a}^L, \mathbf{q}^R \rangle + s\langle \mathbf{a}^R, \mathbf{q}^L \rangle + (s^2 - 1)\langle \mathbf{a}^R, \mathbf{q}^R \rangle\Big). \tag{3.1}$$

Since $\mathbf{a} = \mathbf{a}^L || \mathbf{a}^R$, and

$$\langle \mathbf{a}, \mathbf{q} \rangle = \langle \mathbf{a}^L, \mathbf{q}^L \rangle + \langle \mathbf{a}^R, \mathbf{q}^R \rangle,$$

we have

$$(3.1) = Com_k\Big((\mathbf{a}^L + s\mathbf{a}^R) || s(\mathbf{a}^L + s\mathbf{a}^R); \mathbf{r}'\Big)$$

$$|| Com_{\tilde{k}}\Big(\langle \mathbf{a}^L, \mathbf{q}^L \rangle + s\langle \mathbf{a}^L, \mathbf{q}^R \rangle + s\langle \mathbf{a}^R, \mathbf{q}^L \rangle + s^2\langle \mathbf{a}^R, \mathbf{q}^R \rangle\Big).$$

$$= Com_k\Big(\mathbf{a}' || s\mathbf{a}'; \mathbf{r}'\Big) || Com_{\tilde{k}}\Big(\langle \mathbf{a}', \mathbf{q}' \rangle\Big). \tag{3.2}$$

Since the definition of the key-convolution operation,

$$Com_k(\mathbf{a}' || s\mathbf{a}'; \mathbf{r}') = Com_{k^L}(\mathbf{a}'; \mathbf{r}') || Com_{sk^R}(\mathbf{a}'; \mathbf{0}) = Com_{k'}(\mathbf{a}'; \mathbf{r}'),$$

therefore, we have

$$(3.2) = Com_{k'}(\mathbf{a}'; \mathbf{r}') || Com_{\tilde{k}}(\langle \mathbf{a}', \mathbf{q}' \rangle).$$

From this lemma, we define

$$C' := Com_{k'}(\mathbf{a}'; \mathbf{r}') \big|\big| Com_{\tilde{k}}(\langle \mathbf{a}', \mathbf{q}' \rangle),$$

then the two vector $\mathbf{a}', \mathbf{q}'$ have the half dimension n' of $\mathbf{a}, \mathbf{q}$.

Consequently, the prover can reduce the dimension of the vector $\mathbf{a}$ by sending (C_1, C_2) to the verifier, and changes the proof π to $\pi' := (\mathbf{a}', \mathbf{r}', \langle \mathbf{a}', \mathbf{q}' \rangle, C_1, C_2)$.

3.4.3 Reducing the proof size and coefficient information

We can reduce the proof size to $\log n$ order by recursively using the convolution technique. We assume $n = 2^l$ for simplicity and define

$$\mathbf{a}(1) := \mathbf{a}, \quad \mathbf{q}(1) := \mathbf{q}, \quad \mathbf{r}(1) := \mathbf{r}, \quad k(1) := k,$$

$$C(1) := Com_k(\mathbf{a}(1); \mathbf{r}(1)) \big|\big| Com_{\tilde{k}}(\langle \mathbf{a}(1), \mathbf{q}(1) \rangle).$$

The prover takes new random vectors $\mathbf{r}_1(1), \mathbf{r}_2(1)$ and calculates

$$C_1(1) := Com_k(\mathbf{a}(1)^R || \mathbf{a}(1)^L; \mathbf{r}_1(1))$$
$$\big|\big| Com_{\tilde{k}}(\langle \mathbf{a}(1)^L, \mathbf{q}(1)^R \rangle + \langle \mathbf{a}(1)^R, \mathbf{q}(1)^L \rangle),$$
$$C_2(1) := Com_k(\mathbf{0} || \mathbf{a}(1)^R; \mathbf{r}_2(1)) \big|\big| Com_{\tilde{k}}(\langle \mathbf{a}(1)^R, \mathbf{q}(1)^R \rangle).$$

Furthermore, the prover recursively calculates

$$\mathbf{a}(i+1) := \mathbf{a}(i)^L + s_i \times \mathbf{a}(i)^R, \quad \mathbf{q}(i+1) := \mathbf{q}(i)^L + s_i \times \mathbf{q}(i)^R, \tag{3.3}$$
$$k(i+1) := k(i)^L * s_i \times k(i)^R,$$

and

$$C_1(i) := Com_k(\mathbf{a}(i)^R || \mathbf{a}(i)^L; \mathbf{r}_1(i))$$
$$\big|\big| Com_{\tilde{k}}(\langle \mathbf{a}(i)^L, \mathbf{q}(i)^R \rangle + \langle \mathbf{a}(i)^R, \mathbf{q}(i)^L \rangle), \tag{3.4}$$
$$C_2(i) := Com_k((\mathbf{0} || \mathbf{a}(i)^R); \mathbf{r}_2(i)) \big|\big| Com_{\tilde{k}}(\langle \mathbf{a}(i)^R, \mathbf{q}(i)^R \rangle), \tag{3.5}$$

using the Fiat-Shamir sequence $(s_1, \ldots, s_l)$. Eventually, the prover finds 1-dimensional vector $\mathbf{a}(l+1) \in \mathbf{F}_p$ and

$$\pi^0 \leftarrow ZKP^0_{ext}(pp, k(l+1), Com_{k(l+1)}(\mathbf{0}; \mathbf{r}(l+1)) \big|\big| Com_{\tilde{k}}(\mathbf{0}), \mathbf{r}(l+1)).$$

The prover outputs

$$\pi := (\mathbf{a}(l+1), \langle \mathbf{a}, \mathbf{q} \rangle, \{C_1(i)\}_{i=1}^l, \{C_2(i)\}_{i=1}^l, \pi^0).$$

Thus, π consists of two $\mathbf{F}_p$ elements, $2 \log n$ commitments, and the zero-knowledge proof π^0, so that its data size is $O(\log n)$. We implicitly assume the data size of Com is constant (only depend on the security parameter λ).

We summarize the final version of our protocol.

- **Setup**.
  Execute the setup algorithm of the vector commitment scheme $VC.Setup(1^\lambda) \to pp$, and output pp as the result of $Setup(1^\lambda)$.

- **Trim**.
  Execute the key generation algorithm of the vector commitment scheme $VC.KeyGen(pp, n, t) \to k$ and $VC.KeyGen(pp, 1, 0) \to \tilde{k}$, and output $(k, \tilde{k})$ as the result of $Trim$.

- **Commit**.
  The prover calculates $VC.Commit(pp, k, \mathbf{a}; \mathbf{r}) \rightarrow Com_k(\mathbf{a}; \mathbf{r})$ for the coefficient vector $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ of $f(X) \in \mathbf{F}_p[x]$, and outputs

$$Commit(k, f(X), M; \mathbf{r}) \rightarrow Com_k(\mathbf{a}; \mathbf{r}).$$

- **Open**. The prover defines

$$\mathbf{a}(1) := \mathbf{a}, \quad \mathbf{q}(1) := \mathbf{q}, \quad \mathbf{r}(1) := \mathbf{r}, \quad k(1) := k,$$

  and calculates $\mathbf{a}(l+1), \{C_1(i)\}_{i=1}^{l}, \{C_2(i)\}_{i=1}^{l}$ using the Fiat-Shamir sequence $(s_1, \ldots, s_l)$ and the equations (3.3), (3.4), (3.5). Furthermore, the prover generates

$$\pi^0 \leftarrow ZKP_{ext}^0\big(pp, k(l+1), Com_{k(l+1)}(\mathbf{0}; \mathbf{r}(l+1)) \big|\big| Com_{\tilde{k}}(\mathbf{0}), \mathbf{r}(l+1)\big),$$

  and eventually outputs

$$\pi = (\mathbf{a}(l+1), \langle \mathbf{a}, \mathbf{q} \rangle, \{C_1(i)\}_{i=1}^{l}, \{C_2(i)\}_{i=1}^{l}, \pi^0).$$

- **Verify**
  The verifier obtains $Com_k(\mathbf{a}; \mathbf{r})$ and

$$\pi = (\mathbf{a}(l+1), u, \{C_1(i)\}_{i=1}^{l}, \{C_2(i)\}_{i=1}^{l}, \pi^0),$$

  from the prover, and calculates

$$C(1) := Com_k(\mathbf{a}; \mathbf{r}) \big|\big| Com_{\tilde{k}}(u; \mathbf{0}),$$

  and

$$C(i+1) := C(i) + s_i \times C_1(i) + (s_i^2 - 1) \times C_2(i),$$

  for $i = 1, \ldots, l$. Also, the verifier calculates $\mathbf{q}(l+1) \in \mathbf{F}_p$ and

$$D := C(l+1) - Com_{k(l+1)}(\mathbf{a}(l+1); \mathbf{0}) || Com_{\tilde{k}}(\langle \mathbf{a}(l+1), \mathbf{q}(l+1) \rangle),$$

  using $(s_1, \ldots, s_l)$. Note that if the proof π is honestly generated, D is equal to $Com_{k(l)}(\mathbf{0}; \mathbf{r}(l+1)) \big|\big| Com_{\tilde{k}}(\mathbf{0})$. The verifier can check it using π^0, and outputs **True** if this verification is successful, and outputs **False** otherwise.

Discussion: Proof size and information leakage
In the protocol mentioned above, the data size of poof π is reduced to (2-$\mathbf{F}_p$ elements) $+(2 \log n \times |Com|) + |\pi^0|$.

Further, the information regarding the coefficients of the polynomial $f(X) = a_1 + a_2 X + \cdots + a_n X^{n-1}$ leaked from π is $\mathbf{a}(l+1)$ and $\langle \mathbf{a}, \mathbf{q} \rangle$. Furthermore, we can write $\mathbf{a}(l+1)$ by inner product form with the Fiat-Shamir

sequence $(s_1, \ldots, s_l)$. Let $b(i)_j$ be the j-th bit in the binary representation of $i = [b(i)_l \cdots b(i)_2 b(i)_1]_2$. we define

$$\mathbf{v}_i := \prod_{j=1}^{l} s_j^{b(i)_j}, \tag{3.6}$$

and $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_n)$. Then, it is easy verify that $\mathbf{a}(l+1) = \langle \mathbf{a}, \mathbf{v} \rangle$. Therefore, the information leakage of the coefficients of the polynomial $f(X)$ can be expressed in the form of the inner products $\{\langle \mathbf{a}, \mathbf{v} \rangle, \langle \mathbf{a}, \mathbf{q} \rangle\}$. Note that the original polynomial commitment schemes such as [59, 36] reveal $\langle \mathbf{a}, \mathbf{q} \rangle = f(q)$ to the verifier, and that is one dimensional information about the coefficients of $f(X)$. On the other hand, our polynomial commitment scheme reveals two dimensional information $\{\langle \mathbf{a}, \mathbf{v} \rangle, \langle \mathbf{a}, \mathbf{q} \rangle\}$. This information can be kept secret in the same way as in Section 5.3.2 [36] to modify $f(X)$ in the zero-knowledge protocol layer (see Section 3.5 for more details).

Theorem 10.
The aforementioned polynomial commitment scheme satisfies the completeness, extractability, and computational hiding with the honest verifier model.

Proof of completeness. The completeness is directly proved from the construction.

Proof of extractability. We show that there exists a PPT extractor $\mathcal{E}_{\mathcal{A}}(ck, \mathbf{c}, \mathbf{d}, Q, \mathbf{v}, \pi) \rightarrow (f(X), \mathbf{r})$ such that $\deg f \leq n-1$ and $Commit(ck, f, n, \mathbf{r}) = c$ for an attacker and sub-attacker pair $(\mathcal{A}, \mathcal{A}')$ who can pass the verification procedure. We use an inductive argument for the recursive steps and extract

$$(\mathbf{a}(l+1),\ \mathbf{r}(l+1)),\ (\mathbf{a}(l), \mathbf{r}(l)), \ldots, (\mathbf{a}(1), \mathbf{r}(1))$$

such that

$$C(i) = Com_{k(i)}(\mathbf{a}(i); \mathbf{r}(i)) \big|\big| Com_{\tilde{k}}(\langle \mathbf{a}(i), \mathbf{q}(i) \rangle).$$

In the first step, the proof

$$\pi = (\mathbf{a}(l+1), \langle \mathbf{a}, \mathbf{q} \rangle, \{C_1(i)\}_{i=1}^{l}, \{C_2(i)\}_{i=1}^{l}, \pi^0)$$

includes $\mathbf{a}(l+1)$. Furthermore, the (relaxed) extractability of ZKP_{ext}^0, there exists an extractor $\mathcal{E}$ who can extract a randomness $\mathbf{r}(l+1)$ such that $C(l+1) = Com_{k(l+1)}(\mathbf{a}(l+1); \mathbf{r}(l+1)) \big|\big| Com_{\tilde{k}}(\sim)$ from

$$(\pi^0, Com_{k(l+1)}(\mathbf{0}; \mathbf{r}(1+1)) \big|\big| Com_{\tilde{k}}(\sim)).$$

Next, we show that for each recursive step, we can extract a witness pair of $(\mathbf{a}(m-1), \mathbf{r}(m-1))$ from that of $(\mathbf{a}(m), \mathbf{r}(m))$. The extractor $\mathcal{E}_{\mathcal{A}}$

runs the prover $\mathcal{A}'$ to get the witnesses of $C(m-1), C_1(m-1)$ and $C_2(m-1)$. Then, by using forking lemma (see Section 3.2.6) four times giving random four challenges $s_1, s_2, s_3, s_4 \in \mathbf{F}_p$. The extractor gets four pairs $(\mathbf{a}(m)^{(i)}, \mathbf{r}(m)^{(i)}), i = 1, 2, 3, 4$ such that

$$\Big(C(m) = \Big) \;\; C(m-1) + s_i C_1(m-1) + (s_i^2 - 1)C_2(m-1)$$

$$= Com_{k(m)}(\mathbf{a}(m)^{(i)}; \mathbf{r}(m)^{(i)}) || Com_{\tilde{k}}(\langle \mathbf{a}(m)^{(i)}, \mathbf{q}(m)^{(i)} \rangle). \tag{3.7}$$

We use the first three challenges s_1, s_2, s_3, to compute $\alpha_1, \alpha_2, \alpha_3 \in \mathbf{F}_p$ such that

$$\sum_{i=1}^{3} \alpha_i = 1, \; \sum_{i=1}^{3} \alpha_i s_i = 0, \; \sum_{i=1}^{3} \alpha_i s_i^2 = 0.$$

Then taking a linear combination of the first three equations (3.7), with $\alpha_1, \alpha_2, \alpha_3$ as the coefficients, we can get

$$C(m-1) - C_2(m-1) = Com_{k(m)}\Big(\sum \alpha_i \mathbf{a}(m)^{(i)}; \sum \alpha_i \mathbf{r}(m)^{(i)} \Big)$$

$$|| Com_{\tilde{k}}\Big(\sum \alpha_i \langle \mathbf{a}(m)^{(i)}, \mathbf{q}(m)^{(i)} \rangle \Big). \tag{3.8}$$

Since $k(m) = k(m-1)^L * s_i k(m-1)^R$, using the key convolution equation, we have the equation

$$(3.8) = Com_{k(m-1)}\Big(\sum \alpha_i \mathbf{a}(m)^{(i)} || \sum \alpha_i s_i \mathbf{a}(m)^{(i)}; R \Big)$$

$$|| Com_{\tilde{k}}\Big(\sum \alpha_i \langle \mathbf{a}(m)^{(i)}, \mathbf{q}(m)^{(i)} \rangle \Big), \tag{3.9}$$

where $R = \sum \alpha_i \mathbf{r}(m)^{(i)}$. We define

$$\mathbf{b} := \mathbf{b}^L || \mathbf{b}^R = \sum \alpha_i \mathbf{a}(m)^{(i)} || \sum \alpha_i s_i \mathbf{a}(m)^{(i)}.$$

We take $\beta_1, \beta_2, \beta_3, \gamma_1, \gamma_2, \gamma_3 \in \mathbf{F}_p$ such that

$$\sum_{i=1}^{3} \beta_i = 0, \; \sum_{i=1}^{3} \beta_i s_i = 1, \; \sum_{i=1}^{3} \beta_i s_i^2 = 0,$$

$$\sum_{i=1}^{3} \gamma_i = 0, \; \sum_{i=1}^{3} \gamma_i s_i = 0, \; \sum_{i=1}^{3} \gamma_i s_i^2 = 1,$$

and repeat the same process with these coefficients, we can also compute

$$C_1(m-1) = Com_{k(m-1)}\Big(\sum \beta_i \mathbf{a}(m)^{(i)} || \sum \beta_i s_i \mathbf{a}(m)^{(i)}; R_1 \Big)$$

$$|| Com_{\tilde{k}}\Big(\sum \beta_i \langle \mathbf{a}(m)^{(i)}, \mathbf{q}(m)^{(i)} \rangle \Big), \tag{3.10}$$

and

$$C_2(m-1) = Com_{k(m-1)}\Big(\sum \gamma_i \mathbf{a}(m)^{(i)} || \sum \gamma_i s_i \mathbf{a}(m)^{(i)}; R_2\Big)$$
$$||Com_{\tilde{k}}\Big(\sum \gamma_i \langle \mathbf{a}(m)^{(i)}, \mathbf{q}(m)^{(i)} \rangle\Big). \tag{3.11}$$

We also define

$$\mathbf{b}_1 := \mathbf{b}_1^L || \mathbf{b}_1^R = \sum \beta_i \mathbf{a}(m)^{(i)} || \sum \beta_i s_i \mathbf{a}(m)^{(i)},$$
$$\mathbf{b}_2 := \mathbf{b}_2^L || \mathbf{b}_2^R = \sum \gamma_i \mathbf{a}(m)^{(i)} || \sum \gamma_i s_i \mathbf{a}(m)^{(i)}.$$

Consequently, we can rewrite the left hand side of the equation (3.7) as:

$$Com_{k(m-1)}\Big(\mathbf{b} + s_i \mathbf{b}_1 + s_i^2 \mathbf{b}_2; R'\Big) || Com_{\tilde{k}}(\sim)$$
$$= Com_{k(m)}(\mathbf{a}(m)^{(i)}; \mathbf{r}(m)^{(i)}) || Com_{\tilde{k}}(\sim)$$
$$= Com_{k(m-1)^L * s_i k(m-1)^R}(\mathbf{a}(m)^{(i)}; \mathbf{r}(m)^{(i)}) || Com_{\tilde{k}}(\sim)$$
$$= Com_{k(m-1)}(\mathbf{a}(m)^{(i)} || s_i \mathbf{a}(m)^{(i)}; \mathbf{r}(m)^{(i)})) || Com_{\tilde{k}}(\sim), \tag{3.12}$$

for each $i = 1, 2, 3, 4$. Since the binding property for the commitment key $k(m-1)$, the equation (3.12) shows

$$s_i\big(\mathbf{b}^L + s_i \mathbf{b}_1^L + s_i^2 \mathbf{b}_2^L\big) = \mathbf{b}^R + s_i \mathbf{b}_1^R + s_i^2 \mathbf{b}_2^R$$
$$\iff s_i^3 \mathbf{b}_2^L + s_i^2(\mathbf{b}_1^L - \mathbf{b}_2^R) + s_i(\mathbf{b}^L - \mathbf{b}_1^R) - \mathbf{b}^R = \mathbf{0},$$

for all i. The vector coefficient polynomial

$$\mathbf{b}_2^L X^3 + (\mathbf{b}_1^L - \mathbf{b}_2^R) X^2 + (\mathbf{b}^L - \mathbf{b}_1^R) X - \mathbf{b}^R = \mathbf{0}$$

has four different roots $s_1, s_2, s_3, s_4 \in \mathbf{F}_p$, so that it is equal to the $\mathbf{0}$ polynomial. Thus, we have

$$\mathbf{b}_2^L = \mathbf{b}^R = \mathbf{0}, \quad \mathbf{b}_1^L = \mathbf{b}_2^R, \quad \mathbf{b}^L = \mathbf{b}_1^R,$$

and take $\mathbf{a}(m-1) = \mathbf{a}(m-1)^L || \mathbf{a}(m-1)^R := \mathbf{b}^L || \mathbf{b}_2^R$. We also have desired results:

$$C(m-1) = Com_{k(m-1)}(\mathbf{a}(m-1)^L || \mathbf{a}(m-1)^R; R + R_2) || Com_{\tilde{k}}(\sim),$$
$$C_1(m-1) = Com_{k(m-1)}(\mathbf{a}(m-1)^R || \mathbf{a}(m-1)^L; R_1) || Com_{\tilde{k}}(\sim),$$

and

$$C_2(m-1) = Com_{k(m-1)}(\mathbf{0} || \mathbf{a}(m-1)^R; R_2) || Com_{\tilde{k}}(\sim).$$

Furthermore, it is easy verify that the inner product part of $C(m-1)$ is equal to

$$\begin{aligned}
&\sum \alpha_i \langle \mathbf{a}(m)^{(i)}, \mathbf{q}(m)^{(i)} \rangle + \sum \gamma_i \langle \mathbf{a}(m)^{(i)}, \mathbf{q}(m)^{(i)} \rangle \\
&= \sum \alpha_i \langle \mathbf{a}(m)^{(i)}, \mathbf{q}(m-1)^{L} \rangle + \sum \alpha_i s_i \langle \mathbf{a}(m)^{(i)}, \mathbf{q}(m-1)^{R} \rangle \\
&+ \sum \gamma_i \langle \mathbf{a}(m)^{(i)}, \mathbf{q}(m-1)^{L} \rangle + \sum \gamma_i s_i \langle \mathbf{a}(m)^{(i)}, \mathbf{q}(m-1)^{R} \rangle \\
&= \langle \mathbf{a}(m-1)^{L}, \mathbf{q}(m-1)^{L} \rangle + \langle \mathbf{a}(m-1)^{R}, \mathbf{q}(m-1)^{R} \rangle + \mathbf{0} + \mathbf{0} \\
&= \langle \mathbf{a}(m-1), \mathbf{q}(m-1) \rangle
\end{aligned}$$

because $\mathbf{q}(m)^{(i)} = \mathbf{q}(m-1)^{L} + s_i \times \mathbf{q}(m-1)^{R}$.

Thus, the extractor $\mathcal{E}_{\mathcal{A}}$ recursively extracts witness pairs

$$(\mathbf{a}(l+1),\ \mathbf{r}(l+1)),\ (\mathbf{a}(l), \mathbf{r}(l)), \ldots, (\mathbf{a}(1), \mathbf{r}(1)),$$

and, eventually, outputs a witness $(\mathbf{a}(1), \mathbf{r}(1))$ of the original commitment $(f(X), \mathbf{r}(1))$ such that $f(X) = \mathbf{a}(1)_1 + \mathbf{a}(1)_2 X^2 \cdots + \mathbf{a}(1)_n X^{n-1}$. Furthermore, $\deg f = n - 1 \leq M$. Note that this procedure consists of a quad-tree with forking lemma (see Section 3.2.6). We can see that the extractor $\mathcal{E}_{\mathcal{A}}$ uses $4^{\log n} = n^2$ steps in total and thus runs polynomial in n.

Proof of hiding. We construct a PPT simulator, namely, $\mathcal{S}im$ for this commitment scheme. Since the extractable zero-knowledge property ZKP^{0}_{ext}, there exists a PPT simulator $\mathcal{S}im'$ with a trapdoor $trap'$ such that for any PPT adversaries $\mathcal{A}$ can not distinguish simulator's outputs or the real-world outputs. We take the sequence $(s_1, \ldots, s_l)$ as the trapdoor $trap$ for $\mathcal{S}im$.

Let $(Com_k(\mathbf{a}; \mathbf{r}), \langle \mathbf{a}, \mathbf{q} \rangle, \mathbf{a}(l+1))$ be a commitment and its evaluations. $\mathcal{S}im$ uses $trap$ and $(\mathcal{S}im', trap')$ in the following manner. $\mathcal{S}im$ generates random commitment values

$$\{\widetilde{C_1(i)}\}_{i=1}^{l}, \quad \{\widetilde{C_2(i)}\}_{i=1}^{l},$$

which are distributed identically to the real prover's outputs from the hiding property. Next, $\mathcal{S}im$ calculates

$$\widetilde{C(1)} := Com_k(\mathbf{a}; \mathbf{r}) || Com_{\tilde{k}}(\langle \mathbf{a}(l+1), \mathbf{q}(l+1) \rangle),$$

$$\widetilde{C(i+1)} := \widetilde{C(i)} + s_i \widetilde{C_1(i)} + (s_i^2 - 1)\widetilde{C_2(i)},$$

using $(s_1, \ldots, s_l)$ for $i = 1, \ldots, l$, and

$$D := \widetilde{C(l+1)} - Com_{k(l)}(\mathbf{a}(l+1); \mathbf{0}) || Com_{\tilde{k}}(\langle \mathbf{a}(l+1), \mathbf{q}(l+1) \rangle).$$

Furthermore, $\mathcal{S}im$ gives the commitment value D and $trap'$ to $\mathcal{S}im'$ and obtains π^0. Eventually, $\mathcal{S}im$ outputs

$$(\langle \mathbf{a}, \mathbf{q} \rangle,\ \mathbf{a}(l+1), \{\widetilde{C_1(i)}\}_{i=1}^{l}, \quad \{\widetilde{C_2(i)}\}_{i=1}^{l}, \pi^0).$$

This is a valid proof for the commitment scheme and $\mathcal{A}$ can't distinguish it from the zero-knowledge property of ZKP^0_{ext} and hiding property of $VC.Commit$.

3.5 Application for transparent ZKPs

In this section, we describe the construction of universal and transparent ZKPs as an application of our polynomial commitment schemes. We applied our commitment schemes to Marlin protocol [36]. In this section, we assume that the reader is somewhat familiar with Marlin protocol (see Section 3.5.2 for the description of the protocol).

3.5.1 Overview of Marlin protocol

Marlin protocol is roughly divided into offline and online phases. In the offline phase, a TTP generates system parameters that are used for ZKP of R1CS, and in the online phase, the prover and verifier actually perform ZKP protocol. In addition, the offline phase is further divided into a setup phase that creates a universal SRS and an indexing phase that creates the parameters (commitment and polynomial system) for the R1CS instance. In the original Marlin protocol, the setup algorithm is executed by a trusted third party because it requires secret information.

Let $(\mathcal{I}, x, w) = ((\mathbf{F}_p, H, K, A, B, C), x, w)$ be an R1CS indexed relation for given R1CS. In other words, $H, K \subset \mathbf{F}_p$ and A, B, C are $|H| \times |H|$ matrices over $\mathbf{F}_p$ with $|K| \geq \max\{||A||, ||B||, ||C||\}$, where $||M||$ is the number of non-zero entries of M. The R1CS equation holds if and only if $Az \circ Bz = Cz$ for $z := (x, w) \in \mathbf{F}_p^{|H|}$. The symbol $\circ$ means the entry wise multiplication of two vectors. We call the vector x the statement and w the witness of the R1CS equation. Furthermore, we denote $z_A := Az, \quad z_B := Bz, \quad z_C := Cz$ and take polynomials

$$z_A(X), z_B(X), z_C(X) \in \mathbf{F}_p[X]^{\leq |H|+1}, w(X) \in \mathbf{F}_p[X]^{\leq |w|+1}$$

that agree with $z_A := Az, z_B := Bz, z_C := Cz$ and w on H. The prover is left to convince the verifier that the following two condition holds (see [36] for more details).

1. Entry-wise product: $z_A(h)z_B(h) - z_C(h) = 0, \forall h \in H$.
2. Linear relation: $z_M(h) = \sum_{i \in H} M[h, i]z(i), \forall h \in H, \forall M \in \{A, B, C\}$.

Roughly speaking, the first equation shows the R1CS equation holds, and the second equation shows $z_A(X), z_B(X), z_C(X)$ are correctly generated.

Tweak: Achieving zero-knowledge property

To prove the two equations above, the prover generates 10 polynomial commitments that are related to $z_A(X), z_B(X), z_C(X)$, and $w(X)$. Furthermore,

the prover executes polynomial commitment protocol with the verifier. We apply our commitment schemes to this point. We denote this polynomial set $\{f_j(X)\}_{j=1}^{10}$.

Also, our polynomial commitment scheme reveals 2-dimensional information about the coefficients of the committed polynomials so that we must take $z_A(X), z_B(X), z_C(X)$, and $w(X)$ such that the value of up to two locations in each $z_A(X), z_B(X), z_C(X)$, and $w(X)$ reveal no information about the witness w (see Section 5.3.2 of the Marlin paper [36] for more details). Similarly, the leakage locations for other polynomials $(s(X), g_1(X), h_1(X))$ in $\{f_j(X)\}_{j=1}^{10}$ are doubled, so the same processing is needed.

From Theorem 8.1 in [36], our proposed protocols below are universal ZKPs because the underlying polynomial commitment scheme satisfies the extractability and the hiding properties.

3.5.2 Marlin protocol

The outline of each phase of Marlin is shown below.

Setup. The setup algorithm, on input a security parameter $\lambda \in \mathbf{N}$ and the maximum size $N \in \mathbf{N}$ of R1CS, uses N to compute the degree bound $D \in \mathbf{N}$, samples public parameters $pp \leftarrow PC.Setup(1^\lambda, D)$ for the commitment scheme PC, and outputs a structure reference strings $SRS := pp$.

Index. The indexing algorithm, on input SRS and the index $\mathcal{I} = (\mathbf{F}_p, H, K, A, B, C)$, deterministically calculates the commitment/verify keypair $(ck, rk) \leftarrow PC.Trim^{SRS}(D)$, a polynomial set $\{p_i(X)\}$, uses $\{p_i(X)\}$ to compute a knowledge proof, and commitments $\{c_j\} \leftarrow PC.Commit_{ck}$ for *"empty randomness"*. The set of commitments $\{c_j\}$ corresponds to the statement part x of R1CS. The index algorithm eventually outputs $ipk := (ck, \mathcal{I}, \{p_i(X)\}, \{c_j\})$ for the prover and $ivk := (rk, \{c_j\})$ for the verifier.

Online phase. The prover $\mathbf{P}$ receives (ipk, x, w) and the verifier $\mathbf{V}$ receives (ivk, x).

1. The prover $\mathbf{P}$ calculates random polynomials $z_A(X), z_B(X), z_C(X) \in \mathbf{F}_p[X]^{\leq |H|+1}, w(X) \in \mathbf{F}_p[X]^{\leq |w|+1}$, and finds $h_0(X) \in \mathbf{F}_p[X]$ such that $z_A(X)z_B(X) - z_C(X) = h_0(X)V_H(X)$. Furthermore, the prover $\mathbf{P}$ samples a random mask polynomial $s(X) \in \mathbf{F}_p^{\leq 2|H|}$ and calculates its sum $\sigma_1 := \sum_{\kappa H} s(\kappa)$. In practice, we can take $s(X) = X\tilde{s}(X)$ and thus $\sigma_1 = 0$. The rest of this protocol, we assume $\sigma_1 = 0$.

2. The prover $\mathbf{P}$ calculates commitments

$$Com_{ck}(z_A(X)), Com_{ck}(z_B(X)), Com_{ck}(z_C(X)),$$

$$Com_{ck}(w(X)), Com_{ck}(h_0(X)), Com_{ck}(s(X)),$$

and sends them to the verifier $\mathbf{V}$.

3. The verifier $\mathbf{V}$ samples random elements $\alpha, \eta_A, \eta_B, \eta_C \leftarrow \mathbf{F}_p$ and sends them to the prover $\mathbf{P}$.

4. The prover $\mathbf{P}$ calculates the first sumcheck polynomial $F_1(X)$ using $\alpha, \eta_A, \eta_B, \eta_C$ and finds $g_1(X), h_1(X)$ such that

$$F_1(X) = h_1(X)V_H(X) + Xg_1(X).$$

5. The prover $\mathbf{P}$ calculates commitments

$$Com_{ck}(h_1(X)), Com_{ck}(g_1(X)),$$

and sends them to the verifier $\mathbf{V}$.

6. The verifier $\mathbf{V}$ samples a random element $\beta_1 \leftarrow \mathbf{F}_p$ and sends it to the prover $\mathbf{P}$.

7. The prover $\mathbf{P}$ calculates the second sumcheck polynomial $F_2(X)$ using β_1 and its sum $\sigma_2 := \sum_{\kappa H} F_2(\kappa)$. Furthermore, the prover $\mathbf{P}$ finds $g_2(X), h_2(X)$ such that

$$F_2(X) = h_2(X)V_H(X) + Xg_2(X) + \sigma_2/|H|.$$

Eventually, the prover $\mathbf{P}$ calculates

$$Com_{ck}(h_2(X)), Com_{ck}(g_2(X)),$$

and sends them and σ_2 to the verifier $\mathbf{V}$.

8. The verifier $\mathbf{V}$ samples a random element $\beta_2 \leftarrow \mathbf{F}_p$ and sends it to the prover $\mathbf{P}$.

9. The prover $\mathbf{P}$ calculates the third sumcheck polynomials $a_3(X)$, $b_3(X)$ using β_1, β_2 and its sum $\sigma_3 := \sum_{\kappa K} a_3(\kappa)/b_3(\kappa)$. Furthermore, the prover $\mathbf{P}$ finds $g_3(X), h_3(X)$ such that

$$h_3(X)V_K(X) = a_3(X) - b_3(X)(Xg_3(X) + \sigma_3/|K|).$$

Eventually, the prover $\mathbf{P}$ calculates

$$Com_{ck}(h_3(X)), Com_{ck}(g_3(X)),$$

and sends them and σ_3 to the verifier $\mathbf{V}$.

10. The verifier $\mathbf{V}$ samples a random element $\beta_3 \leftarrow \mathbf{F}_p$ and sends it to the prover $\mathbf{P}$.

11. The prover **P** calculates $h_3(\beta_3), g_3(\beta_3)$ and zero-knowledge proof $\pi^{(3)}$ that proves $h_3(\beta_3), g_3(\beta_3)$ are evaluated values of the committed polynomials $Com_{ck}(h_3(X)), Com_{ck}(g_3(X))$ at β_3. In addition, the prover **P** also calculates $h_2(\beta_2), g_2(\beta_2)$ and its zero-knowledge proof $\pi^{(2)}$ and

$$w(\beta_1), s(\beta_1), h_1(\beta_1), g_1(\beta_1), z_A(\beta_1), z_B(\beta_1), z_C(\beta_1), h_0(\beta_1),$$

and its zero-knowledge proof $\pi^{(1)}$. The equation of the entry-wise product is independent of the step $(4) - (10)$; thus we can check it at the same point β_1. Eventually, the prover **P** sends all evaluated values

$$w(\beta_1), s(\beta_1), z_A(\beta_1), z_B(\beta_1), z_C(\beta_1), h_0(\beta_1)\{h_i(\beta_i)\}, \{g_i(\beta_i)\},$$

and proofs $(\pi^{(1)}, \pi^{(2)}, \pi^{(3)})$ to the verifier **V**.

12. The verifier **V** verifies $(\pi^{(1)}, \pi^{(2)}, \pi^{(3)})$ are valid proofs and the following 4 equations hold:

$$z_A(\beta_1) * z_B(\beta_1) - z_C(\beta_1) = h_0(\beta_1) * V_H(\beta_1),$$

$$s(\beta_1) + F_1(\beta_1) - \sigma_2 z(\beta_1) = h_1(\beta_1) V_H(\beta_1) + \beta_1 g_1(\beta_1),$$

$$r(\alpha, \beta_2)\sigma_3 = h_2(\beta_2) V_H(\beta_2) + \beta_2 g_2(\beta_2) + \sigma_2/|H|,$$

$$h_3(\beta_3) * V_K(\beta_3) = a_3(\beta_3) - b_3(\beta_3)(\beta_3 g_3(\beta_3) + \sigma_3/|K|).$$

If all validations are successful, the verifier **V** outputs **True**, otherwise **False**.

3.5.3 Proposal 1. Universal and transparent ZKP from discrete logarithm

In this section, we show a construction of a universal and transparent ZKP based on the Marlin protocol under the discrete log assumption. We use the same notations in the previous section. Let **G** be a discrete group order $p \in \mathbf{Z}$ with the discrete log assumption.

Proposal 1

Setup. The setup algorithm, on input a security parameter $\lambda \in \mathbf{N}$ and the maximum size $N \in \mathbf{N}$ of R1CS, uses N to compute the degree bound $(D-1) \in \mathbf{N}$, where $D = 2^l$, samples public parameters

$$PC.Setup(1^\lambda, D) \to \big(\mathbf{G}, (g_1, g_2, \dots g_D; h_1, h_2)\big)$$

of the Pedersen vector commitment scheme, and outputs a structure reference strings $SRS := \big(\mathbf{G}, (g_1, g_2, \dots g_D; h_1, h_2)\big)$.

Index. On input SRS and the index $\mathcal{I} = (\mathbf{F}_p, H, K, A, B, C)$, calculates the commitment key $ck \leftarrow PC.Trim^{SRS}(1^\lambda, \mathbf{d}, D)$, a polynomial set $\{p_i(X)\}$, and commitments $\{c_j\} \leftarrow PC.Commit_{ck}$ for *"empty randomness"*. The set of commitments $\{c_j\}$ corresponds to the statement part x of R1CS and $\{p_i(X)\}$ is used for generating a proof. The index algorithm outputs prover-key $pk := (ck, \mathcal{I}, \{p_i(X)\}, \{c_j\})$ for the prover and verifier-key $vk := (ck, \{c_j\})$ for the verifier.

Online phase. We apply all commitment values that appear in the online phase as Pedersen commitments. More precisely, let $f(X) = a_1 + a_2 X + \cdots a_D X^{D-1} \in \{f_j(X)\}_{j=1}^{10}$ be a polynomial in the online phase. We define

$$Com_{ck}(f(X); r, 0) := \Big(\prod_{i=1}^{D} g_i^{a_i}\Big) h_1^r h_2^0.$$

Furthermore, a zero-knowledge proof π that proves $f(q) = u$ is constructed as follows:

- The prover $\mathbf{P}$ commits $Com_{ck}(f(X); r, 0)$.
- The verifier $\mathbf{V}$ samples a random element $q \leftarrow \mathbf{F}_p$ and sends it to the prover $\mathbf{P}$.
- The prover $\mathbf{P}$ sends $u = f(q)$ $(= \langle \mathbf{a}, \mathbf{q} \rangle)$ to the verifier $\mathbf{V}$.
- The verifier $\mathbf{V}$ calculates

$$C(1) := Com_{ck}(f(X); r, 0) * h_2^u = \Big(\prod_{i=1}^{D} g_i^{a_i}\Big) h_1^r h_2^u,$$

- The prover $\mathbf{P}$ calculates $(C_1(1), C_2(1))$ using the equations (3.4), (3.5), and sends it to the verifier $\mathbf{V}$.
- The verifier samples a random element $s_1 \leftarrow \mathbf{F}_p$ and sends it to the prover $\mathbf{P}$, and updates $C(1)$ to $C(2) := C(1) + s_1 C_1(1) + (s_1^2 - 1) C_2(1)$.
- The prover $\mathbf{P}$ calculates $(C_1(2), C_2(2))$ using the equations (3.3), (3.4), (3.5), and sends it to the verifier $\mathbf{V}$.

$$\vdots$$

- The verifier samples a random element $s_l \leftarrow \mathbf{F}_p$ and sends it to the prover $\mathbf{P}$, and updates $C(l)$ to $C(l+1) := C(l) + s_l C_1(l) + (s_l^2 - 1) C_2(l)$.
- The prover $\mathbf{P}$ calculates $\mathbf{a}(l+1)$, and sends it to the verifier $\mathbf{V}$. At this time, the convoluted commitment value is

$$C(l+1) = g(l+1)^{\mathbf{a}(l+1)} * h_1^{r(l+1)} * h_2^{\mathbf{a}(l+1) * \mathbf{q}(l+1)}, \tag{3.13}$$

and the verifier $\mathbf{V}$ can find $C(l+1), g(l+1), \mathbf{a}(l+1)$, and $\mathbf{q}(l+1)$. Thus, the prover $\mathbf{P}$ generates a

$$\pi^0 \leftarrow ZKP^0_{ext}(pp, k(l+1), Com_{k(l+1)}(\mathbf{0}; r(l+1)), r(l+1)),$$

and sends it to the verifier $\mathbf{V}$.

- The verifier $\mathbf{V}$ calculates

$$(h_1^{r(l+1)} =) \;\; C(l+1) \cdot g(l+1)^{-\mathbf{a}(l+1)} h_2^{-\mathbf{a}(l+1)*\mathbf{q}(l+1)},$$

and verifies it using π^0. Eventually, the validation is successful, the verifier $\mathbf{V}$ outputs **True**, otherwise **False**.

Batch verification for proposal 1

The batch process, which verifies multiple proofs simultaneously, is an important optimization. In many applications such as blockchain or cryptocurrencies, the verifier verifies many zero-knowledge proofs added to transactions from users, so that it is necessary to be able to perform these processes at once.

The most heavy processing in the verification procedure is the computing for the convoluted commitment key $g(l+1)^{-\mathbf{a}(l+1)}$. In fact, in our experiments, this processing occupied about 99% of the total, and this procedure consists of $\deg f$ times scalar-multiplications on $\mathbf{G}$. Using the vector $\mathbf{v} \in \mathbf{F}_p^D$ in the equation (3.6), we can write

$$g(l+1)^{-\mathbf{a}(l+1)} = \Big(\prod_{i=1}^{D} g_i^{\mathbf{v}_i}\Big)^{-\mathbf{a}(l+1)} = \prod_{i=1}^{D} g_i^{-\mathbf{a}(l+1)\mathbf{v}_i}.$$

Consequently, if there are T proofs $\pi^{(1)}, \ldots, \pi^{(T)}$, the verifier, in advance, calculates

$$\phi_i := -\sum_{j=1}^{T} \mathbf{a}(l+1)^{(j)} \mathbf{v}_i^{(j)}, \quad i = 1, \ldots, D,$$

where $\pi^{(j)} = (\langle \mathbf{a}^{(j)}, \mathbf{q}^{(j)} \rangle, \{C_1(i)\}_i, \{C_2(i)\}_i, \mathbf{a}(l+1)^{(j)}, (\pi^0)^{(j)})$, and $\mathbf{v}^{(j)}$ is the vector generated by the j-th Fiat-Shamir sequence $(s_1^{(j)}, \ldots, s_l^{(j)})$ with the equation (3.6). Therefore, we have

$$I := \prod_{j=1}^{T} (g(l+1)^{(j)})^{-\mathbf{a}(l+1)^{(j)}} = \prod_{i=1}^{D} g_i^{\phi_i} \tag{3.14}$$

The verifier can calculate the right-hand side of the equation (3.14) with $\deg f$ times scalar-multiplications on $\mathbf{G}$, and this number is independent of the number of proofs T.

The batch verification processing is described below.

Batch verification. Let $User_1, \ldots, User_T$ be T provers who have the same setup parameters and $User_j$ generates a proof $\pi^{(j)}$. Also, we denote $User_j$'s $h_1^{r(l+1)}$ in the equation (3.13) by $h_1^{r(l+1)^{(j)}}$. The prover $User_j$ sends $(\pi^{(j)}, h_1^{r(l+1)^{(j)}})$ to the verifier as a new proof. Note that the prover newly sends $h_1^{r(l+1)^{(j)}}$, but this value can be calculated by the verifier. Thus the leakage of the coefficient information is the same as in the previous protocol.

The verifier calculates $C(l+1)^{(j)}, \mathbf{q}(l+1)^{(j)}$ for all $j = 1, \ldots, T$, and

$$J := \Big(\prod_{j=1}^{T} h_1^{r(l+1)^{(j)}}\Big)\Big(\prod_{j=1}^{T} C(l+1)^{(j)}\Big)^{-1} h_2^{\sum_j \mathbf{a}(l+1)^{(j)}\mathbf{q}(l+1)^{(j)}}.$$

From the equations (3.13), (3.14), if all proofs are correctly generated, then $I = J$. The verifier also checks the zero-knowledge proof $(\pi^0)^{(j)}$ using $h_1^{r(l+1)^{(j)}}$ for each j. Eventually, the validations are successful, the verifier $\mathbf{V}$ outputs **True**, otherwise **False**.

3.5.4 Proposal 2. Universal and transparent ZKP from lattice

In this section, we also show the construction of universal and transparent ZKP under the SIS assumption (see Section 3.2.5). Let $p \in \mathbf{Z}$ be a prime number and $q \in \mathbf{Z}$ be a modulus $p << q$.

Setup. The setup algorithm, on input a security parameter $\lambda \in \mathbf{N}$ and the maximum size $N \in \mathbf{N}$ of R1CS, uses N to compute the degree bound $(D-1) \in \mathbf{N}$, where $D = 2^l$, samples public parameters

$$PC.Setup(1^\lambda, D) \to \big(\mathbf{A}_1, \mathbf{A}_2\big)$$

for the Ajtai vector commitment scheme PC, and outputs a structure reference strings $SRS := \big(\mathbf{A}_1, \mathbf{A}_2\big)$, where $\mathbf{A}_1 \in \mathbf{Z}_q^{m \times D}, \mathbf{A}_2 \in \mathbf{Z}_q^{(m+1)\times 2m \log_p q}$.

Index. The indexing algorithm is the same as the proposal 1.

Online phase. We apply all commitment values that appear in the online phase as Ajtai commitments. More precisely, let $f(X) = a_1 + a_2 X + \cdots a_D X^{D-1} \in \{f_j(X)\}_{j=1}^{10}$ be a polynomial in the online phase. We define

$$Com_{ck}(f(X); \mathbf{r}) := \begin{bmatrix} \mathbf{A}_1 \mathbf{a} \\ \hline 0 \end{bmatrix} + \mathbf{A}_2 \mathbf{r} \quad \in \mathbf{Z}_q^{m+1}.$$

Furthermore, a zero-knowledge proof π that proves $f(q) = u$ is constructed as follows:

- The prover $\mathbf{P}$ commits $Com_{ck}(f(X); \mathbf{r})$.

- The verifier $\mathbf{V}$ samples a random element $q \leftarrow \mathbf{F}_p$ and sends it to the prover $\mathbf{P}$.
- The prover $\mathbf{P}$ sends $u = f(q)$ $(= \langle \mathbf{a}, \mathbf{q} \rangle)$ to the verifier $\mathbf{V}$.
- The verifier $\mathbf{V}$ calculates

$$C := \left[\frac{\mathbf{A}_1 \mathbf{a}}{0} \right] + \mathbf{A}_2 \mathbf{r} + (0, \ldots, 0, u)^T \quad \in \mathbf{Z}_q^{m+1},$$

- The prover $\mathbf{P}$ calculates $(C_1(1), C_2(1))$ using the equations (3.4), (3.5), and sends it to The verifier $\mathbf{V}$.
- The verifier $\mathbf{V}$ samples a random element $s_1 \leftarrow \mathbf{F}_p$ and sends it to the prover $\mathbf{P}$.

$$\vdots$$

- The prover $\mathbf{P}$ calculates $\mathbf{a}(l+1)$, and sends it to the verifier $\mathbf{V}$. At this time, the convoluted commitment value is

$$C(l+1) = \left[\frac{\mathbf{A}_1(l+1)\mathbf{a}(l+1)}{0} \right] + \mathbf{A}_2 \mathbf{r}(l+1) + \left[\frac{\mathbf{0}}{\mathbf{a}(l+1)\mathbf{q}(l+1)} \right],$$

  and the verifier $\mathbf{V}$ can find $C(l+1), \mathbf{A}_1(l+1), \mathbf{a}(l+1)$, and $\mathbf{q}(l+1)$. Thus, the prover $\mathbf{P}$ generates a zk-SNARK π^0 that proves I know $\mathbf{r}(l+1)$ such that

$$\mathbf{A}_2 \mathbf{r}(l+1) = C(l+1) - \left[\frac{\mathbf{A}_1(l+1)\mathbf{a}(l+1)}{\mathbf{a}(l+1)\mathbf{q}(l+1)} \right],$$

  and sends it to the verifier $\mathbf{V}$.
- The verifier $\mathbf{V}$ verifies the above equation using π^0. Eventually, the validation is successful, the verifier $\mathbf{V}$ outputs **True**, otherwise **False**.

Batch verification for proposal 2

Here, we present the batch verification process for our proposal 2 in Section 3.5.4. We use the same notations in Section 3.5.4.

The most heavy processing in the verification procedure is the computing for the convoluted commitment key $-\mathbf{A}_1(l+1)\mathbf{a}(l+1)$. In fact, in our experiments, this processing occupied more than 98% of the total. Using the vector $\mathbf{v} \in \mathbf{F}_p^D$ in the equation (3.6), we can write

$$-\mathbf{A}_1(l+1)\mathbf{a}(l+1) = -\mathbf{a}(l+1) \sum_{i=1}^{D} \mathbf{v}_i \Phi_i = -\sum_{i=1}^{D} \mathbf{a}(l+1)\mathbf{v}_i \Phi_i,$$

where $\mathbf{A}_1 = [\Phi_1 || \cdots || \Phi_D]$.

Consequently, if there are T proofs $\pi^{(1)}, \ldots, \pi^{(T)}$, the verifier, in advance, calculates

$$\phi_i := -\sum_{j=1}^{T} \mathbf{a}(l+1)^{(j)} \mathbf{v}_i^{(j)}, \quad i = 1, \ldots, D,$$

where $\pi^{(j)} = (\langle \mathbf{a}^{(j)}, \mathbf{q}^{(j)} \rangle, \{C_1(i)\}_i, \{C_2(i)\}_i, \mathbf{a}(l+1)^{(j)}, (\pi^0)^{(j)})$, and $\mathbf{v}^{(j)}$ is the vector generated by the j-th Fiat-Shamir sequence $(s_1^{(j)}, \ldots, s_l^{(j)})$ with the equation (3.6). Therefore, we have

$$I := \sum_{j=1}^{T} -\mathbf{a}(l+1)^{(j)} \mathbf{A}_1 (l+1)^{(j)} = \sum_{i=1}^{D} \phi_i \Phi_i \tag{3.15}$$

The verifier can calculate the right-hand side of the equation (3.15) and it is independent of the number of proofs T.

The batch verification processing is described below.

Batch verification. Let $User_1, \ldots, User_T$ be T provers who have the same setup parameters and $User_j$ generates a proof $\pi^{(j)}$. The prover $User_j$ sends $(\pi^{(j)}, \mathbf{A}_2 \mathbf{r}(l+1)^{(j)})$ to the verifier as a new proof. Note that the prover newly sends $\mathbf{A}_2 \mathbf{r}(l+1)^{(j)}$, but this value can be calculated by the verifier. Thus the leakage of the coefficient information is the same as in the previous protocol.

The verifier calculates $C(l+1)^{(j)}, \mathbf{q}(l+1)^{(j)}$ for all $j = 1, \ldots, T$, I using the equation (3.15), and

$$J := \Big(\sum_{j=1}^{T} \mathbf{A}_2 \mathbf{r}(l+1)^{(j)} \Big) - \Big(\sum_{j=1}^{T} C(l+1)^{(j)} \Big) + \left[\frac{\mathbf{0}}{\sum_j \mathbf{a}(l+1)^{(j)} \mathbf{q}(l+1)^{(j)}} \right].$$

If all proofs were correctly generated, then $I = J$. The verifier also checks the zero-knowledge proof $(\pi^0)^{(j)}$ using $\mathbf{A}_2 \mathbf{r}(l+1)^{(j)}$ for each j. Eventually, the validations are successful, the verifier $\mathbf{V}$ outputs **True**, otherwise **False**.

3.6 Performance

3.6.1 Theoretical performance

We now evaluate the computational and proof complexity of our proposed schemes 1, 2. Throughout this section, let D be the degree of the polynomial $f(X)$ over $\mathbf{F}_p$, and we use the same notations in the previous section.

Table 3.3 shows the theoretical performance of our proposed scheme 1, 2. Here, **SM** and **Ad** denote the scalar-multiplication and addition on $\mathbf{G}$; also, **FO** denotes the addition or multiplication on $\mathbf{F}_p$ or $\mathbf{Z}_q$, and $\mathbf{Inn}(m)$ denotes the inner product of m-dimensional vectors on $\mathbf{Z}_q$. If we roughly estimate the prover (resp. verifier) complexity in proposal 1 is $5D\mathbf{SM}$ (resp. $D\mathbf{SM}$),

Table 3.3: Theoretical performance

Proposal 1	Complexity
Prover	$(5D + 4\log D)\mathbf{SM} + (5D + 4\log D)\mathbf{Ad} + 4D\mathbf{FO}$
Verifier	$(D + 2\log D)\mathbf{SM} + 2(\log D)\mathbf{Ad} + 2(\log D)\mathbf{FO}$
Proof size	$(1 + 2\log D)\mathbf{G} + 4\mathbf{F}_p$

Proposal 2	Complexity
Prover	$5m\mathbf{Inn}(D) + 4m(\log D)\mathbf{Inn}(m\log_p q) + 2mD\mathbf{FO}$
Verifier	$m\mathbf{Inn}(D) + 2m(D + \log D)\mathbf{FO}$
Proof size	$(1 + 2\log D)\mathbf{F}_q^m + 4\mathbf{F}_q$

where D is the degree of the polynomial. In fact, in our experiments, these parts occupied $99 \sim 98\%$ of the total processing times. These times correspond to the computation for $Com(f)$ and $\{C_1(i), C_2(i)\}$ (resp. $g(l+1)$). Also, the proof size is about $2\log D$ $\mathbf{G}$-elements. Similarly, the prover (resp. verifier) complexity in proposal 2 is about $5m\mathbf{Inn}(D)$ (resp. $m\mathbf{Inn}(D)$) and the proof size is about $2\log D$ $\mathbf{F}_q^m$-vectors. Thus, in proposal $1, 2$, the prover and verifier times linearly growth with the degree of the committed polynomial $f(X)$; also, the proof sizes logarithmically growth.

3.6.2 Evaluation of our polynomial commitment schemes

We implemented our proposed protocols $1, 2$ by using the NTL library [6] and the libsnark library [4], and performed the benchmarks of our protocols on a single thread of an Intel Core i7-9700K CPU @ 3.60 GHz with 64 GB memory.

Parameters

In the proposal 1, we chose BN128 elliptic curve, called the snark curve, as the discrete group. This discrete group has an order 254-bit prime number with 100-bit security.

In the proposal 2, we chose the plain text space $p = 2^{32} - 5$ pseudo Mersenne prime, modulus $q = 2^{1024} - 1$, and the commitment vector dimen-

Table 3.4: Experimental results of our Marlin without trusted setup

	Variables	Constraints	Setup	Index	Prover	Verifier	Proof size
SHA-256	25561	73568	84.5 s	19.5 s	338.7 s	35.2 s	26.3 KB
Toy	310	620	1.0 s	0.1 s	2.1 s	0.2 s	15.5 KB
Small	32820	65640	111.7 s	9.9 s	204.7 s	20.2 s	24.9 KB
Medium	262195	524390	874.4 s	76.3 s	1615.5 s	175.0 s	28.9 KB
Large	1048630	2097260	3472.8 s	309.3 s	5510.0 s	599.2 s	31.6 KB

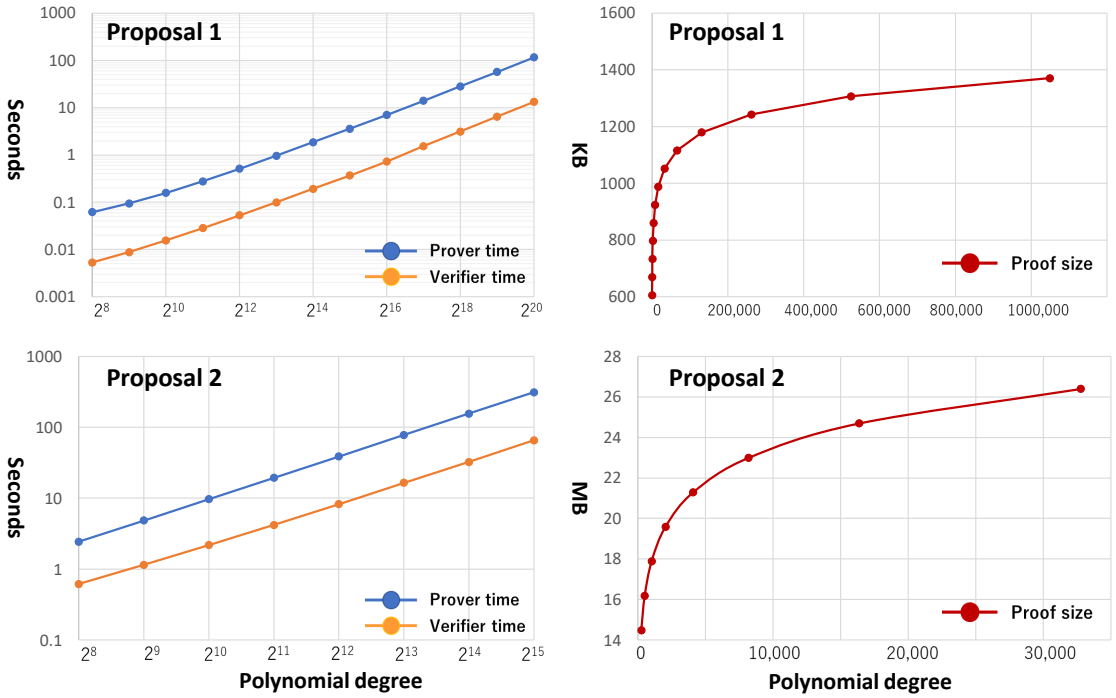

Figure 3.1: Experimental results of polynomial commitment schemes

sion $m = 7030$. Also, at this time, if the random vector $\mathbf{r}$ is taken from $\mathbf{F}_p^{2m \log_p q}$, the norm of committed vector increases by 64 bits in one convolution operation, so that we can evaluate 15 times convolutions and degree 2^{15} polynomials. It was computed in [68] that if the number of columns is very large, then one should solve SIS for a sub-matrix where the number of columns is $T \leq \sqrt{r \log q / \log \delta}$ for a constant δ related to lattice reduction. In our experiment, we take $\delta = 1.1$[1]. With this situation, we expect to find a vector of length

$$\min\{q, 2^{\sqrt{m \log q \log \delta}}\} \approx 2^{995} = B,$$

using BKZ reduction algorithms [50]. In this situation, all norms of our committed vector are bounded by $B/8$. Therefore, this parameter seems to be secure against BKZ reduction attacks with $\delta = 1.1$ in the dimension 7030.

[1]The constant δ is related to the block-size of BKZ reduction algorithm [50]. This algorithm is the currently best algorithm to solve SIS problem. Presently, the optimal lattice reduction set $\delta \approx 1.005$ in dimension 500. However, it is unclear what value this constant takes in the case of our higher dimension, such as 7030.

Results

Our experimental results are depicted in Fig. 3.1. We evaluated random polynomials degree $2^8, 2^9, \ldots, 2^{20}$, (resp. $2^8, \ldots, 2^{15}$) with proposed scheme 1 (resp. 2). As depicted in Fig. 3.1, the processing times of the prover and verifier increase almost linearly, and the proof size increases logarithmically. Our results show that the proof size of the proposal 1 is only 1.4 KB for one million ($= 2^{20}$) degree polynomials. Also, we evaluated batch-verifying time of proposal 1 for this degree 2^{20} polynomials; the time of batch-verifying 100 proofs was 27.9 seconds, and thus, each proof takes 279 msec. In addition, we also evaluated batch-verifying time of proposal 2 for degree 2^{15} polynomials, the time of batch-verifying 100 proofs was 78.6 seconds, and thus, each proof takes 786 msec.

3.6.3 Marlin without trusted setup

We implemented our polynomial commitment schemes to Marlin protocol [36] in C++. Here, we report the evaluation results of the discrete logarithm based scheme (see Section 3.6.4 for results of the lattice based scheme).

Furthermore, we have developed an adapter to use the libsnark library [4], which is a standard zero-knowledge proof library in the zk-SNARK or blockchain area. This adapter converts the libsnark's R1CS and gadgets to our Marlin type transparent ZKPs so that we can reuse various libsnark's R1CS and gadgets in our universal and transparent ZKPs.

Results

We implemented various circuits using the libsnark library. An R1CS for Marlin protocol has two parameters: the number of variables and constraints. Table 3.4 shows the setup, index, prover, and verifier times depend on the number of constraints (see Section 3.6.4 for more details). Also, the size of proof increase logarithmically. In Marlin protocol, the size of the proof is relatively large because it requires committing 10 polynomials, and their maximum degree is $6\times$(the number of constraints). In the SHA-256 circuit, it takes 84.5 seconds to setup a universal circuit that can evaluate up to 2^{17} constraints and 19.5 seconds to index it into the SHA-256 instance. Furthermore, the prover and verifier times are 338.7 and 35.2 seconds, and the size of proof is 26.3 KB.

3.6.4 Additional experimental data

Here, we report the additional experimental data.

Methodology of our experiment

We implemented our proposed schemes: the discrete log and lattice based universal and transparent ZKPs using our adapter in Section 3.6.3 between the libsnark library [4] and Marlin protocol [36]. More precisely, we implemented the zero-knowledge proof for the Bubble sort algorithm of N items (each item is a 32-bit integer) using the libsnark library's gadgets. We set the number of items as $N = 3, 7, \ldots, 4k + 3, \ldots, 151$ for the discrete log based one, and $N = 3, 4, 5, 6$ for the lattice based one. Unfortunately, the lattice (Ajtai commitments) based scheme requires a huge memory, so that $N = 6$ was the limit in our environment (that requires about 30 GB memory space). Therefore, it is described here as a reference value. We set that all parameters of our experiments here are as the same as in Section 3.6. Also, our experimental environment comprises an Intel Core i7-9700K CPU @ 3.60 GHz with a single thread.

Experimental results

We implemented and evaluated our schemes along with our methodology above.

At first, we choose the maximum numbers of variables and constraints for generating the universal SRS for Marlin system. Also, the constraints number and $\max\{||A||, ||B||, ||C||\}$ are roughly the same value, where $Az \circ Bz = Cz$ is a given R1CS instance. Furthermore, the number of variables and the dimension of the vector z are roughly the same value.

Next, we decide the maximum degree of the polynomial for our polynomial commitment scheme. This degree is roughly $6\times$(the maximum number of constraints). For example, if the maximum number of constraints is $100,000$, then we must setup a polynomial commitment scheme that can evaluate up to $600,000$ degree polynomials.

Figure 3.2 shows the performance evaluation of the discrete log based scheme. As depicted in Fig. 3.2, the Setup and Index times are completely dependent on the number of constraints (the yellow line). In our implementation, we take all number of parameter of valuables and constraints as powers of 2 for fast implementation. Therefore, our graphs become stair types. Also, the prover and verifier times depend on the number of constraints + the number of variables. Furthermore, the proof size logarithmically increases with these parameters. The zero-knowledge proof of Marin consists of 10 polynomial commitments and 3 evaluation points so that the proof size is about 10 times larger than a single polynomial commitment scheme. Table 3.5 shows the performance evaluation of the lattice based scheme. This scheme also shows the same behavior as in the case of the discrete log based, but the proof size is about $20,000$ times larger, and requires huge memory space. In table 3.5 examples, that require about 30 GB memory. Thus, it

is necessary to devise a memory reduction techniques in practical use.

Table 3.5: Universal and transparent ZKP from lattice

Variables	Constraints	Setup (s)	Index (s)	Prover (s)	Verifier (s)	Proof (MB)
132	342	216.6	1.7	112.3	25.5	391.4
264	684	434.0	3.3	225.1	48.6	425.1
440	1140	863.9	6.6	375.2	79.8	458.8
660	1710	871.9	6.6	446.7	94.4	458.8

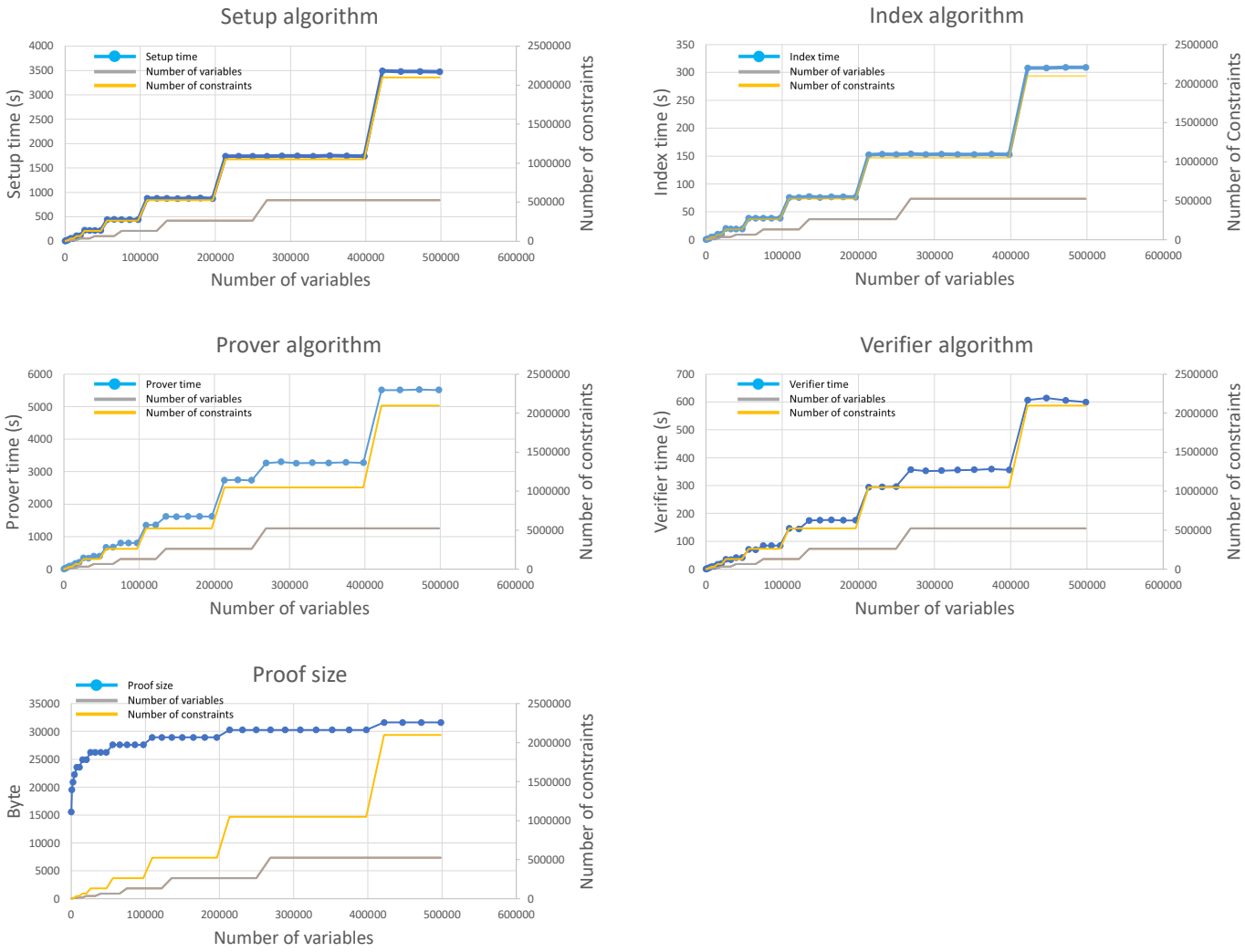

Figure 3.2: Universal and transparent ZKP from discrete log

Chapter 4

Secret Key Management Technology from Biometrics Fuzzy Signature

4.1 Summary

In recent years, cryptocurrencies such as Bitcoin [69] and Ethereum [2], and blockchain, which is a core technology thereof, have received considerable attention as new financial systems. One of the features of these blockchain-based cryptocurrencies is that they do not require a centralized authority such as a bank and can perform settlement processing with low fees. For example, in the Bitcoin system, the transaction minors collect transaction information in the P2P network, and after checking the validity of them, these transactions are put together in a block and written on the blockchain. At this time, the miner performs the approval process called Proof-of-Work (PoW), which is to calculate a hash value equal to or smaller than a specific threshold. PoW requires machine power for performing a large amount of hash calculation and thus has problems in transaction throughput and system scalability. For this reason, the use of permissioned / consortium type blockchains where only designated nodes to participate in the approval process is being considered in enterprises such as the financial area (See Hyperledger Fabric [3], Enterprise Ethereum Quorum [7] consensus algorithm). The validity of a transaction on the blockchain is guaranteed by a user attaching a digital signature based on a public-key cryptosystem, and a miner verifies it. For this reason, if the user loses or leaks the secret key for digital signature by losing the secret key management device or attacking by a malware, problems such as loss of assets on the blockchain and unauthorized use from impersonation occur. In this paper, we summarize the issues of secret key management in blockchain systems and propose a solution to these problems using biometrics-based digital signature technology, which gener-

ates a digital signature from use' s biometric information. In our proposed system, a secret key to be used for digital signature is generated from the user' s biometric information each time and immediately deleted from the memory after using it. Therefore, our blockchain system has the advantage that there is no need for storage for storing secret keys throughout the system. As a result, the user does not have a risk of losing the key management devices and can prevent attacks from malware that steals the secret key. We also show the results of implementing the proposed system on Hyperledger Fabric [3], verifying its effectiveness, and evaluating its performance.

4.2 Preliminary

4.2.1 Overview of blockchain systems

Blockchain is a general term for databases called distributed ledgers that are maintained by nodes on a P2P network. For example, Bitcoin blockchain [69] has ordered records called blocks in a chain-like data structure and is designed based on cryptographic technologies such as public-key cryptography, digital signatures, and hash functions. Blockchain technology is developed into various forms based on Bitcoin. The configuration of a blockchain network can be classified into a public type and a permissioned type as follows.

- Public blockchain:
  It is a blockchain in which an unspecified number of nodes freely participate. Bitcoin and Ethereum are classified as public types. Anyone can participate in the distributed ledger and transaction approval process. In addition, since there is a need to approve transactions among an unspecified number of untrusted participants, use a consensus algorithm such as Proof of Work (PoW) to prevent tampering by malicious participants (for details, see Bitcoin [69]).

- Permissioned blockchain:
  Only designated nodes are participating in the transaction approval process (ledger data may be disclosed outside the network). Hyperledger Fabric [3] and Enterprise Ethereum Quorum [7] are classified as this types.

In both types of blockchain systems, the management of secret-key is crucial because the digital signature using the secret-key is used for user authentication and generation of a transaction.

4.2.2 Biometrics-based fuzzy signature

Biometrics based fuzzy signature [82][67] is a digital signature technology based on biological information. This signature scheme corrects "fluctua-

tions" in biological information obtained from fingerprint sensors or biological sensors and extracts the unique user's secret keys. Of course, when the fluctuations are large, when coming from another person, a different key is output. This process is called a fuzzy extractor [43]. Also, the extracted secret key can be used to generate a digital signature based on a conventional public-key cryptosystem. A feature of this signature method is that a secret key is generated from a user's biometric information each time so that there is no need to store the secret key on a server or key management device. As a result, the user does not have a risk of losing the key management devices and can prevent attacks from malware that steals the secret key. In the conventional biometric authentication system, the biometric information and the secret key are stored in the key management server/device, and the secret key can be used if the image match rate with the biometric information obtained from the sensor is higher than the threshold. Therefore, note that a server/device that manages the secret key is required, and their storage is the attack point. We implemented a fuzzy extractor-type digital signature scheme based on the scheme in [82][67]. The user registration and signature generation processing are as follows.

- User registration:
  At the time of registration, the user inputs biometric information to the sensor and extracts a feature value from the biometric information. Next, the one-way conversion is performed using the PKI secret key and the feature value as input to generate a template data, which including a public key and system parameters. Finally, the user template data is stored in the authentication device. This user's template data is public information, and even if this information is leaked to an attacker, it is difficult to reconstruct the user's secret key or biometric information.

- Generating a digital signature:
  When generating a digital signature, the user inputs biometric information to the authentication sensor device and extracts a feature value. Next, the PKI secret key is reconstructed from the template and the feature value. The correct secret key is reconstructed only when the feature value at the time of registration and the feature value at the time of authentication are sufficiently close. After generating the secret-key, the user generates the digital signature for the transaction and removes the secret key from the memory, immediately. The generated signature can be verified with the ordinary PKI public key.

4.2.3 Security issues of secret-key management

In this section, we summarize related works and the conventional secret key management issues in the blockchain. The validity of a transaction on the

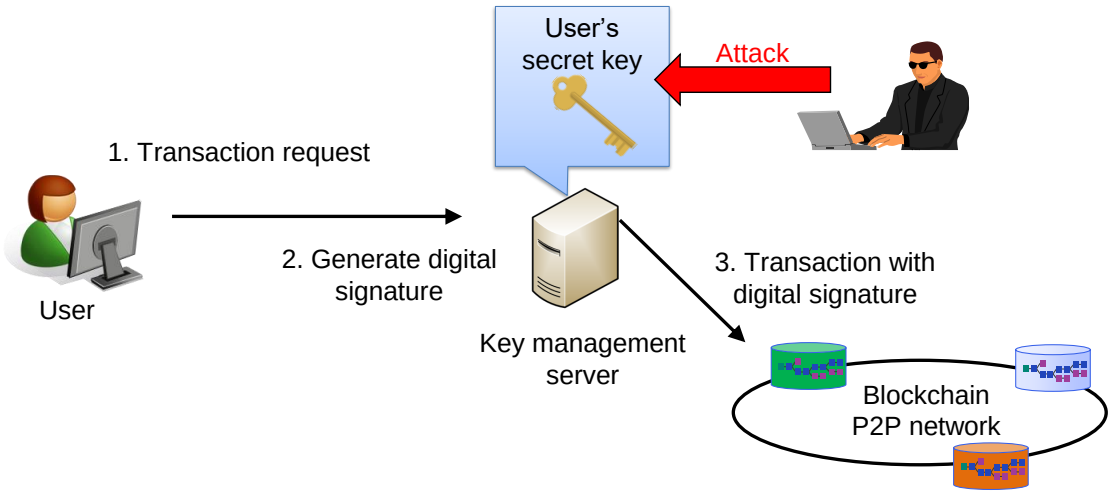

Figure 4.1: Key management system on the server-side

blockchain is ensured by a user attaching a digital signature and verifying the integrity by the miners. For this reason, if the user loses or leaks the secret key, there is a risk of loss of assets on the blockchain and damage to illegal transactions due to spoofing. Secret key management methods in the blockchain system are broadly divided into personal management type and server management type. There is a risk of loss of the management device in the personal management type. On the other hand, in server management type, there is a risk that the user's ID/password for logging in to the server is stolen or leak it due to hacking, such as unauthorized intrusion into the server. Hereinafter, typical key management methods will be described.

- User' s local storage (personal type):
  It is a method in which the secret key file is stored in the user's local storage. Since the user does not need to memorize the password, carry the device, and input information at all, the convenience and usability are high, but there is a high risk that the user terminal is infected with malware, and the secret key is leaked.

- Offline Secure Storage (personal type):
  There is a risk of loss or theft because the secret key is managed using a terminal such as a USB device that is not connected to the network or a two-dimensional code printed on paper. The use of secure devices, such as the Offline Hardware Security Module, is also conceivable, but the risk of loss or theft still exists.

- Hot/Cold Hosted Wallet (server type):
  This method is commonly used in cryptocurrency exchanges. Hot

Hosted Wallet (hereinafter referred to as Hot Wallet) is a method of storing a secret key in a server on a network. There is a risk that the secret key is leaked due to server hacking, malware infection, etc. Cold Hosted Wallet (hereinafter referred to as Cold Wallet) is a method of managing secret keys on a server separated from the network, and the risk of leakage is lower than that of Hot Wallet, but the convenience and usability is low. Both wallets pose a risk of spoofing when the user's login password is leaked. It is also assumed that the server administrator is a Trusted Third Party (TTP). This violates the philosophy of decentralization in blockchains.

For example, in the Hyperledger Fabric [3], a user's secret key is centrally managed by a Fabric SDK server (in the rest of this paper, referred to as a key server) (Hot / Cold Wallet type). As a result, there is a high possibility that this key server will be subject to hacking and spoofing attacks. Therefore, sufficient security measures must be taken (see Figure 4.1). In addition, in the cryptocurrency exchanges, the secret key is managed by the central server, but in recent years, a large number of cryptocurrencies have been stolen from multiple exchanges. Therefore, the security of this secret key management has become a social issue. Also, in a system in which security depends on TTP's server operation, it is difficult to completely prevent malicious server administrators from cheating, and a system in which security does not depend on server operations is desirable. As described above, the management of the secret key is a major issue in the blockchain industry, and there is no definitive measure at present. The essential problem of key management is that there is a storage for storing the secret key. In the next section, we propose a blockchain system that does not need to store a secret key somewhere by using the biometric-based fuzzy signature.

4.3 Proposal

In this section, we propose a novel key management technique using the biometrics-based fuzzy signature described in section 2.

4.3.1 What is the problem?

We describe the problems of the existing scheme in more detail. In the existing method described in the previous section, passwords and dedicated devices/servers are used to securely manage secret keys. However, since secret key data is managed by the device or server, so there are the risks of loss and malicious server administrator's attack. We think the fundamental problem of the secret key management is to store and manage the secret key data in the storage of the device or the server. We believe that it is desirable to remove the storage that stores the secret key from the system.

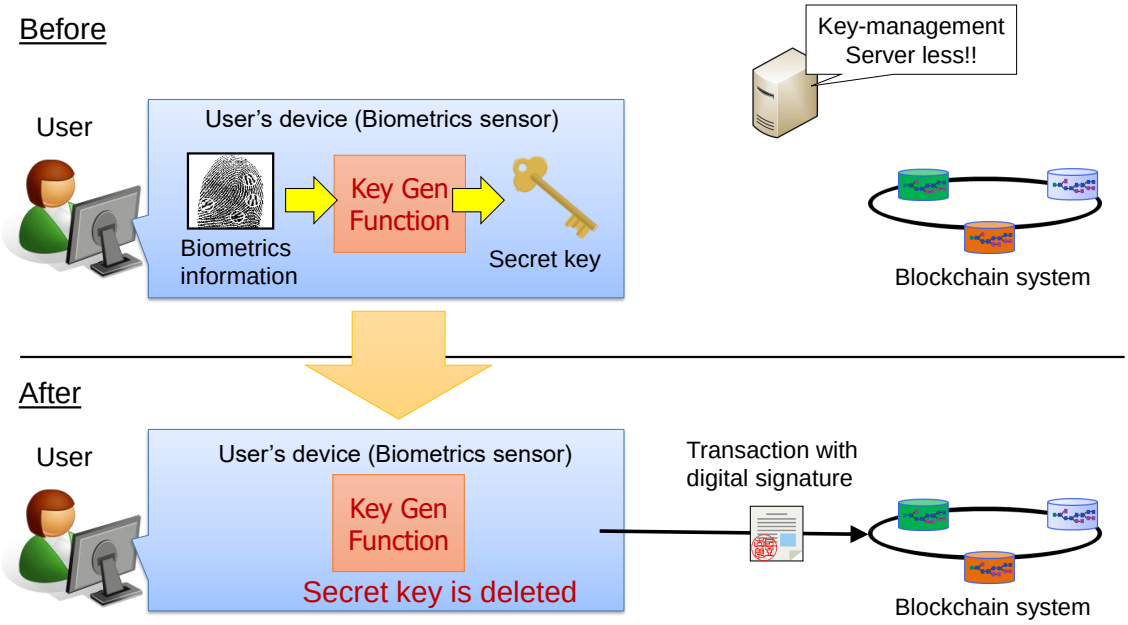

Figure 4.2: Image of our proposal

Furthermore, it is desirable a system in which the secret key exists in the memory only at the moment of generating the digital signature, and delete it from the memory after the signature is generated (see Fig. 4.2). In our proposed scheme, a user generates a secret key from biometric information each time a digital signature is generated. After the digital signature is generated, the secret key is deleted from the memory, thereby eliminating the storage for managing the secret key from the entire system, thereby enabling more secure operation.

4.3.2 Proposed scheme

In our proposed method, the secret key to be used for PKI based digital signature is generated each time from the biometric information of the user using the fuzzy signature method. Also, since the secret key is deleted from the memory after being used for signature generation, there is no need for storage for storing the secret key throughout the blockchain system. By utilizing this feature, it is possible to solve the problems of the secret key management described in the previous section. For example, if the secret key is managed by a personal wallet client application, replace the digital signature generation function with the digital signature generation function of the fuzzy signature, and the application calls it in each time. On the other hand, if the secret key is managed by the Hot/Cold Wallet on the server-side such as Hyperledger Fabric [3] or cryptocurrency exchanges, it is necessary to

modify a part of the server-side signature generation function. Specifically, it is necessary to remove the secret key management component of the server-side and implement an original API that gives a digital signature for the transaction at the client-side. Fig. 4.3 shows an image of the biometrics-based fuzzy signature compatible function implemented in the key server of Hyperledger Fabric or cryptocurrency exchanges. Our proposed protocol using the biometrics-based fuzzy signature consists of the following steps.

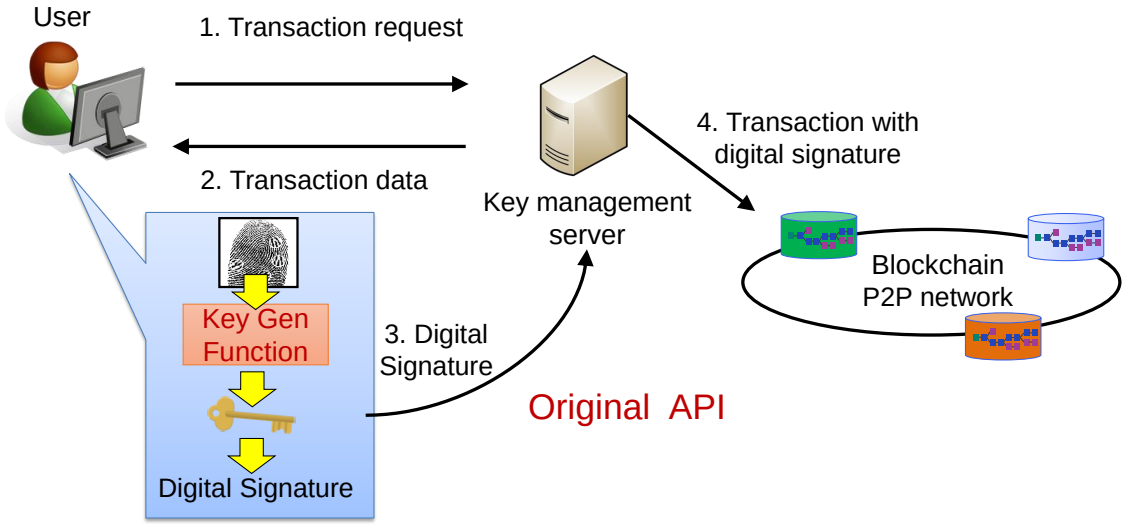

Figure 4.3: Our original API for biometrics-based signature

1. A user sends a transaction generation request to the key management server. Note that, in our proposed scheme, the key management server no longer manages the user' s secret keys.

2. The server generates the payload data of transaction that doesn't include the digital signature and returns it to the user.

3. The user extracts the secret key from biometric information using a biometric authentication sensor and the fuzzy signature's secret key generation engine. After that, the user generates the digital signature for the received transaction data. Finally, the user sends the digital signature value to the key management.

4. The key management server attaches the received digital signature to the transaction and sends it to the blockchain P2P network as a signed transaction.

In the above processing, the user immediately deletes the secret key from the memory of the biometric authentication device after executing step 3.

Next time, when generating a digital signature, the biometric information is input to the biometric authentication sensor again to generate the same secret key. Note that thanks to the fuzzy extractor, the same secret key is output to the user every time, and a different key is obtained to another user. It is important to generate/delete a secret key each time, and to perform operations that do not require a device that stores the secret key. As described above, even when the secret key is managed on the server-side, by partially modifying the system, the secret key management component can be removed, and the secret key management problem can be solved. In addition, by doing so, the risk that the malicious administrator illegally uses the secret key can be eliminated.

Table 4.1: Pros and Cons list of secret-key management

	Device Loss	Malware Attack	Malicious Administrator	Automatic Transaction
User' s local storage	N/A	N/A	Secure	Support
Offline secure storage	N/A	Secure	Secure	N/A
Hot/Cold wallet	Secure	N/A	N/A	Support
Our proposal	Secure	Secure	Secure	N/A

Table 4.1 summarizes the pros and cons list among the existing key management scheme and the proposed scheme. The proposed scheme eliminates the key management storage using the biometrics-based fuzzy signature and reduces the risk of device loss, and leakage due to malware infection or malicious hacker's attack. In addition, since a key management server is not required, it is possible to prevent an attack by a malicious server administrator, and it is suitable for decentralized philosophy. On the other hand, there is a problem in that it is not possible to deal with automatic transactions using smart contracts because the biometric information of the user is always required when generating a digital signature.

4.4 Experimental evaluation on practicality

We implemented our proposed scheme in Hyperledger Fabric [3] and evaluated the size of files and processing time. We also used a finger vein authentication sensor as a biometric sensor, and develop the fuzzy signature algorithm for finger-vein authentication in [82][67]. Moreover, we use the ECDSA 256 bit [58] as a digital signature algorithm in this evaluation. The ECDSA 256 bit is utilized in the open-source blockchain platform, the Hyperledger Fabric. We also implemented the original API for the key management server described in the previous section.

Table 4.2: Implementation results of signature schemes

Results		(Default) PKI	**Our proposal**
File size (byte)	Public key certificate /Template for signature	1 K byte	10 K byte
	Signature in transaction	71 byte	71 byte
Processing time (msec)	Generation of public template	-	499 msec
	Signature generation	78 msec	1306 msec
	Signature verification	70 msec	70 msec

Table 4.2 shows our experimental results. First, we evaluate the file size of a public key certificate, a public template for a finger vein based fuzzy signature, and a signature in a blockchain transaction. The public template for finger vein based fuzzy signature includes the ordinary public key certificate and some auxiliary information for fuzzy signature, and file size is equal to 10 K byte. This file size is larger than the ordinary public key certificate (1Kbyte). However, the 10 K byte is small enough for practical use. The file sizes of the signature are the same in all methods, and they are 71 bytes.

Second, we evaluate the processing time for each signature scheme. The CPU and memory where we perform the evaluation are Intel Celeron N3050 1.6 GHz and 4 GB, respectively. We evaluated in an environment with somewhat small resources, considering the use case in IoT devices or smartphones. A public template generation is executed one time in initial user registration. Thus the processing time 499 msec is fast enough. In our proposed scheme, the processing time of signature generation is 1306 msec, and this is slower than the PKI. However, the processing time is fast enough for practical use. Furthermore, the verification time of the signature is exactly the same. Although the signature generation algorithm requires some time, the proposed scheme is practical. Fig. 4.4 is a prototype of our proposed system, which is a finger vein based fuzzy signature type.

Figure 4.4: Implementation of biometrics-based fuzzy signature on Hyperledger Fabric

4.5 Conclusion and Future works

In this chapter, we describe the problems in managing the user's secret key in the blockchain on the user side and the server-side and proposed a novel key management technology using the biometrics-based fuzzy signature as our solution. In our proposed scheme, the user generates a secret key from biometric information using the fuzzy signature each time. Therefore, the secret key management component is virtually unnecessary. As a result, the risk of loss or leakage of the secret key can be reduced, and a more secure blockchain system can be realized. In addition, as a result of implementation evaluation, it was shown that the overhead is reasonable even in use cases, such as electronic payments in stores. On the other hand, it does not support automatic transactions such as M2M settlement or smart contracts because it requires biometric information. We want to address these issues in the future.

Chapter 5

Decentralized Netting Protocol over Permissioned Blockchain

5.1 Summary

In recent years, cryptocurrency including Bitcoin [69], Ethereum [2], and its core technology blockchain are attracting great attention as new financial settlement systems. The feature of these (public) blockchain based cryptocurrencies is that they do not require a centralized node such as a bank system, and can settle payment with a low fee. For example, Bitcoin miners collect transaction information on a P2P network and have confirmed their validity. After confirming validity, transactions are grouped into a block, and miners solve a cryptographic puzzle for called Proof-of-Work (PoW). Finally the authorized block is stored in the distribution ledger called the blockchain. Meanwhile, transaction consensus methods such as PoW adopted in public blockchains such as Bitcoin require machine power to perform a large amount of hash calculation, so there are problems in transaction throughput and system scalability. For this reason, private / permissioned-type blockchains that only specific nodes participate in transaction approval processing and solve these problems are now being considered for use in enterprises such as the financial field (see for example, private / permissioned-type blockchain Hyperledger Fabric [3], Enterprise Ethereum Quorum [7] consensus algorithm). In this paper, based on the Hyperledger Fabric v1.0 architecture or more general permissioned-type blockchain that equipped with the channel function, we propose a method to execute settlement with netting after remittances are made among multiple entities on the permissioned blockchain. As a feature of the proposed method, in view of decentralization and privacy of the permissioned type blockchain, it is a scheme point that conceals the amount of remittance and the calculation

butt of netting while excluding the central clearing node.

5.2 Contribution and related works

5.2.1 Our contribution

In this paper, we propose a secure netting protocol over permissioned type blockchain such as Hyperledger Fabric [3]. The word netting settlement is settlement means that pays only calculation results obtained by offsetting the amount of money on the books for a plurality of transactions. The meaning of targeted secure settlement in this paper is that the remittance amount and the name of sender and receiver entity are kept secret except for the sender and the receiver entity of the remittance transaction, and furthermore, in netting settlement, the calculated butt of each entity after offsetting is kept concealed by another entity. In addition, the meaning of de-centralized means that the entity participating in the settlement protocol executes processing to P2P and does not require a specific central server or other third parties.

5.2.2 Related works

Project Ubin [10] proposes three netting settlement methods using blockchains, such as proposed in this paper, and uses representative permissioned type blockchain OSSs Hyperledger Fabric [3], Enterprise Ethereum Quorum [7], and Corda [8] and evaluate it. The feature of their three netting settlement schemes proposed in this project is that it has the Gridlock resolution function. The Gridlock resolution means that there are a plurality of remittance instructions awaiting settlement due to insufficient balance, and furthermore, for a so-called ”stunting” state, which is waiting for payment from the other party, partial remission orders it is a process to eliminate the ”stunting” state by successfully combining (see the reference [10] for details). The netting settlement method proposed in this paper does not have the function of the Gridlock resolution. On the other hand, in the three proposal methods of [10], detailed analysis on security has not been performed, and in some methods, the value of the netting butt or its range leaks to other entities there. Also, correspondence when a malicious entity performs processing outside the protocol is out of scope (that is, a honest-but-curious model is assumed).

5.3 Preliminary

5.3.1 Outline of Hyperledger Fabric v1.0

The Hyperledger project is a project founded by the Linux Foundation for the purpose of developing a permissioned type blockchain OSS infrastructure that can be used in the enterprise. The project was established in February 2016, and more than 80 companies, including financial institutions and user companies and IT vendors, are participating in the project. In this paper, we describe a decentralized secure netting protocol using the channel function of Hyperledger Fabric v1.0 [3] developed and proposed by the Hyperledger project. This scheme can also be realized with a permissioned type blockchain having the channel function. In Hyperledger Fabric v 1.0, there is an access control function called channel that creates a group with only a specific node and shares ledger data, not all nodes participating in the blockchain network. This function is useful when we want to disclose the contents of transaction information only to stakeholders. Information such as which peer (node) belongs to the channel is shared by all peers in the network. The figure above shows that peer 1 belongs to ”red” and ”blue” channels, peer 2 belongs to ”red” and ”black” channels, peer n belongs to ”red” ”blue” and ”black” channel. As shown in the figure each transaction has a color of the channel, and transactions colored black are not sent to peer 1.

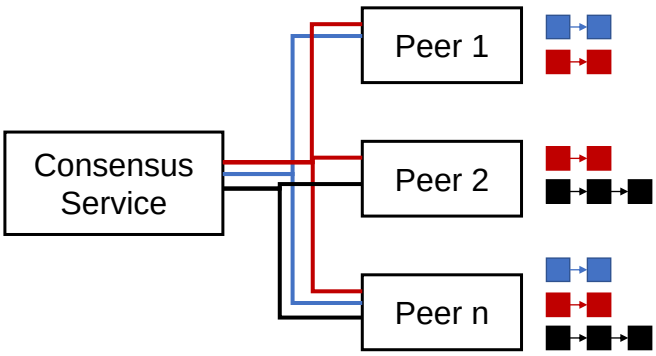

Figure 5.1: Channel structure of Hyperledger Fabric v1.0

5.3.2 Netting settlement

The netting settlement is, for example, a settlement of accounts after canceling interactive payment instructions occurring in a certain period between bank A and bank B.

The above figure is an example of netting settlement. The 50 payment orders from A bank to B bank and the 40 payment orders from B bank to A

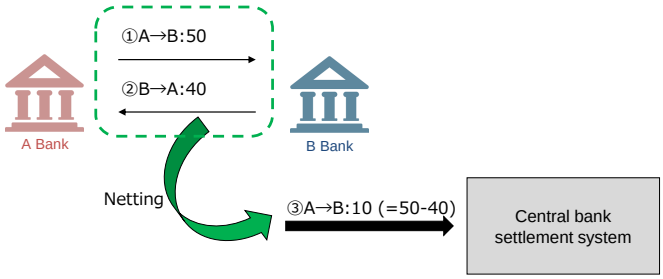

Figure 5.2: Bilateral netting

bank are offset on the book, and in fact, to the settlement system of central bank etc, A bank to B bank send only 10 payment instructions to the A bank. By reducing the fee for remittance by two payment orders into one as described above, it is effective to reduce the load on the system by further reducing the payment order to be input to the settlement system. The netting between two parties as shown in the above figure is called two-way netting or bilateral netting, and the netting settlement processing between three or more entities as shown below is called multilateral netting processing.

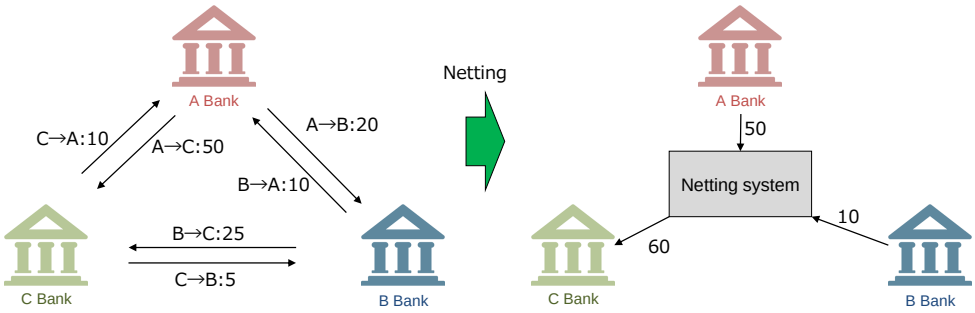

Figure 5.3: Multilateral netting

In this example, six payment orders (left side in the figure) occurred between A, B, and C banks are consolidated into three payment orders (on the right side of the figure) using a central netting system.

5.4 Proposal

In this section, we describe a decentralized settlement protocol using the channel function of Hyperledger Fabric. In this paper, the process of executing the netting settlement concerning remittance between four banks is exemplified, but the same argument applies to the more general case.

5.5 Conclusion and Future works

5.5.1 Channel structure

Let's suppose that the participating banks are four banks, A bank, B bank, C bank, and D bank, and each bank has one peer (node) in the Hyperledger Fabric system. Also prepare dedicated channels for peers between each two banks. For example, payment order transactions from bank A to bank B are stored only in the ledger within the channel between A and B. Hereinafter, the channel between A and B is called a bilateral channel between A-B. Since there are 4 participating banks, a total of 6 bilateral channels exist in this system.

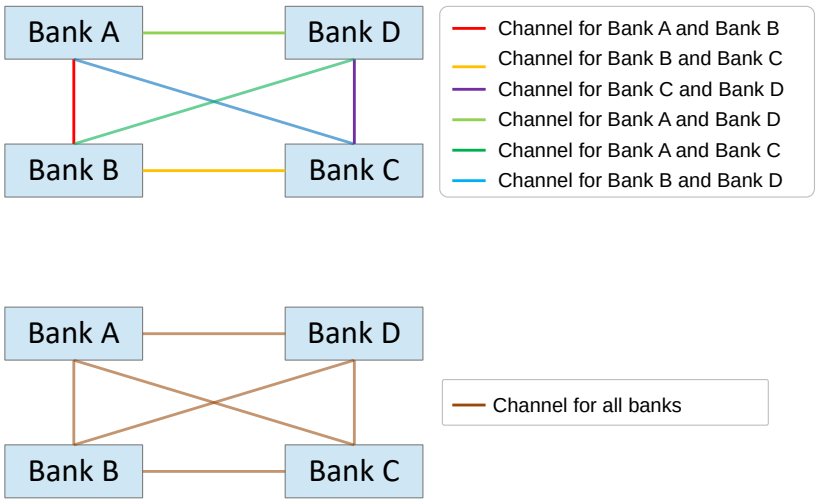

Figure 5.4: Our channel structure

Also, create one multilateral channel for sharing information among all banks.

5.5.2 Insecure netting protocol

Under the channel structure in the previous section, we describe a protocol that performs settlement in a distributed manner in a decentralized environment. This protocol is not secure, as netting calculation butt leak to other banks. In the next section, we describe the netting protocol which kept the netting calculation butt. The following table shows all remittance order transactions that occurred at a certain period.

As shown in the table, 8 transactions occurred at a time and the calculation of netting butt of 8 transactions is A bank +40, B bank −80, C bank +60, D bank +60. Transaction No. 1 is an order of payment 50 from

Table 5.1: Transaction table

Tx No.	A bank	B bank	C bank	D bank
1	-50	50		
2		-40	40	
3			-50	50
4	60			-60
5	-10		10	
6	40			-40
7		-60	60	
8		-30		30
NET	40	-80	60	-20

bank A to bank B. Since this transaction is transmitted only to the bilateral channel between A-B, so C and D bank do not know existence, as a result netting calculation butt A +40 knows only A bank. Similarly, other banks also know only the calculation butt of the netting result by themselves. Under this situation netting settlement is executed according to the following steps.

Insecure decentralized netting protocol

- **Step 0**:
  Any bank sends netting start transactions to the multilateral channel, and all participating banks agree. In doing so (by generating random numbers, etc.), the order of netting is determined, and the order is shared. Here, it is assumed that the order of A bank → B bank → C bank → D bank.

- **Step 1**:
  The A bank generates A → B: −40 (that is, a transaction paid by B bank to A bank 40) in order to offset its calculation butt +40 with 0 and sends it to the bilateral channel between A and B. As a result (because A bank got 40 from B bank), A bank's calculated butt becomes 0. On the other hand, since B bank paid 40, its calculated butt is from −80 to −40.

- **Step 2**:
  Likewise, the B bank generates B → C: 40 (that is, a transaction paid by the B bank to the C bank 40) in order to offset its calculation bottom −40 with 0, and sends it to the bilateral channel between the B-C. As a result (because B paid 40 to the C bank), the calculation butt of B bank will be 0. Meanwhile, C bank got 40, so the calculation butt will be 60 to 20.

- **Step 3**:
  Finally, the C bank generates C → D: −20 (that is, a transaction paying 20 from the D bank to the C bank) to offset its calculation butt +20 with 0, and sends it to the bilateral channel between C-D. As a result (because C got 20 from D bank), C bank's calculated butt becomes 0. Also, because D bank paid 20, the calculation butt will be from −20 to 0, and all netting process ends.

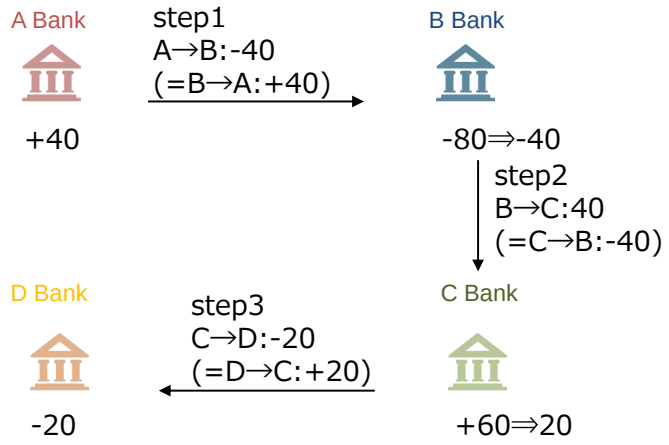

Figure 5.5: Insecure netting protocol

The above figure shows transactions generated in the netting protocols Step 1 to 3. In Step 1, A bank has sent a payment order of −40 to B bank in order to offset its own netting, but here it is leaking that bank A's deposit is +40 to B bank side . In addition, in Step 3, a payment order of −20 is sent from the C bank to the D bank, but this is also leaked to the C bank side that the D bank's calculation butt is −20. As a result, there is a problem that the start and end points of the netting route calculation butts are leaked out. The next section, how to solve this problem will be described.

5.5.3 Secure netting protocol

In this section, we show a solution for the problem of previous section' s netting protocol. A bank adds a random number to the calculated butt, thereby solving the problem that the calculated butt leaks. The structure of the channel and the transaction that occurred are the same as in the previous section.

Secure decentralized netting protocol

- **Step 0**:
  Any bank sends netting start transactions to the multilateral channel, and all participating banks agree. In doing so (by generating random numbers, etc.), the order of netting is determined, and all lines are shared. Here, it is assumed that the order of A bank → B bank → C bank → D bank.

- **Step 1**:
  The A bank generates a random number R and generates a transaction to pay A → B: $-40 - R$ in order to offset $+40 + R$ to 0 in addition to its own calculation butt sends it to the bilateral channel between A-B. As a result, the calculation butt of A bank is $-R$. On the other hand, since B bank paid $40 + R$, the calculated butt of B would be -80 to $-40 + R$.

- **Step 2**:
  Likewise, B bank generates a transaction to pay B → C: $40 - R$ in order to cancel its calculation butt $-40 + R$ to 0, and sends it to the bilateral channel between B and C. As a result, the calculation butt of bank B is zero. On the other hand, the calculation butt of C bank is 60 to $20 + R$.

- **Step 3**:
  Likewise, C bank generates a transaction to pay C → D: $-20 - R$ in order to offset its calculation butt $+20 + R$ to 0, and sends it to the bilateral channel between C and D. As a result, the calculation bottom of C bank is 0. In addition, the calculation bottom of D bank is R.

- **Step 4**:
  Finally, the D bank generates a transaction to pay D → A: $-R$ in order to offset its calculated butt R to 0, and sends it to the bilateral channel between A and D. As a result, the calculation butt of the D bank becomes 0, the calculation bottom of the A bank also becomes 0 and all netting process ends.

The bellow figure shows transactions generated in the netting protocols Step 1 to 4. Random number R is added to each transaction generated by the settlement protocol proposed in the previous section, and the A and D' s butts are concealed.

5.6 Conclusion and future works

In this paper, we propose decentralized netting protocols using the channel function of Hyperledger Fabric which is a permissioned type block chain OSS. In the proposed protocol, by using the channel function, the sender and

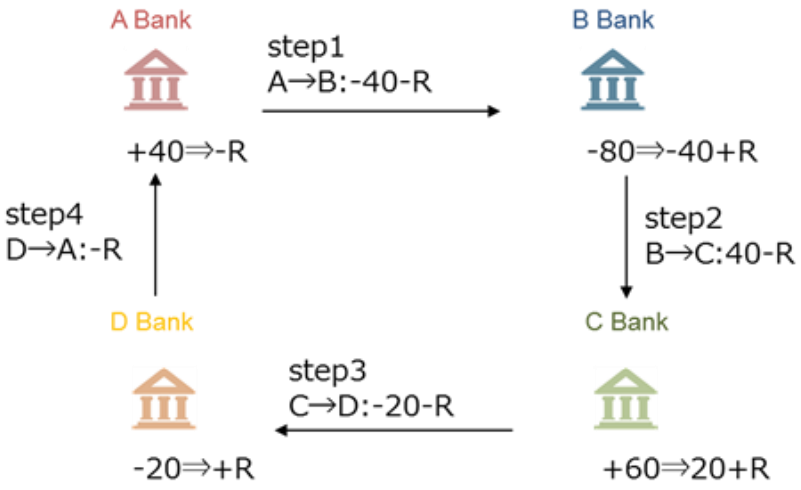

Figure 5.6: Secure netting protocol

receiver of the transaction information, the amount of money is concealed, and the multi-party type netting protocol using the random number is also used to conceal the netting calculation butt. In addition, the proposed method can execute netting settlement on P2P without a specific central organization such as a central server. Future works are as follows.

- **Gridlock resolution**:
  The proposed method does not have the gridlock resolution function, gridlock resolution function is necessary to reduce the overhead of settlement system and to save liquidity.

- **Range of random number** R:
  The proposed method uses random numbers, but obviously, the larger the range of random numbers is, the more secure it becomes. On the other hand, if the random number is large, there is a problem that the remittance amount increases. It is necessary to appropriately set the range of random number according to the use case.

- **Countermeasure against malicious nodes**:
  In this paper, in view point of the permissioned type blockchain, malicious nodes are supposed to be models that do not exist. We should consider the validity of this model and the correspondence to the case where malicious nodes exist.

Bibliography

[1] Coincheck: World's biggest ever digital currency 'theft'. https://www.bbc.com/news/world-asia-42845505.

[2] Ethereum. https://www.ethereum.org/.

[3] Hyperledger fabric. https://www.hyperledger.org/projects/fabric/.

[4] libsnark. https://github.com/scipr-lab/libsnark/.

[5] Monero. https://www.getmonero.org/.

[6] Ntl. https://www.shoup.net/ntl/.

[7] Quorum. https://www.goquorum.com/.

[8] R3 corda. https://docs.corda.net/.

[9] Torbjörn granlund and the gmp development team. gnu mp: The gnu multiple precision arithmetic library, 5.0.5 edition, 2012. http://gmplib.org/.

[10] Ubin project. https://www.mas.gov.sg/schemes-and-initiatives/Project-Ubin.

[11] Zcash protocol specification. https://github.com/zcash/zips/blob/master/protocol/protocol.pdf.

[12] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 99–108. ACM, 1996.

[13] Miklós Ajtai. Generating hard instances of the short basis problem. In Jirí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *Automata, Languages and Programming, 26th International Colloquium, ICALP'99, Prague, Czech Republic, July 11-15, 1999, Proceedings*, volume 1644 of *Lecture Notes in Computer Science*, pages 1–9. Springer, 1999.

[14] Martin R. Albrecht, Carlos Cid, Jean-Charles Faugère, and Ludovic Perret. Algebraic algorithms for lwe. Cryptology ePrint Archive, Report 2014/1018, 2014. `https://eprint.iacr.org/2014/1018`.

[15] Wojciech Banaszczyk. Inequalities for convex bodies and polar reciprocal lattices in r^n II: application of k-convexity. *Discrete & Computational Geometry*, 16(3):305–311, 1996.

[16] Carsten Baum, Jonathan Bootle, Andrea Cerulli, Rafaël del Pino, Jens Groth, and Vadim Lyubashevsky. Sub-linear lattice-based zero-knowledge arguments for arithmetic circuits. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, pages 669–699, 2018.

[17] Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. Everything provable is provable in zero-knowledge. In Shafi Goldwasser, editor, *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, volume 403 of *Lecture Notes in Computer Science*, pages 37–56. Springer, 1988.

[18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 14:1–14:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[19] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable zero knowledge with no trusted setup. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 701–732. Springer, 2019.

[20] 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, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 459–474, 2014.

[21] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. Snarks for C: verifying program executions succinctly

and in zero knowledge. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 90–108, 2013.

[22] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 103–128. Springer, 2019.

[23] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 781–796, 2014.

[24] Eli Ben-Sasson, Lior Goldberg, Swastik Kopparty, and Shubhangi Saraf. DEEP-FRI: sampling outside the box improves soundness. *CoRR*, abs/1903.12243, 2019.

[25] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 326–349, 2012.

[26] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, pages 315–333, 2013.

[27] Dan Boneh, Rosario Gennaro, Steven Goldfeder, and Sam Kim. A lattice-based universal thresholdizer for cryptographic systems. *IACR Cryptology ePrint Archive*, 2017:251, 2017.

[28] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Lattice-based snargs and their application to more efficient obfuscation. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part III*, pages 247–277, 2017.

[29] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Quasi-optimal snargs via linear multi-prover interactive proofs. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III*, pages 222–255, 2018.

[30] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 327–357. Springer, 2016.

[31] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 97–106, 2011.

[32] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Gregory Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 315–334, 2018.

[33] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent snarks from DARK compilers. *IACR Cryptology ePrint Archive*, 2019:1229, 2019.

[34] Matteo Campanelli, Rosario Gennaro, Steven Goldfeder, and Luca Nizzardo. Zero-knowledge contingent payments revisited: Attacks and payments for services. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 229–243, 2017.

[35] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1082–1090. ACM, 2019.

[36] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. Marlin: Preprocessing zksnarks with uni-

versal and updatable srs. Cryptology ePrint Archive, Report 2019/1047, 2019. https://eprint.iacr.org/2019/1047.

[37] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. *IACR Cryptology ePrint Archive*, 2019:1076, 2019.

[38] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, pages 3–33, 2016.

[39] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, pages 445–456, 1991.

[40] George Danezis, Cédric Fournet, Markulf Kohlweiss, and Bryan Parno. Pinocchio coin: building zerocoin from a succinct pairing-based proof system. In *PETShop'13, Proceedings of the 2013 ACM Workshop on Language Support for Privacy-Enhancing Technologies, Co-located with CCS 2013, November 4, 2013, Berlin, Germany*, pages 27–30, 2013.

[41] Hans Delfs and Helmut Knebl. *Introduction to Cryptography - Principles and Applications, Third Edition.* Information Security and Cryptography. Springer, 2015.

[42] Denise Demirel and Jean Lancrenon. How to securely prolong the computational bindingness of pedersen commitments. *IACR Cryptology ePrint Archive*, 2015:584, 2015.

[43] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[44] Léo Ducas and Daniele Micciancio. FHEW: bootstrapping homomorphic encryption in less than a second. In *Advances in Cryptology – EUROCRYPT*, pages 617–640, 2015.

[45] Léo Ducas and Phong Q. Nguyen. Faster gaussian lattice sampling using lazy floating-point arithmetic. In *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, pages 415–432, 2012.

[46] 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, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[47] Ariel Gabizon and Zachary J. Williamson. plookup: A simplified polynomial protocol for lookup tables. *IACR Cryptology ePrint Archive*, 2020:315, 2020.

[48] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *IACR Cryptology ePrint Archive*, 2019:953, 2019.

[49] Steven D Galbraith. Space-efficient variants of cryptosystems based on learning with errors. In *preprint*, 2013. `https://www.math.auckland.ac.nz/~sgal018/compact-LWE.pdf`.

[50] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 31–51. Springer, 2008.

[51] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 626–645, 2013.

[52] Rosario Gennaro, Michele Minelli, Anca Nitulescu, and Michele Orrù. Lattice-based zk-snarks from square span programs. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 556–573, 2018.

[53] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 197–206, 2008.

[54] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, 1991.

[55] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, pages 321–340, 2010.

[56] Jens Groth. On the size of pairing-based non-interactive arguments. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, pages 305–326, 2016.

[57] Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable snarks. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, pages 581–612, 2017.

[58] Don Johnson, Alfred Menezes, and Scott A. Vanstone. The elliptic curve digital signature algorithm (ECDSA). *Int. J. Inf. Sec.*, 1(1):36–63, 2001.

[59] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 177–194. Springer, 2010.

[60] Assimakis Kattis, Konstantin Panarin, and Alexander Vlasov. Redshift: Transparent snarks from list polynomial commitment iops. Cryptology ePrint Archive, Report 2019/1400, 2019. `https://eprint.iacr.org/2019/1400`.

[61] Aggelos Kiayias, Ioannis Konstantinou, Alexander Russell, Bernardo David, and Roman Oliynykov. A provably secure proof-of-stake blockchain protocol. *IACR Cryptology ePrint Archive*, 2016:889, 2016.

[62] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 723–732. ACM, 1992.

[63] Max Kubát. Virtual currency bitcoin in the scope of money definition and store of value. *Procedia Economics and Finance*, 30(30):409–416, 2015.

[64] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Lattice-based zero-knowledge arguments for integer relations. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, pages 700–732, 2018.

[65] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for lwe-based encryption. In *Topics in Cryptology - CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*, pages 319–339, 2011.

[66] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge snarks from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 2111–2128. ACM, 2019.

[67] Takahiro Matsuda, Kenta Takahashi, Takao Murakami, and Goichiro Hanaoka. Fuzzy signatures: Relaxing requirements and a new construction. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *Applied Cryptography and Network Security - 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings*, volume 9696 of *Lecture Notes in Computer Science*, pages 97–116. Springer, 2016.

[68] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In Johannes Buchmann Daniel J.Bernstein and Erik Dahmen, editors, *Post-quantum Cryptography,*, pages 147–191. Springer 2008, 2008.

[69] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[70] Anca Nitulescu. Lattice-based zero-knowledge snargs for arithmetic circuits. In *Progress in Cryptology - LATINCRYPT 2019 - 6th International Conference on Cryptology and Information Security in Latin America, Santiago de Chile, Chile, October 2-4, 2019, Proceedings*, pages 217–236, 2019.

[71] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 238–252, 2013.

[72] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1991.

[73] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, pages 89–114, 2019.

[74] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93, 2005.

[75] Jeff John Roberts and Nicolas Rapp. Exclusive: Nearly 4 million bitcoins lost forever, new study says. `https://fortune.com/2017/11/25/lost-bitcoins/`.

[76] Teppei Sato, Mitsuyoshi Imamura, and Kazumasa Omote. Survey on tracking of leaked nem by coincheck incident. *IEICE*, 118(30):35–41, 2018.

[77] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 1989.

[78] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.

[79] Srinath T. V. Setty. Spartan: Efficient and general-purpose zksnarks without trusted setup. *IACR Cryptology ePrint Archive*, 2019:550, 2019.

[80] Yannick Seurin. On the exact security of schnorr-type signatures in the random oracle model. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 554–571. Springer, 2012.

[81] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In Rainer Böhme and Tatsuaki Okamoto, editors, *Financial Cryptography and Data Security - 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*, volume 8975 of *Lecture Notes in Computer Science*, pages 507–527. Springer, 2015.

[82] Kenta Takahashi, Takahiro Matsuda, Takao Murakami, Goichiro Hanaoka, and Masakatsu Nishigaki. A signature scheme with a fuzzy private key. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *Applied Cryptography and Network Security - 13th International Conference, ACNS 2015, New York, NY, USA, June 2-5, 2015, Revised Selected Papers*, volume 9092 of *Lecture Notes in Computer Science*, pages 105–126. Springer, 2015.

[83] Riad S. Wahby, Ioanna Tzialla, Abhi Shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zksnarks without trusted setup. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 926–943. IEEE Computer Society, 2018.

[84] Jiaheng Zhang, Tiancheng Xie, Yupeng Zhang, and Dawn Song. Transparent polynomial delegation and its applications to zero knowledge proof. *IACR Cryptology ePrint Archive*, 2019:1482, 2019.

[85] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. Blockchain challenges and opportunities: a survey. *IJWGS*, 14(4):352–375, 2018.

Publication List

International Conference Paper:

- Ken Naganuma, Masayuki Yoshino, Hisayoshi Sato, Nishio Yamada, Takayuki Suzuki, and Noboru Kunihiro. Decentralized Netting Protocol over Consortium Blockchain. 2018 International Symposium on Information Theory and Its Applications (ISITA 2018), Singapore, October 28-31, 2018.
- Ken Naganuma, Masayuki Yoshino, Noboru Kunihiro, Atsuo Inoue, Yukinori Matsuoka and Mineaki Okazaki. Post-Quantum zk-SNARK for Arithmetic Circuits using QAPs. The 15th Asia Joint Conference on Information Security (AsiaJCIS 2020), Taipei, Taiwan, August 20-21, 2020.
- Ken Naganuma, Kenta Takahashi, Yousuke Kaga, Takayuki Suzuki, Masayuki Yoshino, and Noboru Kunihiro. New Secret Key Management Technology for Blockchains from Biometrics Fuzzy Signature. 15th Asia Joint Conference on Information Security (AsiaJCIS 2020), Taipei, Taiwan, August 20-21, 2020.

International Conference Poster:

- Ken Naganuma, Suzuki Takayuki, Kenta Takahashi, Yousuke Kaga, Masayuki Yoshino, and Noboru Kunihiro. Secure Key-management Technology for Blockchain using Biometrics Information. 2019 14th International Workshop on Security (IWSEC 2019), Tokyo, Japan, August 28-30, 2019. (Best poster award).

Domestic Conference Paper:

- Ken Naganuma, Takayuki Suzuki, Masayuki Yoshino, Hisayoshi Sato, and Kenta Takahashi. Decentralized netting protocol over Hyperledger Fabric. 2018, Symposium on Cryptography and Information Security (SCIS 2018), Niigata, Japan, January 23-26, 2018.
- Ken Naganuma, Takayuki Suzuki, Kenta Takahashi, Yosuke Kaga, Nishio Yamada, Noboru Kunihiro, and Masayuki Yoshino. Key management technology for blockchain using PBI. 2019 Symposium on

Cryptography and Information Security (SCIS 2019), Shiga, Japan, January 22-25, 2019.

- Ken Naganuma, Atsuo Inoue, Mineaki Okaszaki, Masayuki Yoshino, Anirban Basu, and Noboru Kunihiro. Post-quantum zk-SNARKs for Arithmetic Circuits. 2020 Symposium on Cryptography and Information Security (SCIS 2020), Kochi, Japan, January 28-31, 2020.