博士論文

Cryptographic Schemes Towards Transparent and Fair Digital Currency

Taishi HIGUCHI
樋口 大志

情報セキュリティ大学院大学
情報セキュリティ研究科
情報セキュリティ専攻

2025年09月

Cryptographic Schemes Towards Transparent and Fair Digital Currency

Taishi HIGUCHI

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

INSTITUTE OF INFORMATION SECURITY, JAPAN

September 2025

List of Publications

This thesis is based on the below publications.

A Peer-Reviewed Journal Article

[**HO25b**] Taishi Higuchi and Akira Otsuka. "Open-Cash: An Anonymous Electronic Cash Scheme with Open-Source Observers Based on BBS+ Signatures", *IEEE Access*, accepted (under preparation for publication).

Peer-Reviewed Papers in Proceedings of International Conference

[**HO24**] Higuchi Taishi and Akira Otsuka. "Electronic Cash with Open-Source Observers." International Conference on Financial Cryptography and Data Security. Cham: Springer Nature Switzerland, 2023.

[**HO25a**] Taishi Higuchi and Akira Otsuka, "Fair-Anonymity: A Novel Fairness Notion for Cryptocurrency," 18th International Conference on Network Security & Applications (CNSA 2025). CS & IT Conference Proceedings, 2025.

A Peer-Reviewed Journal Article (2nd Author)

[**THO22**] Takahashi, Taisei, Taishi Higuchi, and Akira Otsuka. "VeloCash: Anonymous Decentralized Probabilistic Micropayments With Transferability." IEEE Access 10 (2022): 93701-93730.

Abstract

Digital currencies face two persistent yet conflicting demands: preserving user anonymity and enforcing sufficient transparency to deter illicit activities. This dissertation explores these challenges through two core contributions: an open-source observer-based electronic cash framework ("Open-cash") and a fair anonymity protocol ("Fair-Anonymity") for public blockchains.

Open-cash extends conventional electronic cash (e-cash) by incorporating open-source observer programs running on tamper-proof secure processors. By enabling verifiable attestation while concealing individual transactions from the observer, the scheme achieves anonymity, one-more unforgeability, and double-spender traceability. Two main variants are presented: one based on blind signatures and another on BBS+ signatures, the latter offering simpler proofs and potential scalability benefits.

Fair-Anonymity shifts focus to public blockchains such as Bitcoin. It defines a novel fairness notion that ties the probability of payer identification strictly to the total transaction amount, independent of any subdivision into smaller denominations. The protocol instantiates this notion using a committed k-out-of-n oblivious transfer scheme and advanced proof techniques (e.g., One-out-of-Many proofs), ensuring that regulatory authorities can probabilistically trace large-value transactions, while honest users retain practical anonymity for routine payments.

Together, these contributions offer a balanced path forward for digital currencies: enhanced privacy for everyday usage, credible oversight for high-value payments, and cryptographically grounded mechanisms to meet contemporary regulatory requirements. By unifying secure hardware with modern cryptographic protocols, the presented approaches address critical demands of anonymity, auditability, and trust in the evolving digital economy.

Acknowledgment

First and foremost, I would like to express my heartfelt gratitude to my supervisor, Professor Akira Otsuka, for his dedicated and compassionate guidance throughout my studies at the Institute of Information Security in Japan. His support and mentorship have been invaluable throughout my research journey.

In 2020, I had the good fortune of being introduced to Professor Otsuka by one of my superiors at work. Initially, however, I had reservations about undertaking a research project. Having studied physics up to my master's degree and subsequently joining a computer company, I was a complete novice in information science and cryptography. Although I anticipated significant challenges, I trusted Professor Otsuka's leadership and believed in my own potential, which motivated me to embark on this journey.

I joined his laboratory to investigate blockchain security issues using contemporary cryptographic techniques. The beginning was far from easy, but Professor Otsuka patiently guided me through the fundamentals of cryptography, from zero-knowledge proofs to techniques such as Oblivious Transfer.

His thorough and considerate guidance ultimately allowed me to advance to the forefront of this research field. Achieving these results would have been extremely difficult without his unwavering support. Words alone cannot fully convey my gratitude for his dedication.

I also extend my sincere appreciation to the management team at Sakura Information Systems, as well as to my supportive superiors and colleagues, who have consistently encouraged and assisted me throughout this research.

Furthermore, I would like to express my appreciation to the creators of my hobbies, who provided comfort and motivation during difficult periods of my research. In particular, I am grateful to the brewers of the delightful sake "Wakamusume" and to

my friend, a talented illustrator, whose wonderful artworks always brought me joy and inspiration.

Finally, I wish to thank my parents, my sister, and my beloved friends. Their unwavering support, understanding, and encouragement have been an immense driving force behind my research.

Contents

List of Publications **iii**

Abstract **v**

Acknowledgment **vii**

Contents **ix**

List of Figures **xiii**

1 Introduction **1**

1.1 Introduction 1

1.1.1 Introduction for Open-cash 2

1.1.2 Introduction for Fair-Anonymity 4

1.1.3 Structure of the Thesis 5

2 Open-cash: Electronic Cash with Open-Source Observers based on Blind Signatures **7**

2.1 Preliminaries for Open-cash 8

2.1.1 Security Notions 8

2.1.2 Attested Execution Secure Processors 8

2.2 Blind Indirect Attestation 10

2.2.1 Overview 10

2.2.2 Definition of Blind Indirect Attestation 11

2.3 Construction of Open-cash 13

2.4 Security Proofs 21

2.4.1 Overview of Security Notions 21
2.4.2 Anonymity 21
2.4.3 One-More Unforgeability 22
2.4.4 Double-Spending Protection 23
2.5 Conclusion 24

3 Open-cash utilizing Zero-knowledge Proofs based on BBS+ Signatures 25
3.1 Main Contribution 25
3.2 Specification and Security Notion of our E-cash 26
3.2.1 Specification of Open-cash protocols 26
3.2.2 Definition of Security Notions 28
3.3 Background and Building Blocks 32
3.3.1 Background on Bilinear Maps 32
3.3.2 Cryptographic Assumptions 33
3.3.3 BBS+ Signature Scheme 34
3.3.4 Protocols for Proof of Knowledge 35
3.4 Construction of the Basic Open-Cash Scheme Based on BBS+ Signatures 36
3.4.1 Protocols 36
3.4.2 Security Proofs 44
3.5 Construction of Open-cash Scheme with BBS+ Observer 52
3.6 Conclusion 61

4 Fair-Anonymity: A Novel Fairness Notion for Cryptocurrency 63
4.1 k-out-of-n Committed Oblivious Transfer 65
4.1.1 Definitions of k-out-of-n Committed Oblivious Scheme 66
4.1.2 Construction of COT_n^k scheme 67
4.2 Fairness 70
4.2.1 Definition of Fairness 70
4.2.2 Exponential Saturation Function 71
4.3 Fair-Anonymity 72
4.3.1 Overview of Fair-Anonymity Protocols 72

4.3.2 System Setup 73
4.3.3 Registration Protocol 74
4.3.4 Execution Protocol 74
4.3.5 Fair-Tracing 77
4.4 Security Notions of Fair-Anonymity 77

5 Conclusion **83**

List of Figures

2.1 Attested Execution Secure Processors (AESP) 9
2.2 Opening protocol 16
2.3 Withdraw protocol 17
2.4 Spend protocol 18
2.5 The operation of the open-source Observers program in AESP 20

3.1 Open protocol 38
3.2 Withdraw protocol 40
3.3 Spend protocol 43
3.4 Open protocol with observer 53
3.5 Withdraw protocol with observer 55
3.6 Spend protocol with observer 60

4.1 Overview of k-out-of-n Commited Oblivious Transfer 66
4.2 Overview of Execution Protocol of Fair-Anonymity 75

Chapter 1

Introduction

1.1 Introduction

In recent years, the rapid expansion of digital currencies and advancements in blockchain technology have significantly enhanced convenience in cashless payments and facilitated fast, cost-effective transactions. These innovations offer substantial potential for promoting financial inclusion, especially among unbanked populations in developing regions. Moreover, public blockchains—distributed ledger systems characterized by decentralized management, transparency, and cryptographic immutability—have opened pathways for novel economic ecosystems through technologies such as smart contracts, non-fungible tokens (NFTs), and Soulbound Tokens (SBTs). Consequently, there is considerable interest and active research in these areas across academia, industry, and government initiatives.

Despite these advancements, significant privacy concerns persist, particularly with centrally issued digital currencies such as electronic money and Central Bank Digital Currencies (CBDCs). Unlike traditional cash, digital transactions allow issuers to collect extensive user payment data, often within opaque internal processes. Balancing the conflicting demands of user anonymity and regulatory compliance thus remains a pressing challenge. Concurrently, cryptocurrencies, celebrated within their communities for decentralization and strong privacy protections, face scrutiny due to misuse in ransomware attacks, money laundering, and terrorist financing, raising international concerns from an Anti-Money Laundering and Combating the Financing of Terrorism

(AML/CFT) perspective. Hence, there is an urgent need for cryptographic schemes that incentivize legitimate usage while effectively deterring criminal exploitation.

This thesis addresses these critical challenges by proposing two interrelated cryptographic solutions, each focusing on distinct aspects of digital currency systems:

- **Open-cash**: A centrally issued electronic cash system equipped with open-source observers to reconcile anonymity and regulatory oversight.
- **Fair-Anonymity**: A novel fairness framework for cryptocurrencies utilizing probabilistic traceability, where the likelihood of payer identification depends solely on transaction amounts.

In the subsequent sections, I provide detailed introductions to each research direction, highlighting their specific contexts, objectives, and contributions.

1.1.1 Introduction for Open-cash

Electronic cash (e-cash) systems ideally strive to achieve anonymity, unforgeability, and sufficient transparency to deter illicit activities. Following Chaum's pioneering RSA-based blind signatures aimed at user anonymity [Cha83; Cha85], numerous subsequent proposals have improved both efficiency and traceability [CFN90; Cha+90; OO92; Fer94a; Fer94b]. To further reconcile anonymity with regulatory compliance, concepts such as "wallets with observers" emerged [CP93; Bra94], wherein tamper-proof modules operate on behalf of banks or authorities, preserving user privacy.

Previous Works and Challenges

Brands [Bra93] proposed an elegant restrictive blind signature scheme embedding user identities directly; however, its one-more unforgeability remained unproven. Baldimtsi and Lysyanskaya [BL12; BL13] subsequently introduced a provably secure extension of Brands' approach based on the Pointcheval–Stern methodology [PS96]. Nevertheless, prior works did not offer concrete constructions of fully functional observers, particularly those supporting more flexible forms of oversight beyond simple double-spend prevention.

My Contributions in Open-cash

This thesis advances the state-of-the-art by introducing the following contributions within the Open-cash framework:

1. **Construction of Observers:** I significantly extend the Baldimtsi–Lysyanskaya blind signature framework, integrating active observers that not only detect but proactively prevent double spending. These observers can also be flexibly configured to provide additional forms of transparency beyond simple double-spending detection.

2. **Open-Source Observers:** I propose open-source observers running within tamper-proof Attested Execution Secure Processors (AESPs) [PST17], offering cryptographic proof of execution integrity. By providing publicly verifiable source code, these observers alleviate distrust, ensuring neither unauthorized surveillance nor misuse by central authorities.

3. **Blind Indirect Attestation (Novel Primitive):** I introduce a novel cryptographic primitive termed "Blind Indirect Attestation," which leverages secure hardware-based attestation (strong cryptographic signatures from secure elements) combined with sophisticated cryptographic methods (Brands' Blind Signature and BBS+ signature-based zero-knowledge proofs). This primitive allows verification of secure hardware attestations without revealing sensitive attestation keys or compromising user anonymity, thus enhancing both security and privacy.

4. **Two Realizations: Blind Signatures and BBS+:** I present two concrete instantiations of my Open-cash concept:

   - The *blind-signature-based* scheme ensures user anonymity during the withdrawal phase, achieves one-more unforgeability, and allows precise double-spender traceability.

   - The *BBS+ signature-based* scheme, grounded in bilinear pairings and the q-SDH assumption [BBS04; ASM06], employs zero-knowledge proofs at the spending phase to preserve anonymity. Its structure is notably simple, enabling efficient handling of proofs of possession of digital signatures (e.g.,

identity certificates), thus naturally extending to modern societal demands for greater transparency.

5. **Practical Feasibility:** Although primarily theoretical, my protocol design aligns well with capabilities available in contemporary smartphone secure elements (such as those in Android and iOS devices). While direct implementation benchmarks are beyond my scope, existing secure hardware clearly supports the performance requirements implied by my theoretical constructions, highlighting potential practical viability at scale.

1.1.2 Introduction for Fair-Anonymity

While centralized digital currencies may incorporate observer modules, decentralized cryptocurrencies like Bitcoin operate in an open, permissionless environment where no single authority can unilaterally impose transaction oversight. Yet their strong anonymity features raise legitimate concerns about money laundering and other criminal abuse.

Related Work

The tension between anonymity and oversight in public blockchains has long been recognized. Classic e-cash literature introduced systems with trustee-based tracing [BGK93], multi-trustee models, or specialized ledger designs [CLM07; Bla+11], but these approaches seldom integrate well with standard blockchain tokens such as Bitcoin. They also lack mechanisms to probabilistically identify high-value transfers in a manner robust to *split attacks*, where a large transaction is artificially split to avoid thresholds.

Challenges and My Solution

I identify two core problems:

- **Resisting split transactions:** Merely placing a threshold on single payments fails if users can freely divide a large sum into small chunks.
- **Stable anonymity for typical usage:** Everyday, low-value transactions merit high anonymity, whereas exceptionally large amounts likely deserve proportionally

higher scrutiny.

To address both, *Fair-Anonymity* proposes an *exponential saturation function* that assigns a probability of payer identification purely based on the total transaction amount. Even if the user splits the payment, the *collective* identification probability remains essentially the same. This property—called *fairness*—is realized through an extended k-out-of-n *Committed Oblivious Transfer* [Lai+18] combined with zero-knowledge proofs like One-out-of-Many proofs [GK15].

My Contribution of the Second Study "Fair-Anonymity"

1. **Formal Fairness Notion:** I introduce a new definition of fairness in tracing, ensuring that the total coin value alone determines trace probability, unaltered by subdivision.

2. **Committed k-out-of-n OT with ZK Proofs:** My protocol extends the efficient 2-round k-out-of-n oblivious transfer scheme [Lai+18], adding public verifiability to confirm the correct probability distribution of "ID vs. dummy" pieces without revealing the user's identity.

3. **Blockchain Compatibility:** Once integrated into a transaction, the resulting fairness proofs allow any public verifier to confirm correct usage. The authority can trace a user ID only with the designated probability, and not otherwise. Hence, *Fair-Anonymity* elegantly balances legitimate user privacy and AML demands for high-value transactions.

1.1.3 Structure of the Thesis

The remainder of this thesis is structured as follows:

- **Chapter 1 (Introduction):** Presents the motivation and objectives underlying my research, emphasizing the critical balance between anonymity and regulatory oversight in digital currency systems. It briefly outlines the two major contributions: Open-cash (for centrally issued digital currencies) and Fair-Anonymity (for decentralized cryptocurrencies).

- **Chapter 2 (Open-cash—Blind Signature-Based Scheme):** Provides an in-depth exploration of the Open-cash scheme. It begins by reviewing relevant cryptographic preliminaries, focusing particularly on blind signatures and the newly proposed Blind Indirect Attestation (BIA). It then details the Open-cash protocol construction, accompanied by rigorous security proofs.

- **Chapter 3 (Open-cash—BBS+ Signature-Based Observers):** Extends the discussion on Open-cash by examining an alternative instantiation based on BBS+ signatures. This chapter explains how zero-knowledge proofs utilizing bilinear pairings can effectively replace blind signatures, maintaining anonymity, enhancing traceability, and providing additional flexibility for transparency and oversight.

- **Chapter 4 (Fair-Anonymity—Traceability for Public Blockchains):** Introduces Fair-Anonymity, addressing anonymity and regulatory compliance in decentralized blockchain environments. The chapter defines the novel fairness notion, justifies the use of the exponential saturation function, and demonstrates how committed Oblivious Transfer combined with zero-knowledge proofs effectively enforces traceability probabilities based on transaction values.

- **Chapter 5 (Conclusions and Future Work):** Summarizes the key findings, identifies the limitations inherent in the current research, and outlines promising directions for future investigation toward practical and secure digital currency systems that balance privacy and oversight.

Chapter 2

Open-cash: Electronic Cash with Open-Source Observers based on Blind Signatures

In this chapter, I propose a novel concept of open-source observers which can achieve both cryptographic anonymity and highly flexible transparency at the same time.

Concretely, this work contributes on the following points:

1. I first show the construction of observers over the brands' type e-cash scheme proposed by Baldimtsi and Lysyanskaya [BL12; BL13].

2. I introduce the novel concept of "open-source observers" which makes the behavior of tamper-proof devices transparent, thus it can dispel the distrust against the authorities.

3. I introduce the notion and construction of Blind Indirect Attestation as the main primitive to achieve Open-cash, an e-cash scheme with open-source observers, assuming the existence of Attested Execution Secure Processors (AESPs) [PST17] and the standard assumptions such as discrete logarithm problems and random oracles.

4. The construction is efficient and almost feasible, as even modern smartphones equip with programmable secure processors.

2.1 Preliminaries for Open-cash

In this section, I give several security notions and introduce AESPs. I also propose the novel concept of Blind Indirect Attestation (BIA).

2.1.1 Security Notions

I define the *secure blind signature* as digital signatures which satisfy two properties - *blindness* and *one-more unforgeability*. The secure e-cash scheme with anonymity and one-more unforgeability can be constructed based on the secure blind signature.

Blindness

The blindness ensures that a signer can sign messages without knowing them. So a malicious signer in the blind signature scheme cannot link an $(m, \sigma(m))$ pair to any particular execution of the protocol. In e-cash protocols, this blindness can separate the link between coins withdrawn and spent. The property guarantees protection for user privacy.

One-More Unforgeability

$Unforgeability$ means that a correctly withdrawn coin is used only once, in other words, a coin can not be double-spent.
A user interacting with a signer S cannot output an additional, valid message/signature pair $(m, \sigma(m))$ no matter how many pairs of messages/ signatures of S he has seen (protection for the signer).

Definition 2.1 (Secure Blind Signature Scheme)**.** *A blind signature scheme is secure if the two properties of Blindness and One-more unforgeability are satisfied.*

2.1.2 Attested Execution Secure Processors

Attested Execution Secure Processor [PST17] (AESP) is a formal abstraction of modern trusted hardware. As shown in Fig. 2.1 they defined the ideal functionality of AESP as $\mathcal{G}_{att}$ which can install any program prog inside $\mathcal{G}_{att}$, acts like a modern secure element, when receiving a message "install". It executes the installed program

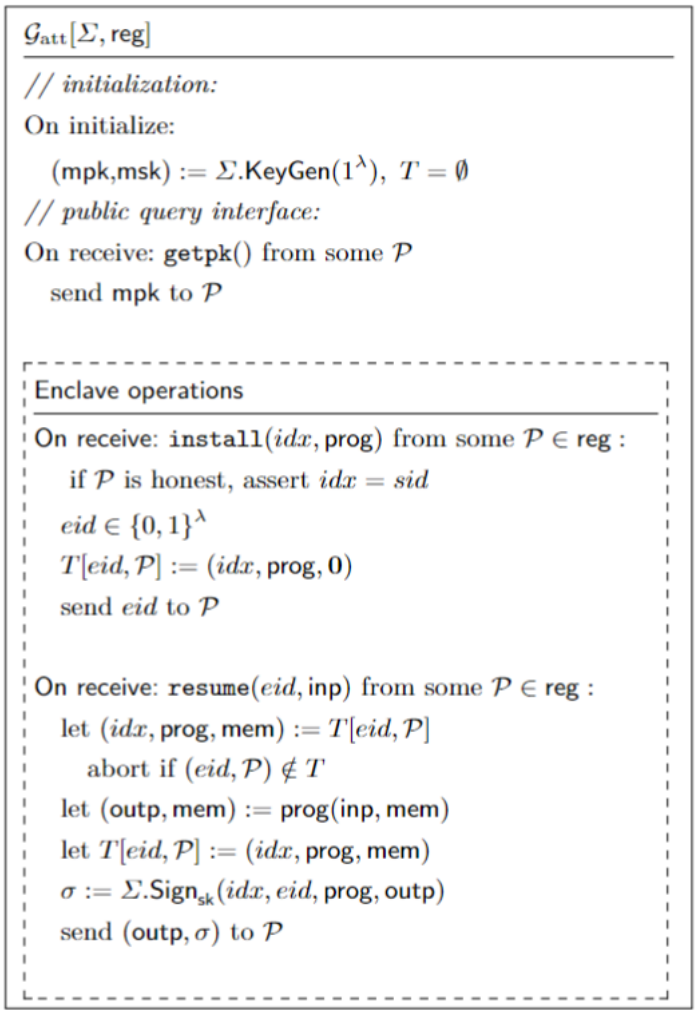

Figure 2.1: Attested Execution Secure Processors (AESP)

when receiving a message "resume". By calling resume, the program is called with the input inp, is run securely in the enclave and then outputs outp with an attestation $\sigma := \Sigma.\mathsf{Sign}_{\mathsf{sk}}(idx, eid, \mathsf{prog}, \mathsf{outp})$ with the signing key sk of $\mathcal{G}_{att}$. The attestation σ signed with a hardware key guarantees the result of calculations run there. In our Open-cash scheme, users will install an open-source observer program from the cloud in their AESPs. The observer program outputs the results with its attestation σ_{ATT} to prove that the Open-cash protocols, proposed in Section 2.3, were executed correctly. In our scheme, the attestation is verified indirectly using the Blind Indirect Attestation introduced in the next subsection 2.2, in order to keep users' private information secret.

2.2 Blind Indirect Attestation

In this section, I introduce a novel notion of *Blind Indirect Attestation (BIA)*, which is a variant of Direct Anonymous Attestation [CC04].

2.2.1 Overview

This can turn the direct attestation by AESP into the indirect one with blindness. In order to ensure that any open-source observer programs $Prog_{\mathcal{O}}$ (introduced in Section 2.3) correctly manage the anonymous payments, BIA enables the bank to certify indirectly that there exists an injective map to the withdraw view from the deposit view. The bank checks that the white-listed open-source program $Prog_{\mathcal{O}}$ is correctly installed in the withdrawer's AESP, thus every coin can be payable only once because $Prog_{\mathcal{O}}$ maintains the payment status of each coin. On the deposit protocol, the bank confirms that the coin is acceptable by verifying the Proof of Knowledge for satisfying the witness relation, thus any valid deposit of coins at the bank is ensured to be controlled by at least one of the set of $Prog_{\mathcal{O}}$.

Even if the indirect attestation is blinded its attestation ability remains the same as the original σ_{ATT} as long as the signature is a correct (not forged) one. The blind signature enables the attestation to be executed anonymously. This feature consistently enhances the security of our Open-cash scheme.

2.2.2 Definition of Blind Indirect Attestation

I give the novel notion of *Blind Indirect Attestation* which can turn the direct attestation by AESP into the indirect one with Blindness.

Non-Blind Indirect Attestation

Firstly we consider the indirect attestation without blindness for simplicity.

Definition 2.2 (Proof of Knowledge [Kog])**.** *An interactive proof system P, V for an NP relation $\mathcal{R}$ is a proof of knowledge with knowledge error ϵ, if there exists a probabilistic polynomial-time extractor E such that for every w and every probabilistic polynomial-time prover P^* :*

$$\Pr\left[(x,w) \in \mathcal{R} : w \leftarrow \mathcal{E}^{P^*}(x)\right] \geq \Pr\left[\langle P^*, V\rangle(x) = 1\right] - \epsilon$$

Definition 2.3 (Indirect Attestation)**.** *Suppose an enclave identified by eid running a program $Prog_i$ within a session identified by sid generates (x, w) that satisfies NP-relation $\mathcal{R}$, namely, $(x, w) \in \mathcal{R}$, and outputs a*

$$(sid, eid, Prog_i, x, \sigma_{ATT})$$

as an attestation signature on an AESP as assumed and stores the witness w in securely isolated storage inside the enclave in the AESP. If malicious prover P^* succeeded in convincing a verifier that $(x, w) \in \mathcal{R}$ with the same x as in the attestation signature that P^* knows the witness w, there must exist an extractor which can output the witness w with knowledge error probability $\epsilon > 0$. More formally we have

$$\Pr\left[(x,w) \in \mathcal{R} : w \leftarrow \mathcal{E}^{P^*}(x)\right] \geq \Pr\left[\langle P^*, V\rangle(x) = 1\right] - \epsilon.$$

Theorem 2.1. *Suppose we have EUF-CMA signature schemes and a Schnorr [Sch90] Proof of Knowledge protocol on a witness relation $\mathcal{R} = \{(x, w) \mid x = (g, g^w)$ where $g \in \mathbb{G}, w \in \mathbb{Z}_q^*$ and q is a prime number$\}$. Then there exists an indirect attestation under the Random Oracle model.*

Blind Indirect Attestation

Next, I introduce the blind version of the indirect attestation. We use a way similar to the well-known one by which several signature schemes can be turned into blind signature schemes [PS96].

Definition 2.4 (Blind Indirect Attestation)**.** *Suppose an enclave identified by eid running a program $Prog_i$ within a session identified by sid generates $(\widetilde{x}, \widetilde{w})$ that satisfies NP-relation $\mathcal{R}$, namely, $(\widetilde{x}, \widetilde{w}) \in \mathcal{R}$, and outputs a blinded statement x such that*

$$(sid, eid, Prog_i, x, \sigma_{ATT}).$$

as an attestation signature on an AESP as assumed and stores the witness $\widetilde{w}$ in securely isolated storage inside the enclave in the AESP.

Given a blind signature of the verifier σ_V on a message $\widetilde{x}$, that is,

$$(\widetilde{x}, \sigma_V),$$

then if malicious prover P^* succeeded in convincing a verifier that $(\widetilde{x}, \widetilde{w}) \in \mathcal{R}$ with the same $\widetilde{x}$ as in the attestation signature that P^* knows the witness $\widetilde{w}$, there must exist an extractor which can output the witness $\widetilde{w}$ with knowledge error probability $\epsilon > 0$. More formally we have

$$\Pr\left[(\widetilde{x}, \widetilde{w}) \in \mathcal{R} : \widetilde{w} \leftarrow \mathcal{E}^{P^*}(\widetilde{x})\right] \geq \Pr\left[\langle P^*, V\rangle(x) = 1\right] - \epsilon.$$

If the verifier is convinced the proof of knowledge with regard to the witness $\widetilde{w}$ that satisfies the witness relation $(\widetilde{x}, \widetilde{w}) \in \mathcal{R}$, the prover must be one of the enclaves that generated the witness $(\widetilde{x}, \widetilde{w})$ that some AESP had produced the attestation signatures on the $\widetilde{x}$ in the blinded form with overwhelming probability.

Theorem 2.2. *Suppose we have EUF-CMA signature schemes, secure blind signature schemes and Schnorr [Sch90] Proof of Knowledge protocol on a witness relation $\mathcal{R} = \{(\widetilde{x}, \widetilde{w}) \mid \widetilde{x} = (g, g^{\widetilde{w}})$ where $g \in \mathbb{G}, w, \widetilde{w} \in \mathbb{Z}_q^*$ and q is a prime number$\}$. Then there exists a blind indirect attestation under the Random Oracle model.*

Proof. First, we assume the signature $(\widetilde{x}, \sigma_V)$ corresponds to exactly one statement-witness pair $(x, w) \in \mathcal{R}$ and the verification algorithm outputs 1 on inputs (x, σ_V). This must be the case that the knowledge error probability of the indirect blind attestation is the same as the previous indirect attestation as stated in Theorem 2.1.

Therefore, we consider the probability that the verification algorithm outputs 1 but there is no corresponding statement x which appeared in one of the foregoing attestation signatures

$$(sid, eid, \mathrm{Prog}_i, x, \sigma_{\mathrm{ATT}})\,. \tag{2.1}$$

In the case the $\widetilde{x}$ is *forged* because the map Φ from the set of unblinded statements $\{\widetilde{x}\}$ to the set of blinded statements $\{x\}$, namely, $\Phi : \{\widetilde{x}\} \to \{x\}$, must be injective. The probability that $(\widetilde{x}, \sigma_V)$ is successfully forged is negligible by the One-more unforgeability property of the underlying secure blind signature.

□

Even if the indirect attestation is blinded its attestation ability keeps the same as the original σ_{ATT} as long as the signature is a correct (not forged) one. The blind signature enables the attestation to be executed anonymously. This feature consistently enhances the security of the Open-cash scheme.

2.3 Construction of Open-cash

In this section, I propose Open-cash protocols. The Open-cash is constructed over the Brands' type blind signature scheme by Baldimtsi and Lysyanskaya [BL12; BL13]. I expand the scheme by introducing observers into it and construct Open-cash protocols to achieve a secure open-source e-cash system, where the observer is an open-source program installed in AESP.

Overview of Protocols

The Open-cash consists of the following protocols - *Setup*, *Opening*, *Withdrawal*, *Spend*, and *Deposit*.

1. **Setup system:**

Setup is executed only once. A bank picks up parameters to create a public key used in all of the Open-cash protocols.

2. **Opening protocol:**

   When a user wants to open an account in the Open-cash system he firstly executes the opening protocol (shown in Fig.2.2) interacting with a bank. He should install the open-source observer program in his device distributed by the bank in advance. He and his observer pick secret parameters individually to generate a public key that serves as his account and sends it to the bank.

3. **Withdraw protocol:**

   In the withdraw protocol (shown in Fig.3.2), a user, the user's observer and a bank are going to run a blind signature under the shared public key so that the user can withdraw one coin signed by the bank. In this protocol, the observer picks up a random parameter as a secret key for the withdrawn coin and keeps it in its secure storage to prevent the user from double-spending the coin.

4. **Spend protocol:**

   In the spend protocol (shown in Fig.3.3), a user can spend his withdrawn coin at a shop with the shop id. When a user wants to pay his coin, the shop firstly calculates a temporary id and sends it to the user. Then, the user blinds it with the parameters he only knows and sends it to his observer. The observer in the enclave of the AESP checks if the parameter corresponding to the coin to be spent is still in the secure storage, then outputs the results with its attestation. The user is going to sign the temporary id by sending it to the bank along with the coin. The shop executes the verifications and if the verifications hold, it accepts the payment and stores the payment transcript.

5. **Deposit protocol:**

   A shop should execute the deposit protocol if it wants to deposit a coin. He should send the payment transcript to the bank along with the time of the transaction. Then the bank checks the temporary id and the time in order to ensure that the shop id encoded in the temporary id corresponds to the shop's one and the shop does not try to deposit the same coin again. If something is wrong the bank

doesn't accept the coin. After that, the bank verifies the signature on the coin and executes the verification equation.

The operations of the open-source observer are shown both in the diagrams of the protocols and in Fig. 2.5. is used to keep our scheme secure: In the opening protocol, the execution of the generation of the statement-witness pair and the registration of the statement is over the Indirect Attestation with σ_{ATT}. In the withdraw protocol, firstly the verification for the satisfaction of the NP-relation $(o_1; A_{\mathcal{O}})$ is executed by the PoK over the Indirect Attestation, and if the relation is satisfied, a blind signature (i.e. a coin) is issued with the Blind Indirect Attestation. In the spend protocol, the PoK for the witness of the user-id and the coin is executed over the Blind Indirect Attestation.

Setup System

Firstly a tuple (g_2, G_1, G_2) is generated at random, and a number $\boldsymbol{x} \leftarrow \mathbb{Z}_q$. A Bank $\mathcal{B}$ picks up two parameters $w_1, w_2 \leftarrow \mathbb{Z}_q$, and calculates $H = G_1^{w_1} G_2^{w_2}$. The pair (G_1, G_2, H) is a public key.

Opening Protocol

When a user $\mathcal{U}$ wants to open an account he installs an open-source Observers program distributed by the Bank in the AESP of his device such as a smartphone. After that, as shown in Fig.2.2, $\mathcal{O}$ in the enclave of the AESP picks up at random $o_1 \leftarrow \mathbb{Z}_q$ inside, stores it in secure isolated storage, and outputs $A_{\mathcal{O}} = G_1^{o_1}$ with the attestation σ_{ATT}. $\mathcal{U}$ generates at random a secret number $U \leftarrow \mathbb{Z}_q$ as his identity, computes $g_1 = A_{\mathcal{O}}(G_1^U G_2)$ as a public key. $\mathcal{U}$ sends g_1 and the attestation σ_{ATT} along with his identification (e.g. a passport) to $\mathcal{B}$ while keeping U secret. After the verification, $\mathcal{B}$ stores the identifying information of $\mathcal{U}$ in the account database together with g_1. $\mathcal{B}$ calculates $h = g_1^{w_1} g_2^{w_3}$ from a random number $w_3 \leftarrow \mathbb{Z}_q$ and sends it to $\mathcal{U}$. $\mathcal{U}$ stores $(g_1, A_{\mathcal{O}}, h; U)$.

Withdrawal Protocol

In the withdraw protocol, $\mathcal{U}$ and $\mathcal{B}$ are going to run a blind signature scheme, under the shared instance $((H, G_1, G_2), (h, g_1, g_2))$ so that the user can withdraw one coin signed by $\mathcal{B}$.

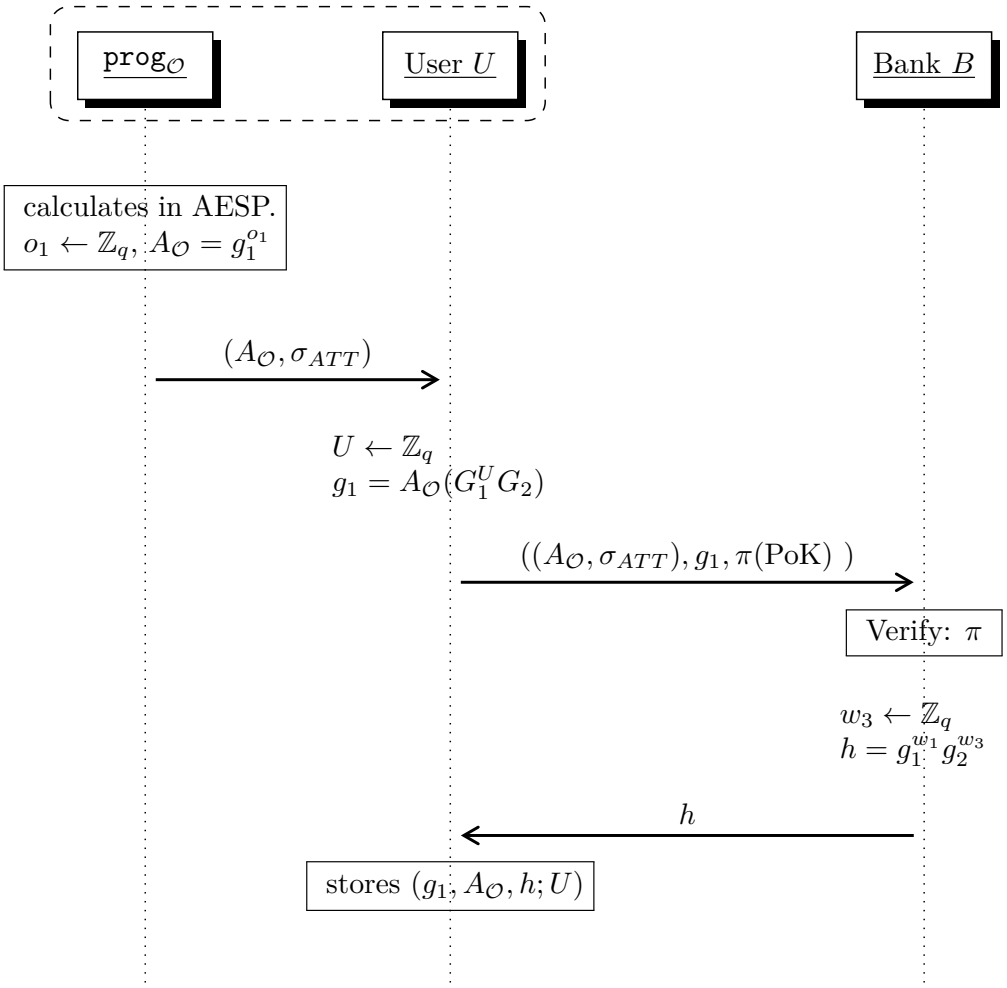

Figure 2.2: Opening protocol

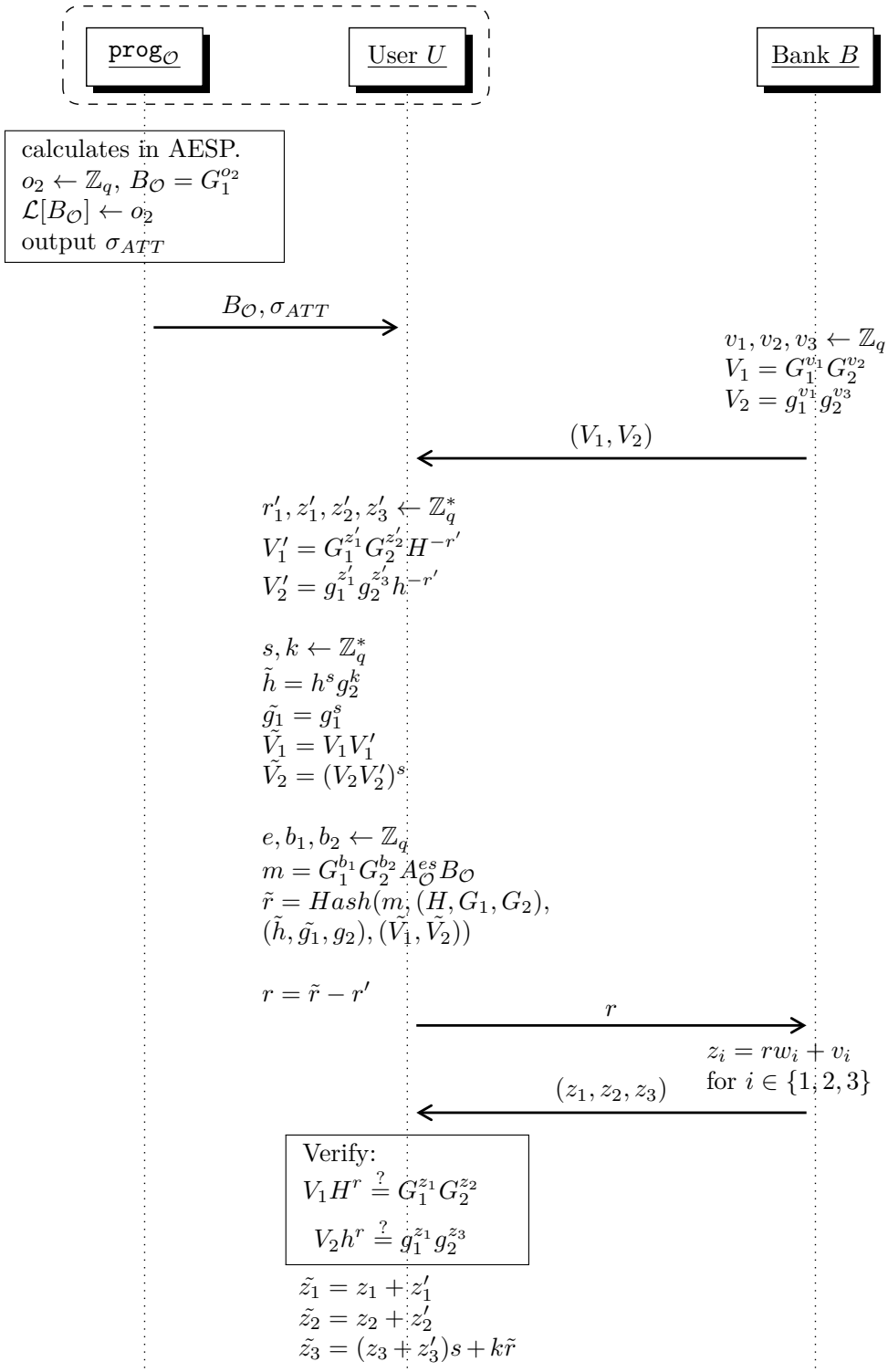

Figure 2.3: Withdraw protocol

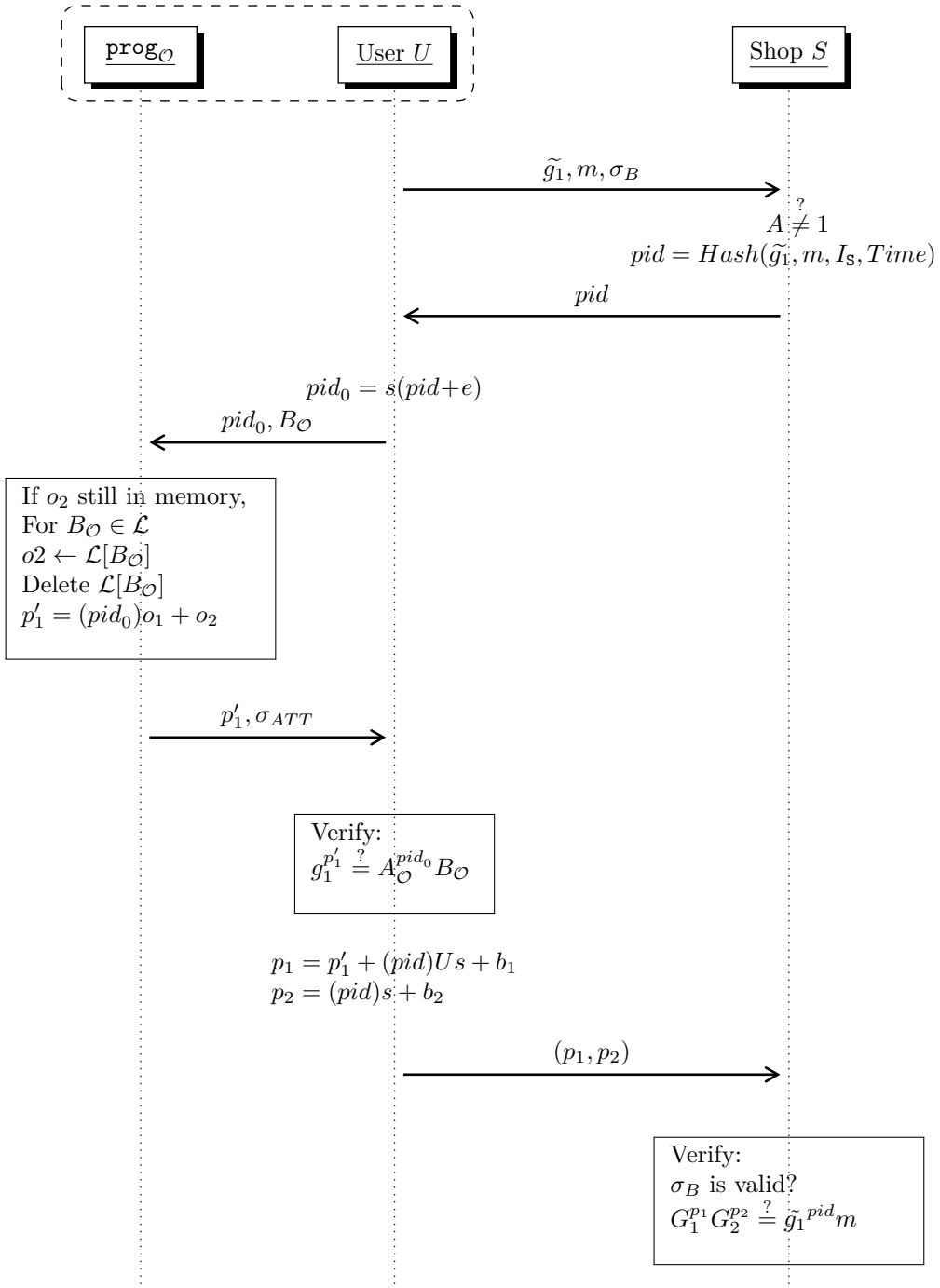

Figure 2.4: Spend protocol

As shown in Fig. 3.2, the process goes on as follows: First of all, $\mathcal{O}$ in the enclave of the AESP picks up at random $o_2 \leftarrow \mathbb{Z}_q$ inside and stores it in secure isolated storage and outputs $B_{\mathcal{O}} = G_1^{o_2}$ with the attestation σ_{ATT}. $\mathcal{B}$ picks $(v_1, v_2, v_3) \leftarrow \mathbb{Z}_q$ uniformly at random, calculates $V_1 = G_1^{v_1} G_2^{v_2}$ and $V_2 = g_1^{v_1} g_2^{v_3}$ and then sends them to $\mathcal{U}$. After $\mathcal{U}$ receives them, he selects uniformly at random $(r', z_1', z_2', z_3') \leftarrow \mathbb{Z}_q$ and computes the simulated values $V_1' = G_1^{z_1'} G_2^{z_2'} H^{-r'}$ and $V_2' = g_1^{z_1'} g_2^{z_3'} h^{-r'}$. Then, $\mathcal{U}$ selects random parameters $(s, k) \leftarrow \mathbb{Z}_q$ uniformly and computes:

$$\tilde{h} = h^s g_2^k, \quad \tilde{g}_1 = g_1^s, \quad \tilde{V}_1 = V_1 V_1', \quad \tilde{V}_2 = (V_2 V_2')^s$$

Note that $\tilde{g_1} = G_1^{Us} G_2^s$, where we can write $W_1 = Us$ and $W_2 = s$ and the User is actually proving that he knows a witness (W_1, W_2) such that $\tilde{g}_1 = G_1^{W_1} G_2^{W_2}$. Next U picks $e, b_1, b_2 \leftarrow \mathbb{Z}_q$ and calculates $m = G_1^{b_1} G_2^{b_2} A_{\mathcal{O}}^{es} B_{\mathcal{O}}$ and the hash

$$\tilde{r} = Hash\left(m, (H, G_1, G_2), \left(\tilde{h}, \tilde{g}_1, g_2\right), \left(\tilde{V}_1, \tilde{V}_2\right)\right)$$

and sends them to $\mathcal{B}$ the blinded value $r = \tilde{r} - r'$. At the end, $\mathcal{B}$ computes:

$$z_i = rw_i + v_1 \text{ for } i \in \{1, \ldots, 3\}$$

and sends (z_1, z_2, z_3) to $\mathcal{U}$.

Spend Protocol

In the spend protocol (shown in Fig.3.3), $\mathcal{U}$ can spend his withdrawn coin at a shop $\mathcal{S}$ with the shop id I_s. Firstly, $\mathcal{S}$ calculates the hash

$$pid = Hash\left(\widetilde{g_1}, m, I_s, \text{Time}\right).$$

and sends it to $\mathcal{U}$. Then, $\mathcal{U}$ blinds it using (s, e) and sends it to $\mathcal{O}$. $\mathcal{O}$ in the enclave of the AESP checks if o_2 is still in storage, then for the $(o_2, B_{\mathcal{O}})$ corresponding to the coin it computes $p' = (pid_0)o_1 + o_2$ and outputs it along with the attestation σ_{ATT}. $\mathcal{U}$

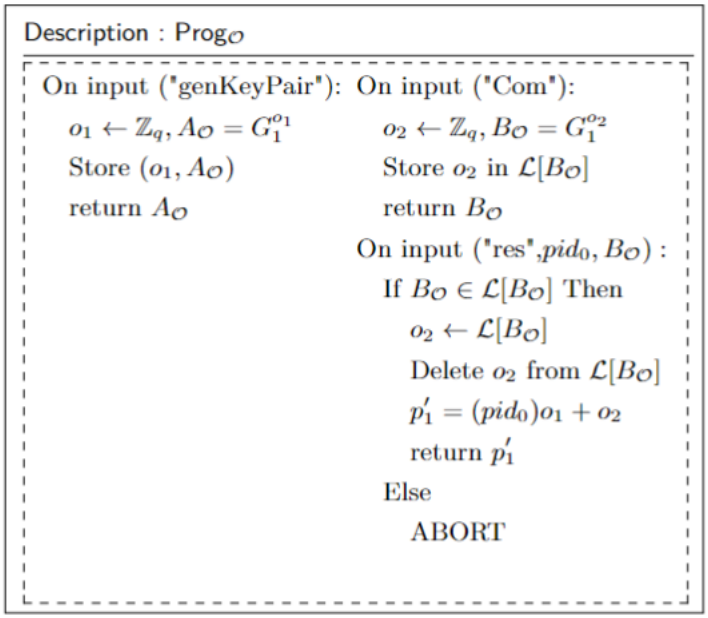

Figure 2.5: The operation of the open-source Observers program in AESP

is going to sign *pid* by sending to the bank:

$$p_1 = p_1' + (pid)Us + b_1$$
$$p_2 = (pid)s + b_2$$

along with the coin. The Shop verifies the validity of the coin and checks whether:

$$G_1^{p_1} G_2^{p_2} \stackrel{?}{=} m\tilde{g_1}^{pid}.$$

If the two verifications hold, $\mathcal{S}$ accepts the payment and stores $((m, \tilde{g_1}), \sigma_B, (p_1, p_2), pid)$.

Deposit Protocol

If $\mathcal{S}$ wants to deposit a coin, he should send the spending transcript, consisting of $((m, \tilde{g_1}), \sigma_B, (p_1, p_2), pid)$ to B along with the Time of the transaction. Then $\mathcal{B}$ checks (*pid*, Time) in order to ensure that the I_s encoded in the *pid* corresponds to $\mathcal{S}$'s one and $\mathcal{S}$ does not try to deposit the same coin again. If something is wrong the bank does not accept the coin. After that, $\mathcal{B}$ verifies the signature on the coin and that (p_1, p_2) is valid by checking whether $m\tilde{g_1}^{pid} \stackrel{?}{=} G_1^{p_1} G_2^{p_2}$. If so, B stores $((m, \tilde{g_1}), \sigma_B, (p_1, p_2), pid)$ if something does not verify, $\mathcal{B}$ rejects.

2.4 Security Proofs

The Open-cash scheme satisfies the security properties of anonymity and one-more unforgeability. It can also prevent double-spending attacks.

2.4.1 Overview of Security Notions

- **Anonymity**: The Open-cash scheme ensures that banks and shops can not know the private information about users from the coins on the withdraw and spend protocol. Moreover, the views of the coins withdrawn and spent are separated.

- **One-More Unforgeability**: Malicious users can not forge a coin, i.e., $l+1$ coins never exist for the correctly withdrawn l coins.

- **Double-Spending Prevention and Double-Spender Traceability**: Assuming AESPs are not compromised, the probability that a user can execute double-spending successfully is negligible. Moreover, even if an AESP is tampered with and double-spending is executed by an attacker, the bank can extract the double-spender's id with the observer from the two payment transcripts for the double-spent coin.

2.4.2 Anonymity

The ideal e-cash system should be anonymous like real money, in other words, it should satisfy anonymity. In the blind signature scheme, a signer can not distinguish messages if the signature scheme satisfies blindness. Because the withdrawn coins are signatures signed by a signer (i.e. a bank), in an e-cash scheme based on a blind signature scheme, he can not know the information about the message including the identity of the user withdrawing coins.
So e-cash schemes over secure blind signatures satisfy anonymity.

Lemma 2.1. *E-cash schemes constructed with secure blind signatures satisfy anonymity.*

The Brands' type scheme [BL12] satisfies anonymity, and we prove that the extension by introducing observers into the scheme does not affect the anonymity property:

Theorem 2.3. *The Open-cash scheme satisfies anonymity.*

Proof. We prove that our extension does not compromise its original anonymity. In our protocol, the variables (o_1, o_2) are introduced in order to construct observers. Using them, we calculated $A_{\mathcal{O}} = G_1^{o_1}$, $B_{\mathcal{O}} = G_1^{o_2}$, $g_1 = G_1^U G_2 A_{\mathcal{O}} = G_1^{U+o_1} G_2$ and $m = G_1^{b_1} G_2^{b_2} A_{\mathcal{O}}^{es} B_{\mathcal{O}}$.

We can prove that our scheme includes the original one as a special case; by letting $o_1 = 0$ and $o_2 = 0$, then the variables in the Open-cash protocols are

$$A_{\mathcal{O}} = 1$$

$$B_{\mathcal{O}} = 1$$

$$g_1 = G_1^U G_2$$

$$m = G_1^{b_1} G_2^{b_2}.$$

This shows our protocol exactly reduces to the Brands' type scheme. So even if the bank colludes with the observer, meaning that the bank knows (o_1, o_2), the anonymity property still holds. □

2.4.3 One-More Unforgeability

In [BL12; BL13], it was proved that blind Schnorr signatures have the property of one-more-unforgeability. The probability that legitimately withdrawn coins through the withdraw protocol is negligible. Conversely, if a PPT adversary $\mathcal{A}$ could forge a coin, which means there exist $l + 1$ coins for correctly withdrawn l coins, the adversary can forge Schnorr signatures in the random oracle and generic group model.

Theorem 2.4 ([BL13, Theorem 3])**.** *Consider the modified Brands' withdrawal/blind signature scheme in the random oracle model. If there exists a probabilistic polynomial time Turing machine which can perform a "one-more" forgery, with non-negligible probability, even under a parallel attack, then the discrete logarithm can be solved in polynomial time.*

Theorem 2.5. *Consider our e-cash scheme, based on the modified Brands' withdraw/blind signature with Observers in the random oracle model. If there exists a probabilistic*

polynomial time Turing machine that can perform an "one-more" forgery, with non-negligible probability, even under a parallel attack, then the discrete logarithm can be solved in polynomial time.

Proof. In our protocol, the cryptographic modification to the original one is the introduction of the new parameter (o_1, o_2) in order to construct observers;

$$g_1 : G_1^U G_2 \to A_{\mathcal{O}} \left(G_1^U G_2 \right) \tag{2.2}$$

$$m : G_1^{b_1} G_2^{b_2} \to G_1^{b_1} G_2^{b_2} A_{\mathcal{O}}^{es} B_{\mathcal{O}} \tag{2.3}$$

where $A_{\mathcal{O}} = G_1^{o_1}$ and $B_{\mathcal{O}} = G_1^{o_2}$.
In the proof of Theorem 2.4, the contents of g_1 and m do not affect both the proof logic and the conclusion, so the theorem can be proven in the same way. □

2.4.4 Double-Spending Protection

The Open-cash has a powerful anti-double-spending property. Under the AESP assumption, the probability that double-spending is executed successfully is negligible.

Theorem 2.6. *Assuming AESP is not compromised the probability that a user can execute double-spending successfully is negligible.*

Proof. In order to execute double-spending successfully in spite of the existence of BIA, the adversary must impersonate his observer, which means he must calculate o_1 for $G_1^{o_1}$ by himself.
For the probability, directly from the assumption that discrete logarithm problem is hard, there exists a negligible function $negl(\lambda)$ such that

$$\Pr\left[(q, G_1, G_2, G_1^{o_1}) \leftarrow S\left(1^{\lambda}\right); o_1' \leftarrow \mathcal{A}^{\text{Schnorr }(G_1, o_1)}\left(G_1, G_1^{o_1}\right) : o_1' = o_1\right] \leq negl(\lambda).$$

□

Moreover, even if the AESP assumption is broken, Open-cash enables banks to extract the id of the attacker who double-spent one coin to two different *pid*s.

Theorem 2.7. *A bank can extract an attacker's user id with the observer if and only if the user executes a double-spending.*

Proof. When a malicious user double-spent one coin the bank will receive the same coin $((m, \tilde{g}_1), \sigma_B)$ for two different $(pid; p_1, p_2)$ and $(pid'; p_1', p_2')$. The bank can trace the attacker who double-spent the coin by computing:

$$G_1^{\frac{p_1 - p_1'}{pid - pid'}} G_2^{\frac{p_2 - p_2'}{\mathrm{pid} - pd'}}$$

which is equal to

$$G_1^{W_1} G_2^{W_2}$$

and by knowing $W_1 = (U + o_1)s$ and $W_2 = s$, the bank can compute the secret key (id) U of the double-spender.

□

2.5 Conclusion

In this chapter, I first defined the concept of open-source observers. And then I showed construction of Open-cash which is the Brands' type provably-secure offline electronic cash scheme with observers extending the scheme by Baldimtsi and Lysyanskaya, by newly introducing Blind Indirect Attestation with AESP attestation. I also proved that Open-cash satisfies anonymity, one-more unforgeability, and double-spending protection.

The concept of open-source observers can potentially achieve both cryptographic anonymity and highly flexible transparency at the same time while enforcing payers' legitimate behavior, which is desirable for a wide range of e-cash systems including Central Bank Digital Currency.

Chapter 3

Open-cash utilizing Zero-knowledge Proofs based on BBS+ Signatures

3.1 Main Contribution

In the previous chapter, I proposed an Open-cash scheme based on blind signatures. In this chapter, I present an Open-cash scheme founded on zero-knowledge proofs. By employing BBS+ signatures, this scheme can potentially provide greater flexibility and extensibility compared to the blind signature-based Open-cash. A key feature of this work itself is, the same as blind signature based one, that the bank or regulatory authority publishes the observer program as open-source program, ensuring transparency regarding its internal logic and security properties. Below, I summarize the main contributions of this work:

- **Open-source Observer under AESP:** I propose the concept of an open-source observer executed within a tamper-proof environment by releasing it via the public cloud. While our preliminary work [HO24] introduced the idea of an open-source observer in conjunction with a provably secure blind signature-based e-cash protocol, this work demonstrates that a similar scheme can be realized utilizing the Proof of Knowledge of Signature (SPK) of BBS+ signatures, further enhancing verifiability and flexibility in e-cash schemes.

- **Attestation of Protocol Execution:** Under the AESP assumption, the result of each execution of open-cash protocols (Open, Withdraw, and Spend protocol)

inside the secure element, can have an attestation that binds the hash of the observer program to the computation result, guaranteeing that a correct, unmodified observer carried out the protocol. This ensures that payments are correctly processed according to the intended security policies without disclosing users' private information.

- **Indirect Verification for AESP Attestation:** As shown in the previous chapter, our prior work presented Blind Indirect Attestation (BIA) [HO24], enabling third parties to verify the observer's AESP attestation in a blind manner within the blind signature scheme. In this chapter, I show that the same property can be achieved more naturally with SPK of BBS+ signatures. Consequently, recipients can verify that the observer program authorized a transaction, without undermining the payer's anonymity.

- **Extensibility with BBS+ Signatures:** BBS+ signatures support selective disclosure and efficient zero-knowledge proofs, which could make the open-cash scheme potentially compatible with other applications such as verifiable credentials (VCs) [Sed+21].

3.2 Specification and Security Notion of our E-cash

In this section, I present an overview of the protocols and define the security notions underpinning our E-cash scheme.

3.2.1 Specification of Open-cash protocols

The E-cash protocol consists of five phases: *Setup*, *Open*, *Withdraw*, *Spend*, and *Deposit*.

We follow a lifecycle-driven design principle that maps each security-critical boundary to exactly one protocol phase.

- **Setup** is executed once by $\mathcal{B}$ the assumed hardness assumptions (e.g., SDH), publishing the bank's issuer key.

- **Open** authenticates a user, registers ID_u, and isolates KYC data from the subsequent anonymity-preserving phases.

- **Withdraw**, **Spend**, and **Deposit** correspond to the classic issuance–payment–redemption triad as described by Brands' wallet-with-observer e-cash [Bra94], but instantiated here with BBS+ signatures.

This decomposition reflects the operational boundaries of real-world payment systems.

I now give a brief description of each protocol. Three entities participate: the bank $\mathcal{B}$, users $\mathcal{U}$, and shops $\mathcal{S}$.

- **Setup:** Executed once to initialize the system. On input of the security parameter 1^k, the algorithm outputs public parameters pp and the bank secret key γ:

$$(pp, \gamma) \leftarrow \mathit{Setup}(1^k). \tag{3.1}$$

  The bank also initializes two empty sets, $\mathbb{S}$ and $\mathbb{T}$, for spent-coin signatures and transcripts, respectively.

- **Open:** Equivalent to opening a bank account, this interactive protocol is executed once between $\mathcal{U}$ and $\mathcal{B}$. The user discloses their identity, and if the bank approves, a user identifier u is generated, and the mapping (u, I_u) is stored in the bank's database:

$$\langle (u, I_u), I_u \rangle \leftarrow \mathit{Open}_{\mathcal{U},\mathcal{B}}\langle (pp, w), (pp, w, \gamma) \rangle \tag{3.2}$$

- **Withdraw:** The protocol enables coin withdrawal through interaction between the user $\mathcal{U}$ and the bank $\mathcal{B}$. If all verification checks succeed, the user obtains a BBS+ coin signature σ, while the bank internally updates its ledger (output $\perp$). If any check fails, both parties output $\perp$.

$$\langle \perp, \sigma / \perp \rangle \leftarrow \mathit{Withdraw}_{\mathcal{B},\mathcal{U}}\langle (pp, \gamma), (pp, u) \rangle \tag{3.3}$$

- **Spend:** This protocol is designed for spending coins and can be executed offline. I denote by $m \in \{0,1\}^*$ the payment message (e.g., shop identifier, transaction timestamp) bound to each coin.

$$\langle \perp, \tau / \perp \rangle \leftarrow \mathit{Spend}_{\mathcal{U},\mathcal{S}}\langle (pp, m, \sigma(I_u)), \perp \rangle \tag{3.4}$$

Equation 3.4 represents the interactive *Spend* protocol. If the zero-knowledge proof verifying the coin's validity and that it is unspent succeeds, the shop obtains the transcript τ. Otherwise, both parties output the failure symbol $\perp$.

- **Deposit:** The shop sends the received user ID and coin information as a transcript to the bank at any time. The bank verifies the transcript and accepts or rejects the coin.

$$\mathit{valid}/\mathit{invalid} \leftarrow \mathit{Deposit}_{\mathcal{B},\mathcal{S}}\langle pp, \tau \rangle \tag{3.5}$$

Note that **Setup** is a system-wide procedure executed once by the bank; therefore Equation 3.1 is independent of the number of registered users. A user who has already completed the *Open* protocol and holds a stored identifier I_u skips the Open in later sessions and directly invokes *Withdraw*; new users, in contrast, must run Open exactly once to enroll their identifier and KYC data.

3.2.2 Definition of Security Notions

Anonymity

We say that an E-cash scheme satisfies the anonymity property if no adversary can win the following anonymity game. As mentioned earlier, given a signature and a private key, an adversary could determine whether the signature was generated using the private key. Hence, the adversary should not be given access to either key. The anonymity game between a challenger and an adversary $\mathcal{A}$ is defined as follows.

Informally, the goal of the adversary is to win by correctly guessing, from two coins correctly withdrawn by two distinct users, which user corresponds to a randomly chosen coin used in a spending transaction.

Formally, I give the following definition:

Definition 3.1. *An electronic cash scheme* Π *is said to be anonymous, if for any probabilistic polynomial-time adversary* $\mathcal{A}$*, the advantage in winning the anonymity*

game described in Algorithm 1 is negligible in k.

Algorithm 1 defines the challenger-adversary experiment for anonymity. In Section 3.4, we prove that every transcript produced by the real protocol is computationally indistinguishable from a transcript output by a simulator that lacks all secret keys. Hence, no probabilistic polynomial-time adversary can win Algorithm 1 with non-negligible advantage.

Algorithm 1 $\mathrm{Game}^{\mathtt{Anon}}_{\mathcal{A},\mathcal{U},\mathcal{B}}(1^k)$

1: $(pp, w, \gamma) \leftarrow \mathit{Setup}(1^k)$
2: $(\sigma_0, \sigma_1) \leftarrow \mathcal{A}^{\mathit{Open},\mathit{Withdraw},\mathit{Spend},\mathit{Deposit}}$
3: $b \xleftarrow{\$} \{0, 1\}$
4: $\tau_b \leftarrow \mathit{Spend}(\sigma_b)$
5: $b' \leftarrow \mathcal{A}(\tau_b)$
6: **return** 1 **if** $b' = b$ **else** 0

One-More Unforgeability

Formally, I define the following cryptographic game:

Algorithm 2 $\mathrm{Game}^{\mathtt{Unforge}}_{\mathcal{A},\mathcal{B},\mathcal{S}}(1^k, pp)$

1: $(pp, w, \gamma) \leftarrow \mathit{Setup}(1^k)$
2: Let $\mathbb{U} := \emptyset, \mathbb{S} := \emptyset, \mathbb{S}_\perp := \emptyset, \mathbb{T} := \emptyset$
3: $\mathcal{A}$ invokes the oracles

$$\begin{cases} \sigma \leftarrow \mathcal{A}^{\mathit{Open},\mathit{Withdraw}} \\ \tau \leftarrow \mathcal{A}^{\mathit{Spend}} \end{cases}$$

in any order and any number of times.
4: **return** 1 **if** $|\mathbb{S}| < |\mathbb{T}|$ **else** 0

Algorithm 3 Oracle : $\mathit{Open}(u, pp)$

1: $(pp, w, \gamma) \leftarrow \mathit{Setup}(1^k)$
2: Create I_u from u
3: **If** $I_u \in \mathbb{U}$, **Abort**
4: Update $\mathbb{U} := \mathbb{U} \cup \{I_u\}$
5: **return** I_u

Algorithm 4 Oracle : $\mathit{Withdraw}(I_u, pp)$

1: **If** $I_u \notin \mathbb{U}$, **Abort**
2: Create signature $\sigma(I_u)$ with γ
3: $\mathbb{S} \leftarrow \mathbb{S} \cup \{\sigma(I_u)\}$
4: **return** $\sigma(I_u)$

Algorithm 5 Oracle: *Spend&Deposit*$(\sigma(I_u), pp)$

1: Verify: $\sigma(I_u) \notin \mathbb{S}_\perp$
2: **If** the verification fails, **Abort**
3: Calculate transcript τ for $\sigma(I_u)$
4: $\mathbb{T} \leftarrow \mathbb{T} \cup \{\tau\}$, $\mathbb{S}_\perp \leftarrow \mathbb{S}_\perp \cup \{\sigma(I_u)\}$
5: **return** τ

Definition 3.2. *An electronic cash scheme* Π *is said to be one-more unforgeable, if for every probabilistic polynomial-time adversary* $\mathcal{A}$*, the probability of winning the one-more unforgeability game described in Algorithm 2 is negligible in* k*.*

Double-Spender Traceability

Informally, an e-cash scheme satisfies Double-spender Traceability if there exists an extractor that can extract the user identifier u from two distinct transcripts (τ, τ') derived from spending the same coin with different payment messages (m, m').

Formally,

Definition 3.3 (Double-Spender Traceability)**.** *Let* $\mathbb{S}$ *and* $\mathbb{U}$ *be two sets of withdrawn coins and transcripts of spent coins, respectively. There exists a probabilistic polynomial extractor* $\mathcal{E} : \mathbb{S} \times \mathbb{S} \to \{u_i\}$ *such that*

$$\Pr\Bigg[u \leftarrow \mathcal{E}(\tau, \tau') \Big| (pp, w, \gamma) \leftarrow \mathit{Setup}(1^k), \\ \langle \perp, \sigma(I_u) \rangle \leftarrow \mathit{Withdraw}_{\mathcal{B},\mathcal{U}} \langle (pp, \gamma), (pp, u) \rangle, \\ \tau \leftarrow \mathit{Spend}\,(\sigma(I_u), m) \wedge \tau' \leftarrow \mathit{Spend}(\sigma(I_u), m') \\ \wedge\, m \neq m' \Bigg] = 1 - \mathit{negl}(k). \tag{3.6}$$

This definition captures the standard security notion required for electronic cash schemes. The extractor fails primarily when the underlying cryptographic assumptions are violated, which occurs with at most negligible probability regarding the security parameter k.

With an observer installed in a tamper-resistant device, the scheme additionally

provides stronger properties such as preventive double-spending enforcement and reversibility if the hardware assumption is compromised.

Double-Spend Prevention

A primary requirement for electronic cash is that no user can successfully spend the same coin more than once. Concretely, once the *Withdraw* protocol has issued a valid coin, the probability of successfully executing a second (or subsequent) *Spend* with the same signature must be negligible. In systems where an observer is installed within an *Attested Execution Secure Processor (AESP)*, any attempt to reuse the coin is detected and rejected *before* it can succeed, because the adversary cannot forge the observer's attestation nor bypass its internal checks unless the AESP is compromised. Thus, assuming the AESP remains uncompromised, the observer program proactively enforces double-spend prevention.

Reversibility

Even if the tamper-resistant observer is compromised (i.e., all secret information stored within the AESP is leaked to the adversary), the scheme gracefully reverts to the basic, observer-free Open-cash scheme. That is, the core security properties of the Open-cash - anonymity, one-more unforgeability, and double-spender traceability - still hold. Hence, the presence of an observer strengthens double-spend prevention under normal conditions, but in the worst-case scenario where the AESP is compromised, the protocol still maintains its fundamental security guarantees.

Unlinkability of Views

In the design, the observer participates in both the *Withdraw* and *Spend* protocols. The variables transmitted to the observer and those shared with the bank or shop are randomized (separated) in such a way that it is infeasible to correlate them and extract the user's secret information, even if the observer colludes with the bank (or the shop). In other words, even if the observer and the bank/shop attempt to combine their respective transcripts, the randomization process ensures that individual protocol values cannot be linked together to de-anonymize the user.

Open-Source Observer We call an observer *open-source* if the bank (or a relevant authority) publicly distributes its source code, thereby providing transparency into its internal logic and ensuring the enforcement of legitimate usage rules. Additionally, when the observer executes within an AESP, each execution produces a tamper-evident *attestation* containing the hash of the installed observer program and its outputs, which can be indirectly verified through Proofs of Knowledge (PoKs). By comparing this hash against a whitelist of authorized programs, one can confirm:

1. That the observer program has not been altered or replaced (i.e., no unauthorized code modifications occurred).

2. That the processing steps strictly follow the open-source code, ensuring transparency regarding transaction validation.

Hence, users gain confidence that the observer's computations were executed correctly by the intended, unmodified software, while the bank obtains cryptographically verifiable assurances that potential double-spending attempts will be detected.

3.3 Background and Building Blocks

In this section, I first review the concept of bilinear maps, then discuss the assumptions underlying the Open-cash scheme, and finally review the essential building blocks used in the construction. I follow the notation introduced in [BBS04] and [BL10].

3.3.1 Background on Bilinear Maps

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative cyclic groups of prime order p. Let g_1 and g_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. We define $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ as an admissible bilinear map if it satisfies the following properties:

- *Bilinear*: For all $u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}$, it holds that $e(u^a, v^b) = e(u, v)^{ab}$.

- *Non-degenerate*: $e(g_1, g_2) \neq 1$ and $e(g_1, g_2)$ is a generator of $\mathbb{G}_T$.

- *Computable*: There exists an efficient algorithm to compute $e(u, v)$ for any $u \in \mathbb{G}_1$ and $v \in \mathbb{G}_2$.

The pair $(\mathbb{G}_1, \mathbb{G}_2)$ is often referred to as a bilinear group pair. Throughout this chapter, we consider bilinear maps $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are multiplicative cyclic groups of prime order p. Although we could set $\mathbb{G}_1 = \mathbb{G}_2$, we allow for the more general case where $\mathbb{G}_1 \neq \mathbb{G}_2$. This flexibility enables us to avoid restricting the choice to supersingular elliptic curves, allowing us to leverage certain families of elliptic curves to achieve shorter private keys and signatures.

3.3.2 Cryptographic Assumptions

The security of the Open-cash construction relies on the Strong Diffie-Hellman (SDH) assumption and the Decisional Diffie-Hellman (DDH) assumption. Specifically, the SDH assumption is used to prove the one-more unforgeability of the Open-cash scheme, while the DDH assumption is employed to ensure double-spend prevention. These assumptions are stated as follows:

Strong Diffie-Hellman Assumption

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be cyclic groups of prime order p, generated by g_1 and g_2, respectively. The q-Strong Diffie-Hellman (q-SDH) problem in $(\mathbb{G}_1, \mathbb{G}_2)$ is defined as follows:

Given a $(q+3)$-tuple of elements $(g_1, g_1^{\gamma}, \ldots, g_1^{\gamma^q}, g_2, g_2^{\gamma})$ as input, output a pair $\left(g_1^{1/(\gamma+x)}, x\right)$, where $x \in \mathbb{Z}_p^*$.

An algorithm $\mathcal{A}$ has advantage ϵ in solving the q-SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$ if

$$\Pr\left[\mathcal{A}\left(g_1, g_1^{\gamma}, \ldots, g_1^{\gamma^q}, g_2, g_2^{\gamma}\right) = \left(g_1^{1/(\gamma+x)}, x\right)\right] \geq \epsilon \tag{3.7}$$

where the probability is over the random choice of γ and the randomness of $\mathcal{A}$.

Definition 3.4. *The (q, t, ϵ)-SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$ if no t-time algorithm has advantage at least ϵ in solving the q-SDH problem.*

Decisional Diffie-Hellman Assumption

Let $\mathbb{G}$, generated by g, be a cyclic group of prime order p. The Decisional Diffie-Hellman (DDH) problem in $\mathbb{G}$ is defined as follows:

Given a tuple of elements (g, g^a, g^b, g^c) as input, determine whether $c = ab$. Output 1 if $c = ab$, and 0 otherwise.

An algorithm $\mathcal{A}$ has advantage ϵ in solving the DDH problem in $\mathbb{G}$ if

$$\Big| \Pr\left[g \leftarrow \mathbb{G}, a, b \leftarrow \mathbb{Z}_p : \mathcal{A}(g, g^a, g^b, g^{ab}) = 1\right] - \Pr\left[g \leftarrow \mathbb{G}, a, b, c \leftarrow \mathbb{Z}_p : \mathcal{A}(g, g^a, g^b, g^c) = 1\right] \Big| \geq \epsilon, \tag{3.8}$$

where the probability is taken over the uniform random choice of parameters to $\mathcal{A}$ and the random bits of $\mathcal{A}$.

Definition 3.5. *The (t, ϵ)-DDH assumption holds in $\mathbb{G}$ if no t-time algorithm has advantage at least ϵ in solving the DDH problem in $\mathbb{G}$.*

There are many groups in which the DDH problem is believed to be intractable. The best known algorithm for solving DDH is a full discrete logarithm algorithm. However, for cyclic groups equipped with symmetric pairings, the DDH problem is tractable.

3.3.3 BBS+ Signature Scheme

In this subsection, I review the BBS+ signature scheme, which is utilized in the Open-cash construction. The signature scheme is a variant of the Boneh–Boyen signature scheme. Let $(\mathbb{G}_1, \mathbb{G}_2)$ be a bilinear group pair of some prime order p. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a computable bilinear pairing function. The BBS+ signature scheme is defined as follows:

Key Generation Select $g_1, h_1, h_2 \leftarrow \mathbb{G}_1$, $g_2 \leftarrow \mathbb{G}_2$, $\gamma \leftarrow \mathbb{Z}_p^*$, and compute $w = g_2^{\gamma}$. The public key is the tuple (g_1, g_2, h_1, h_2, w). The private key is γ.

Sign Given a public key (g_1, g_2, h_1, h_2, w), the corresponding private key γ, and a message $m \in \mathbb{Z}_p$, the signing algorithm performs the following steps:

1. Choose $x, y \leftarrow \mathbb{Z}_p$.
2. Compute $A := (g_1 h_1^m h_2^y)^{1/(\gamma + x)}$.

The signature on m is $\sigma := (A, x, y)$.

Verify Given a public key (g_1, g_2, h_1, h_2, w), a message m, and a signature $\sigma = (A, x, y)$, the verification algorithm checks the following condition:

$$e(A, g_2^x w) \stackrel{?}{=} e(g_1 h_1^m h_2^y, g_2). \tag{3.9}$$

If the equality holds, the signature is valid; otherwise, it is invalid.

Lemma 3.1. *The BBS+ signature scheme is unforgeable against adaptive chosen message attacks under the q-SDH assumption.*

The proof of Lemma 3.1 is provided by Au et al. in [ASM06].

3.3.4 Protocols for Proof of Knowledge

In the scheme, we use various protocols to prove knowledge of discrete logarithms and relations among them. To describe these protocols, we follow the well-known notation originally introduced by Camenisch and Stadler [CS97], which is also adopted in [BL10]. For example,

$$\text{PoK}\{(a, b) : y_1 = g_1^a h_1^b \wedge y_2 = g_2^a h_2^b\} \tag{3.10}$$

denotes a proof of knowledge of integers a and b satisfying $y_1 = g_1^a h_1^b$ and $y_2 = g_2^a h_2^b$, where g_1, h_1 are generators of some group $\mathbb{G}_1$, and g_2, h_2 are generators of some group $\mathbb{G}_2$.

The variables inside the parentheses denote the values whose knowledge is being proven, while all other parameters are assumed to be known to the verifier. Using this notation, proof of knowledge protocols can be described concisely without delving into all details.

In the random oracle model, such proof of knowledge protocols can be transformed into signature schemes using the Fiat–Shamir heuristic. We use the notation $\text{SPK}\{(a) : y = z^a\}(m)$ to denote a signature on a message m obtained using this approach.

In this chapter, we rely on the following known proof of knowledge protocols:

- **Proof of knowledge of discrete logarithms.** A proof of knowledge of a discrete logarithm of an element $y \in \mathbb{G}$ with respect to a base z is denoted by

$\text{PoK}\{(a) : y = z^a\}$. A proof of knowledge of a representation of an element $y \in \mathbb{G}$ with respect to multiple bases $z_1, \ldots, z_v \in \mathbb{G}$ is denoted as

$$\text{PoK}\{(a_1, \ldots, a_v) : y = z_1^{a_1} \cdots z_v^{a_v}\}. \tag{3.11}$$

- **Proof of knowledge of equality.** A proof of equality of discrete logarithms of two group elements $y_1, y_2 \in \mathbb{G}$ with respect to bases $z_1, z_2 \in \mathbb{G}$, respectively, is denoted as

$$\text{PoK}\{(a) : y_1 = z_1^a \wedge y_2 = z_2^a\}. \tag{3.12}$$

  This protocol can also prove the equality of discrete logarithms of two elements $y_1 \in \mathbb{G}_1$ and $y_2 \in \mathbb{G}_2$, with respect to bases $z_1 \in \mathbb{G}_1$ and $z_2 \in \mathbb{G}_2$, even when these elements are in different groups $\mathbb{G}_1$ and $\mathbb{G}_2$

3.4 Construction of the Basic Open-Cash Scheme Based on BBS+ Signatures

In this section, I present a simplified version of the Open-cash scheme that does *not* incorporate an observer. This serves as the foundation upon which I introduce an observer in the next section. Even if the AESP assumption (addressed in Section 3.2) is compromised, the extended scheme reduces to the construction described here, illustrating the concept of reversibility.

3.4.1 Protocols

Below, I outline five protocols—*Setup*, *Open*, *Withdraw*, *Spend*, and *Deposit*—that constitute the core of the Open-cash system without an observer. Three entities participate in these protocols: $\mathcal{B}$ (the bank), $\mathcal{U}$ (the user), and $\mathcal{S}$ (the shop).

Overview

- **Setup Protocol.** Executed once by the bank, this protocol generates secret and public parameters used throughout the system. At the end of this protocol, $\mathcal{B}$ obtains a private key γ and publicly releases all other parameters.

- **Open Protocol.** A user who wishes to open an account interacts with the bank to establish a personal identifier in the system. The user creates a public key (representing the user's account) and sends it to the bank, which validates the user's identity. The bank then records this account information in its database.

- **Withdraw Protocol.** A user interacts with the bank to withdraw a single coin signed by the bank's private key. The user receives a valid signature under the BBS+ scheme as the coin, which can later be spent. This signature is bound to the user's account but does not reveal the user's identity to third parties.

- **Spend Protocol.** The user spends the coin offline at a shop by utilizing the SPK of the BBS+ signature. The shop verifies the proof to ensure the coin's validity. Upon successful verification, the shop accepts the payment.

- **Deposit Protocol.** The shop eventually deposits the received coin with the bank. The bank confirms that the coin has not already been deposited (i.e., not double-spent) and credits the shop's account if all checks pass.

Setup:

Given a security parameter 1^k, the bank chooses bilinear groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ of prime order p, and a cyclic group $\mathbb{G}$ of order p, where the Decisional Diffie–Hellman (DDH) problem is assumed to be hard. Let g_1, g_2, g be the generators of $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}$ respectively. The bank randomly selects

$$h_1, h_2, h_3, h_4, h_5 \leftarrow \mathbb{G}_1, \quad \gamma \leftarrow \mathbb{Z}_p^*, \tag{3.13}$$

and then computes $w := g_2^{\gamma}$. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map function and $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_p$ be a collision-resistant hash function. We treat $\mathcal{H}$ as a random oracle

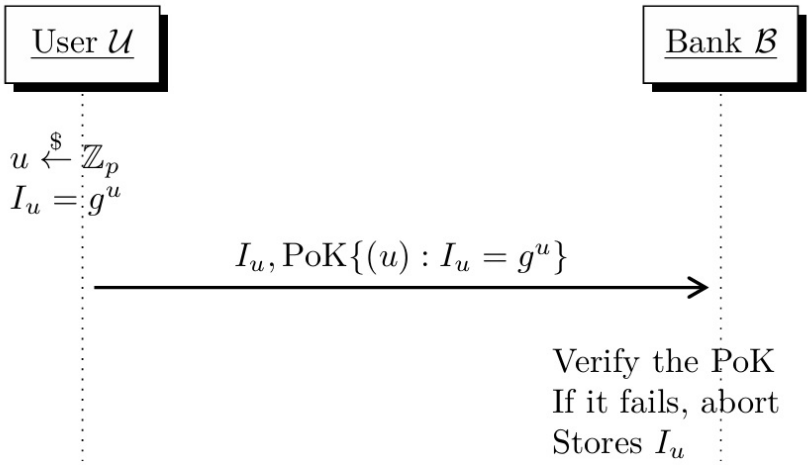

Figure 3.1: Open protocol

in the security proof. The public parameters and the issuing private key ($bsk = \gamma$) are defined as:

$$(pp, bsk) := \Big(\big(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}, g_1, g_2, g, h_1, h_2, h_3, h_4, h_5, w\big),\ \gamma \Big). \tag{3.14}$$

Open:

To open an account, a user $\mathcal{U}$ interacts with the bank $\mathcal{B}$, which verifies the user's identity (e.g., with official documents). The protocol proceeds as follows:

1. $\mathcal{U}$ selects $u \leftarrow \mathbb{Z}_p$ at random and computes $I_u := g^u$.

2. $\mathcal{U}$ sends I_u to $\mathcal{B}$ along with a proof of knowledge:

$$\mathrm{PoK}\{(u) : I_u = g^u\}. \tag{3.15}$$

3. If the proof is valid, $\mathcal{B}$ stores I_u in its database as the user's account record.

4. $\mathcal{U}$ retains (u, I_u) locally.

The figure is illustrated in Fig. 3.1.

Withdraw:

The Withdraw protocol (Fig. 3.2) is an interactive procedure between $\mathcal{B}$ and $\mathcal{U}$:

1. $\mathcal{U}$ randomly selects $y', s, x_1, x_2 \leftarrow \mathbb{Z}_p$, then computes

$$T := h_1^{y'}\, h_2^{u}\, h_3^{s}\, h_4^{x_1}\, h_5^{x_2}. \tag{3.16}$$

2. $\mathcal{U}$ sends T to $\mathcal{B}$ along with the following proof of knowledge:

$$\text{PoK}\{(y', u, s, x_1, x_2) : T = h_1^{y'} h_2^{u} h_3^{s} h_4^{x_1} h_5^{x_2} \ \wedge\ I_u = g^{u}\}. \tag{3.17}$$

3. $\mathcal{B}$ randomly selects $y'', x \leftarrow \mathbb{Z}_p^*$ and computes

$$A = \big(g_1\, T\, h_1^{y''}\big)^{\frac{1}{x+\gamma}}. \tag{3.18}$$

4. $\mathcal{B}$ sends the BBS+ signature (A, x, y'') to $\mathcal{U}$.

5. $\mathcal{U}$ verifies that

$$e(A,\, w\, g_2^{x}) \ \stackrel{?}{=}\ e\big(g_1\, h_1^{y'}\, h_2^{u}\, h_3^{s}\, h_4^{x_1}\, h_5^{x_2},\ g_2\big), \tag{3.19}$$

   and then sets $y := y' + y''$.

6. $\mathcal{U}$'s final coin (signature) is $(A, x, y;\ (u, s, x_1, x_2))$.

Spend:

A user $\mathcal{U}$ spends a withdrawn coin by engaging in an interactive proof protocol with the shop $\mathcal{S}$, as depicted in Fig. 3.3. Each payment transaction is uniquely identified by a pair consisting of the shop's identifier $\text{ID}_{\mathcal{S}}$ and a transaction timestamp. We denote this pair as the payment message $m = (\text{ID}_{\mathcal{S}}, Time)$. The steps of the spend protocol are as follows:

1. $\mathcal{U}$ computes $K_1 = G_1^{u} G_2^{s}$ and $K_2 = G_1^{x_1} G_2^{x_2}$.

2. $\mathcal{U}$ randomly selects $a \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, computes in $\mathbb{Z}_p$ that $b := y + ax$, and sets $T := A h_1^{a}$.

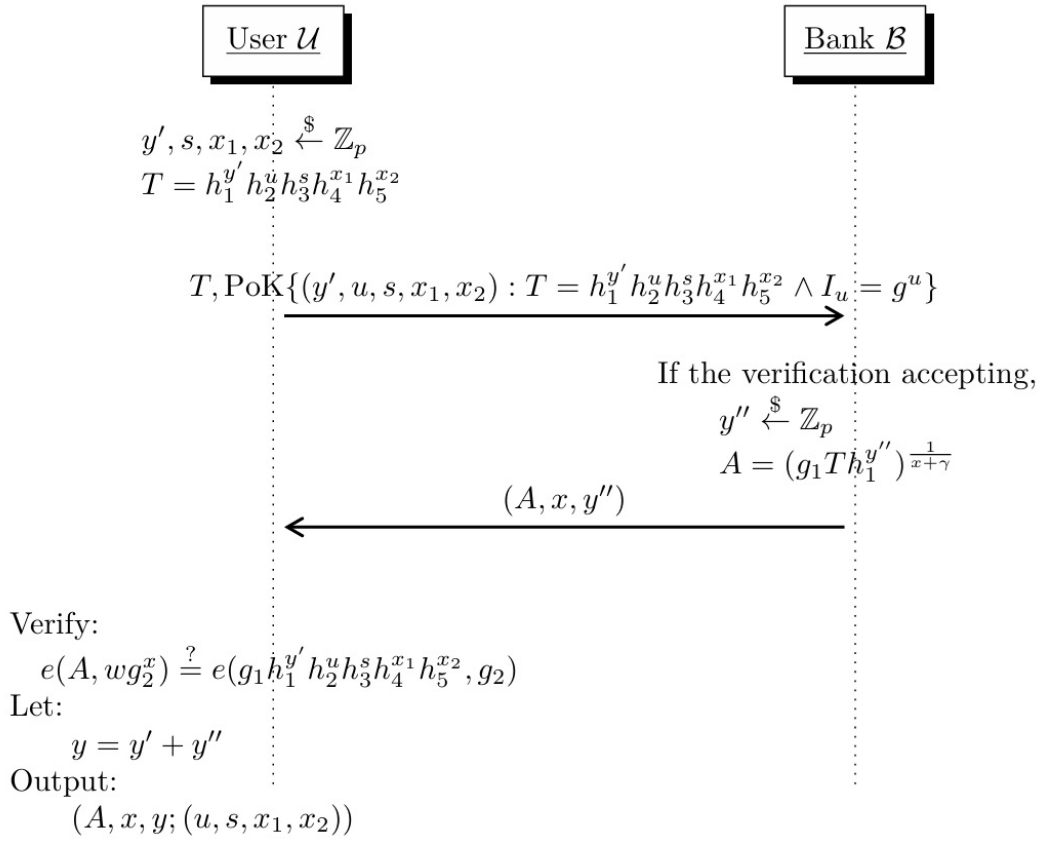

Figure 3.2: Withdraw protocol

3. $\mathcal{U}$ executes the following proof of knowledge of signature protocol, proving knowledge of $(x, u, s, x_1, x_2, a, b)$ without revealing them:

$$\begin{aligned}&\mathrm{SPK}\Big\{(x,u,s,x_1,x_2,a,b) : K_1 = G_1^u G_2^s \wedge K_2 = G_1^{x_1} G_2^{x_2}\\ &\wedge e(T,g_2)^{-x} e(h_1,g_2)^b e(h_1,w)^a e(h_2,g_2)^u e(h_3,g_2)^s\\ &e(h_4,g_2)^{x_1} e(h_5,g_2)^{x_2} = \frac{e(T,w)}{e(g_1,g_2)}\Big\}(m). \end{aligned} \tag{3.20}$$

(a) $\mathcal{U}$ randomly selects $r_x, r_u, r_a, r_b, r_s, r_{x_1}, r_{x_2} \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\begin{aligned}R_1 =& G_1^{r_u} G_2^{r_s}, R_2 = G_1^{r_{x_1}} G_2^{r_{x_2}}\\ R_3 =& e(T,g_2)^{-r_x} \cdot e(h_2,g_2)^{r_u} \cdot e(h_3,g_2)^{r_s}\\ &\cdot e(h_4,g_2)^{r_{x_1}} \cdot e(h_5,g_2)^{r_{x_2}}\\ &\cdot e(h_1,g_2)^{r_b} \cdot e(h_1,w)^{r_a}. \end{aligned} \tag{3.21}$$

(b) $\mathcal{U}$ then computes $c_1 \leftarrow \mathcal{H}(pp, K_1, K_2, T, R_1, R_2, R_3, m)$.

(c) $\mathcal{U}$ computes in $\mathbb{Z}_p$

$$\begin{aligned}&z_x := r_x + c_1 x, z_u := r_u + c_1 u, z_a := r_a + c_1 a,\\ &z_b := r_b + c_1 b, z_s := r_s + c_1 s, z_{x_1} := r_{x_1} + c_1 x_1,\\ &z_{x_2} := r_{x_2} + c_1 x_2 \end{aligned} \tag{3.22}$$

4. $\mathcal{U}$ sets $\sigma := (K_1, K_2, T, c_1, z_x, z_u, z_a, z_b, z_s, z_{x_1}, z_{x_2})$ and sends (σ, m) to $\mathcal{S}$.

5. $\mathcal{S}$ first verifies that

$$\begin{aligned}K_1, K_2 \in \mathbb{G}, T \in \mathbb{G}_1,\\ z_x, z_f, z_a, z_b, z_s, z_{x_1}, z_{x_2} \in \mathbb{Z}_p. \end{aligned} \tag{3.23}$$

6. $\mathcal{S}$ computes $c_2 \leftarrow \mathcal{H}(\sigma, m)$ and sends it to $\mathcal{U}$.

7. $\mathcal{U}$ calculates

$$d_1 = c_2 u + x_1$$
$$d_2 = c_2 s + x_2 \tag{3.24}$$

and sends (d_1, d_2) to $\mathcal{S}$.

8. $\mathcal{S}$ computes

$$\tilde{R}_1 := G_1^{z_u} G_2^{z_s} K_1^{-c_2}, \tag{3.25}$$
$$\tilde{R}_2 := G_1^{z_{x_1}} G_2^{z_{x_2}} K_2^{-c_2}, \tag{3.26}$$
$$\begin{aligned}\tilde{R}_3 := {} & e(T, g_2)^{-z_x} \cdot e(h_2, g_2)^{z_u} \cdot e(h_3, g_2)^{z_s} \\ & \cdot e(h_4, g_2)^{z_{x_1}} \cdot e(h_5, g_2)^{z_{x_2}} \\ & \cdot e(h_1, g_2)^{z_b} \cdot e(h_1, w)^{z_a} \cdot \left(\frac{e(g_1, g_2)}{e(T, w)}\right)^{c_1}.\end{aligned} \tag{3.27}$$

9. $\mathcal{S}$ verifies that

$$c_1 \stackrel{?}{=} \mathcal{H}(pp, K_1, K_2, T, \tilde{R}_1, \tilde{R}_2, \tilde{R}_3, m) \tag{3.28}$$
$$c_2 \leftarrow \mathcal{H}(\sigma, m) \tag{3.29}$$
$$G_1^{d_1} G_2^{d_2} \stackrel{?}{=} K_1^{c_2} K_2. \tag{3.30}$$

10. If all verifications pass, $\mathcal{S}$ outputs valid; otherwise, it outputs invalid.

Deposit:

When the shop $\mathcal{S}$ deposits the coin, it forwards the payment transcript (including σ and any relevant timestamp and shop ID to the bank $\mathcal{B}$. The bank verifies the transcript's authenticity and checks that the coin has not already been deposited. If the verification succeeds, $\mathcal{B}$ accepts the coin and credits the shop's account; otherwise, the deposit is rejected.

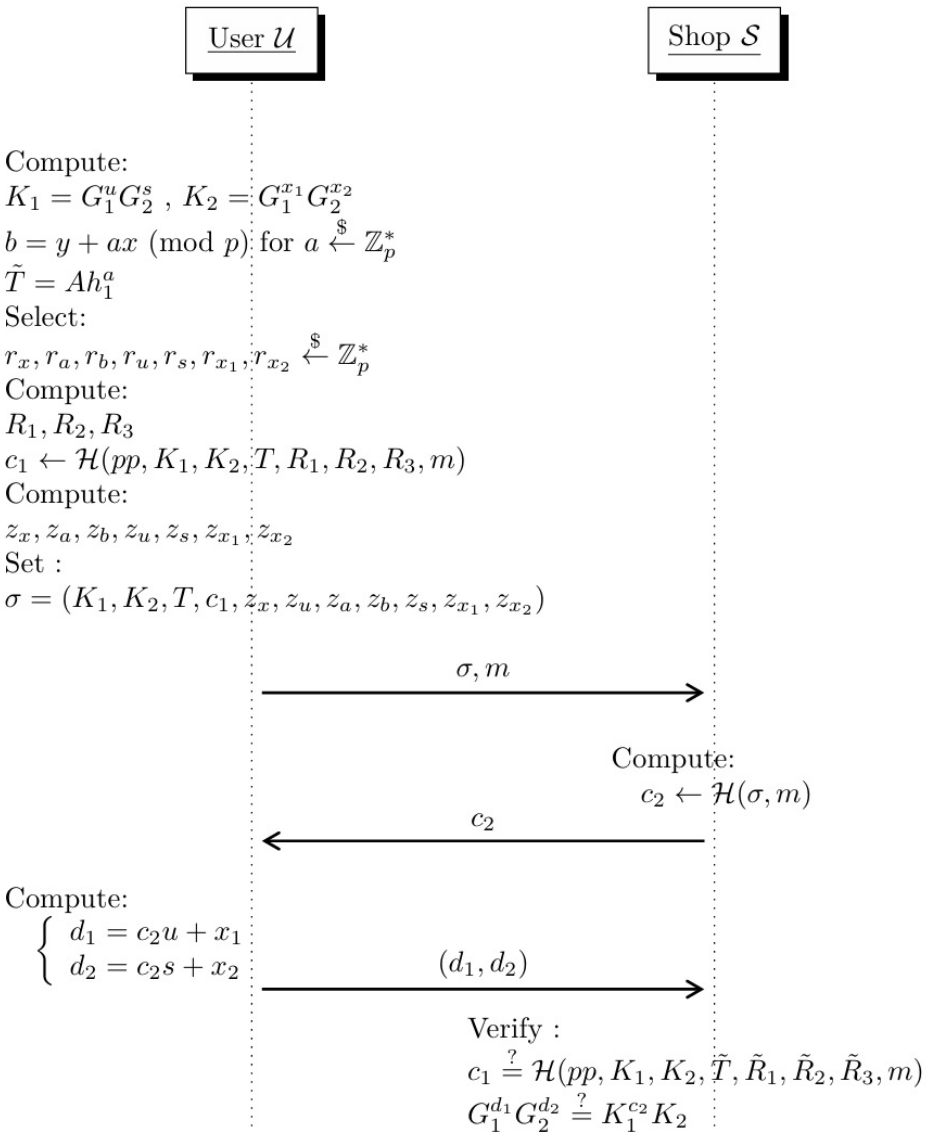

Figure 3.3: Spend protocol

3.4.2 Security Proofs

In this subsection, I present formal security proofs showing that Open-cash satisfies the notions of anonymity, one-more unforgeability, and double-spender traceability, as defined in Section 3.2.2.

Anonymity

Here, we prove that the Open-cash scheme satisfies the anonymity notion defined in Section 3.2, using a technique similar to the anonymity proof for the EPID scheme [BL10].

Before presenting the theorem and its proof, I first introduce the following two lemmas. Note that Lemma 3.3 requires sampling K_2 by first uniformly choosing random values $c, d_1, d_2 \in \mathbb{Z}_q$, and then setting:

$$K_2 \leftarrow G_1^{d_1} G_2^{d_2} / K_1^c. \tag{3.31}$$

However, K_2 can take all values in $\mathbb{G}$ with the uniform probability with a sufficiently large degree of freedom. Hence, this does not affect the uniformly random choice of K_2 which is required in Lemma 3.2.

Lemma 3.2. *The transcript π_σ, defined as:*

$$\pi_\sigma = (T, c_1, z_x, z_a, z_b, z_u, z_s, z_{x_1}, z_{x_2}, K_1, K_2) \tag{3.32}$$

can be simulated.

Proof. Observe that $T = Ah^a$, given the random choice of $a \xleftarrow{\$} \mathbb{Z}_p, T$ is distributed uniformly at random in $\mathbb{G}_1$. The simulator can simulate T by choosing $T' \xleftarrow{\$} \mathbb{G}_1$. Next, since c_1 is derived from the random oracle hash function $\mathcal{H}$, it is uniformly distributed over $\mathbb{Z}_p$. It is also easy to see that $z_x, z_a, z_b, z_u, z_s, z_{x_1}, z_{x_2}$ are uniformly distributed over $\mathbb{Z}_p$ given the random choice of $r_x, r_a, r_b, r_u, r_s, r_{x_1}, r_{x_2}$. The simulator chooses a random $c_1' \xleftarrow{\$} \mathbb{Z}_p$ and then chooses $z_x', z_a', z_b', z_u', z_s', z_{x_1}', z_{x_2}' \xleftarrow{\$} \mathbb{Z}_p$. For $K_1 = {G_1}^u {G_2}^s$ given the random choice of $s \xleftarrow{\$} \mathbb{Z}_p^*$, K_1 is distributed uniformly at random in $\mathbb{G}$. The simulator can simulate K_1 by choosing ${K_1}' \xleftarrow{\$} \mathbb{G}$. Similarly, given the random choices of

$x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p^*$, $K_2 = {G_1}^{x_1}{G_2}^{x_2}$ is distributed uniformly at random in $\mathbb{G}$, and the simulator can simulate K_2 by choosing ${K_2}' \xleftarrow{\$} \mathbb{G}$. Thus, the output of the simulator is distributed identically to the distribution of the correct transcripts, that is,

$$\begin{aligned}&\left(T', c_1', z_x', z_a', z_b', z_u', z_s', z_{x_1}', z_{x_2}', K_1', {K_2}'\right)\\ &\quad\sim (T, c_1, z_x, z_a, z_b, z_u, z_s, z_{x_1}, z_{x_2}, K_1, K_2)\,. \end{aligned} \tag{3.33}$$

Hence, the lemma holds. □

The following lemma states that the proof of knowledge regarding the witness (u, s) of $K_1 = G_1^u G_2^s$ can be simulated by a simulator that does not know the witness. Note that, in the actual Open-cash scheme, the commitment $K_2 = G_1^{x_1} G_2^{x_2}$, typically generated dynamically within the PoK, is fixed in advance as part of the message block in the BBS+ signature σ when withdrawing a coin from the bank.

Lemma 3.3. *The transcript of the proof of knowledge that* $\pi_K = \mathrm{PoK}\{(u, s) : K_1 = G_1^u G_2^s\}$*, defined as:*

$$\pi_K = (K_2, c_2, d_1, d_2) \tag{3.34}$$

can be simulated.

Proof. Recall that the prover generates π_K as follows. Given $K_2 = G_1^{x_1} G_2^{x_2}$ where $x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p$ and

$$c_2 \leftarrow \mathcal{H}(\sigma, \mathrm{ID}_{\mathcal{S}}, Time), \tag{3.35}$$

the response tuple (d_1, d_2) is calculated as:

$$d_1 = c_2 u + x_1 \tag{3.36}$$

$$d_2 = c_2 s + x_2. \tag{3.37}$$

On the other hand, the simulator that simulates π_K can be constructed as follows: Observe that c_2 is the result of the hash function $\mathcal{H}$. As we model $\mathcal{H}$ as a random oracle, c is uniformly distributed over $\mathbb{Z}_p$. It is also easy to see that d_1, d_2 are uniformly

distributed over $\mathbb{Z}_p$ given the random choice of $x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p$. The simulator can simulate c_2, d_1, d_2 by choosing a random $c_2', d_1', d_2' \xleftarrow{\$} \mathbb{Z}_p$.

$K_2 = G_1^{x_1} G_2^{x_2}$ is also distributed uniformly at random in $\mathbb{G}$ because of the random choice $x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p$. The simulator gets K_2' by $G_1^{d_1'} G_2^{d_2'} / K_1^{c_2'}$. Here, K_2' can take all values in $\mathbb{G}$ with the uniform probability. Thus, the output of the simulator is distributed identically to the distribution of the correct transcripts, that is,

$$(K_2', c_2', d_1', d_2') \sim (K_2, c_2, d_1, d_2)\,. \tag{3.38}$$

Hence, the lemma holds. □

Note that the proof of Lemma 3.2 requires to choose $K_2 \xleftarrow{\$} \mathbb{G}$, but here we require to compute K_2 with random values (c_2', d_1', d_2'). However, all the random values (c_2', d_1', d_2') are uniformly sampled from $\mathbb{Z}_p$, hence it has a sufficiently large degree of freedom. Thus, this does not interfere with the uniformly random choice of K_2 required in Lemma 3.2.

Theorem 3.1. *The Open-cash is anonymous in the random oracle model.*

Proof. From Lemmas 3.2 and 3.3, there exists a simulator that, without knowing the witness corresponding to a transcript $\tau = (\pi_\sigma, \pi_K)$, generates a transcript with an identical distribution. Hence, any adversary $\mathcal{A}$ obtains no information to distinguish between given transcripts τ_0 and τ_1. The probability that $\mathcal{A}$ wins the anonymity game defined in Algorithm 1 is exactly 1/2. □

One-More Unforgeability

Here, similarly to the anonymity proof, we prove that the Open-cash scheme satisfies the notion of one-more unforgeability as defined in Section 3.2, following a proof technique similar to that used in [BL10].

Before presenting the theorem and proof, we first introduce the following Lemma.

Lemma 3.4. *There exists a knowledge extractor that extracts a tuple* $(A, x, y, u, s, x_1, x_2)$ *from a prover, such that* $e(A, g_2^{xw}) = e(g_1 h_1^y h_2^u h_3^s h_4^{x_1} h_5^{x_2}, g_2)$, $K_1 = G_1^u G_2^s$, *and* $K_2 = G_1^{x_1} G_2^{x_2}$.

Proof. Suppose that a knowledge extractor can rewind the prover and control the output of $\mathcal{H}$. The prover first computes T, then chooses $r_x, r_a, r_b, r_u, r_s, r_{x_1}, r_{x_2}$ and computes R_1, R_2, R_3. For a hash value c_1, the prover outputs $z_x, z_a, z_b, z_u, z_s, z_{x_1}, z_{x_2}$. For another hash value c_1', the prover outputs $z_x', z_a', z_b', z_u', z_s', z_{x_1}', z_{x_2}'$. If the prover is convincing, the verification in the spend protocol would pass successfully. In other words, for two set of values, $\hat{R}_1 = R_1$, $\hat{R}_2 = R_2$, and $\hat{R}_3 = R_3$.

Define $\Delta c_1 = c_1 - c_1'$, $\Delta z_x = z_x - z_x'$, and similarly for $\Delta s_a, \Delta s_b, \Delta z_u, \Delta z_s, \Delta z_{x_1}, \Delta z_{x_2}$. Then, let

$$\begin{aligned} \hat{x} := \Delta z_x / \Delta c_1, \hat{a} := \Delta z_a / \Delta c_1, \hat{b} := \Delta z_b / \Delta c_1, \hat{u} := \Delta z_u / \Delta c_1, \\ \hat{s} := \Delta z_s / \Delta c_1, \hat{x_1} := \Delta z_{x_1} / \Delta c_1, \hat{x_2} := \Delta z_{x_2} / \Delta c_1. \end{aligned} \tag{3.39}$$

These values $\hat{x}, \hat{a}, \hat{b}, \hat{u}, \hat{s}, \hat{x_1}, \hat{x_2}$, satisfy the following three equations:

$$\begin{aligned} e(T, g_2)^{-\hat{x}} \cdot e(h_1, w)^{\hat{a}} \cdot e(h_1, g_2)^{\hat{b}} \cdot e(h_2, g_2)^{\hat{u}} \cdot e(h_3, g_2)^{\hat{s}} \\ \cdot e(h_4, g_2)^{\hat{x_1}} \cdot e(h_5, g_2)^{\hat{x_2}} = \frac{e(T, w)}{e(g_1, g_2)}, \\ K_1 = G_1^{\hat{u}} G_2^{\hat{s}}, K_2 = G_1^{\hat{x_1}} G_2^{\hat{x_2}}. \end{aligned} \tag{3.40}$$

Let $\hat{y} := \hat{b} - \hat{a}$, we derive

$$\begin{aligned} e(T, g_2^{\hat{y}} w) &= e(T, w) \cdot e(T, g_2)^{\hat{x}} \\ &= e(g_1, g_2) \cdot e(h_1, g_2)^{\hat{f}} \cdot e(h_2, g_2)^{\hat{b}} \cdot e(h_2, w)^{\hat{a}} \\ &= e(g_1 h_1^f h_2^y, g_2) \cdot e(h_2, g_2)^{\hat{a}\hat{x}} \cdot e(h_2, w)^{\hat{a}} \\ &= e(g_1 h_1^{\hat{y}} h_2^{\hat{u}} h_3^{\hat{s}} h_4^{\hat{x}_1} h_5^{\hat{x}_2}, g_2) \cdot e(h_1, g_2^{\hat{a}\hat{x}} w). \end{aligned} \tag{3.41}$$

Thus we have

$$e(T \cdot h_2^{-\hat{a}}, g_2^{\hat{y}} w) = e(g_1 h_1^{\hat{y}} h_2^{\hat{u}} h_3^{\hat{s}} h_4^{\hat{x}_1} h_5^{\hat{x}_2}, g_2). \tag{3.42}$$

Let $\hat{A} := T h_2^{-\hat{a}}$, the extractor obtains a tuple $(\hat{A}, \hat{x}, \hat{y}, \hat{u}, \hat{s}, \hat{x}_1, \hat{x}_2)$ satisfying

$$e(\hat{A}, g_2^{\hat{x}} w) = e(g_1 h_1^{\hat{y}} h_2^{\hat{u}} h_3^{\hat{s}} h_4^{\hat{x}_1} h_5^{\hat{x}_2}, g_2). \tag{3.43}$$

Hence, the lemma holds. □

Theorem 3.2. *The Open-cash scheme is one-more unforgeable in the random oracle model under the Strong Diffie-Hellman assumption in* $(\mathbb{G}_1, \mathbb{G}_2)$.

Proof. Suppose algorithm $\mathcal{A}$ breaks the one-more unforgeability of the Open-cash scheme. We prove the theorem by reduction, constructing an algorithm $\mathcal{B}$ that breaks the unforgeability of the BBS+ signature scheme [ASM06; CDL16].

Algorithm $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

Setup. Let $(\mathbb{G}_1, \mathbb{G}_2)$ be a bilinear group pair of prime order p with generators g_1 and g_2, respectively. $\mathcal{B}$ is given a public key $(g_1, g_2, h_1, h_2, h_3, h_4, h_5, w)$ of the BBS+ signature scheme, where $h_1, h_2, h_3, h_4, h_5 \in \mathbb{G}_1$ and $w \in \mathbb{G}_2$. $\mathcal{B}$ proceeds as follows:

1. $\mathcal{B}$ selects a cyclic group $\mathbb{G}$ of order p with generator g and gives $\mathcal{A}$ the public parameters $\text{pp} = (pp, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}, g_1, g_2, g, h_1, h_2, h_3, h_4, h_5, w)$.

2. $\mathcal{B}$ maintains two lists $\mathbb{S}, \mathbb{T}$, representing sets of signatures and transcripts, respectively. Initially, $\mathbb{S} := \emptyset$ and $\mathbb{T} := \emptyset$.

Hash Queries. At any time, $\mathcal{A}$ may query the hash function $\mathcal{H}$. $\mathcal{B}$ responds with random values in $\mathbb{Z}_q$ while ensuring consistency.

Open Queries. $\mathcal{A}$ can select an witness $u \in \mathbb{Z}_p$ representing the witness of a user and registers the identity of the new user $I_u = g^u \in \mathbb{G}$ with $\mathcal{B}$.

Withdraw Queries. When $\mathcal{A}$ makes the i-th withdraw query, two cases may occur:

1. $\mathcal{B}$ and $\mathcal{A}$ run the withdraw protocol as follows: $\mathcal{A}$ executes 1-2 of the withdraw protocol. As step 2 of the withdraw protocol is a proof of knowledge protocol, $\mathcal{B}$ obtains $(y_i', u_i, s_i, x_{1i}, x_{2i})$ by rewinding $\mathcal{A}$. $\mathcal{B}$ now queries the BBS+ signature signing oracle on u_i, s_i, x_{1i}, x_{2i} and obtains (A_i, x_i, y_i). $\mathcal{B}$ computes $y_i'' := y_i - y_i'$ and returns (A_i, x_i, y_i'') to $\mathcal{A}$. $\mathcal{B}$ also appends $(A_i, x_i, y_i; u_i, s_i, x_{1i}, x_{2i})$ to $\mathbb{S}$.

2. $\mathcal{B}$ generates a fake signature as follows: $\mathcal{B}$ randomly selects $u_i, s_i, x_{1i}, x_{2i} \xleftarrow{\$} \mathbb{Z}_p$. Next, $\mathcal{B}$ sets $(A_i, x_i, y_i) := \star$ which indicates that $\mathcal{B}$ does not know the corresponding BBS+ signature on u_i, s_i, x_{1i}, x_{2i}. Finally, $\mathcal{B}$ appends $(A_i, x_i, y_i; u_i, s_i, x_{1i}, x_{2i})$ to $\mathbb{S}$.

Spend Queries.

$\mathcal{A}$ asks for a spend query to $\mathcal{B}$ with a unique text string m. This is actually a signature proof of knowledge on a message m with the set of witnesses at index i in $\mathbb{S}$.

The response of $\mathcal{B}$ is divided into the following two cases:

1. If $(A_i, x_i, y_i) \neq \star$, $\mathcal{B}$ follows the spending protocol with the witness $(A_i, x_i, y_i; u_i, s_i, x_{1i}, x_{2i})$ to obtain a signature proof of knowledge π_σ on m. Simultaneously, $\mathcal{B}$ executes another proof of knowledge with $(u_i, s_i, x_{1i}, x_{2i})$ to obtain π_K. Then $\mathcal{B}$ sends $\tau = (\pi_\sigma, \pi_K)$, which is a transcript of spent coin, and returns τ to $\mathcal{A}$. $\mathcal{B}$ appends τ to $\mathbb{T}$.

2. In the case where $(A_i, x_i, y_i) = \star$, i.e., $\mathcal{B}$ does not have a valid BBS+ signature for the i-th withdraw query but has u_i, s_i, x_{1i}, x_{2i}, $\mathcal{B}$ can simulate a signature proof of knowledge on m as follows: $\mathcal{B}$ runs the first step of the withdraw algorithm and obtains a (K_1, K_2) pair. $\mathcal{B}$ then simulates the remainder of σ by choosing $T \xleftarrow{\$} \mathbb{G}_1$ and $z_x, z_a, z_b, z_u, z_s, z_{x_1}, z_{x_2} \xleftarrow{\$} \mathbb{Z}_p$. $\mathcal{B}$ computes R_1, R_2, R_3 using equations in step 3 of the spend protocol, and then patches the hash oracle such that

$$\mathcal{H}(pp, K_1, K_2, T, R_1, R_2, R_3, m) = c_1. \tag{3.44}$$

If this causes a collision in the hash oracle, $\mathcal{B}$ reports failure and aborts. Otherwise, algorithm $\mathcal{B}$ sets

$$\tau_\sigma := (K_1, K_2, T, c_1, z_x, z_a, z_b, z_u, z_s, z_{x_1}, z_{x_2}). \tag{3.45}$$

Simultaneously, $\mathcal{B}$ executes another proof of knowledge with $(u_i, s_i, x_{1i}, x_{2i})$ to obtain π_K. Finally, $\mathcal{B}$ obtains $\tau = (\pi_\sigma, \pi_K)$ and return τ to $\mathcal{A}$. $\mathcal{B}$ appends τ to $\mathbb{T}$.

Observe that the *Spend* protocol is essentially a proof of knowledge protocol

$$\begin{aligned} \mathrm{PoK}\Big\{ & (A, x, y; (u, s, x_1, x_2)) : \\ & e\left(A, g_2^x w\right) = e\left(g_1 h_1^{\hat{y}} h_2^{\hat{u}} h_3^{\hat{s}} h_4^{\hat{x}_1} h_5^{\hat{x}_2}, g_2\right) \\ & \wedge K_1 = G_1^u G_2^s \wedge K_2 = G_1^{x_1} G_2^{x_2} \Big\} \end{aligned} \tag{3.46}$$

Using the knowledge extractor of Lemma 3.4 ,$\mathcal{B}$ obtains an $(A^*, x^*, y^*; u^*, s^*, x_1^*, x_2^*)$ tuple from $\tau^* \in \mathbb{T}$ such that

$$\begin{aligned} e\left(A^*, g_2^{x^*} w\right) = e\left(g_1 h_1^{y^*} h_2^{u^*} h_3^{s^*} h_4^{x_1^*} h_5^{x_2^*}, g_2\right), \\ K_1 = G_1^{u^*} G_2^{s^*}, \quad K_2 = G_1^{x_1^*} G_2^{x_2^*}. \end{aligned} \tag{3.47}$$

At the end of this game, $\mathcal{B}$ outputs the tuple $(A^*, x^*, y^*, u^*, s^*, x_1^*, x_2^*)$. Observe that, (A^*, x^*, y^*) is a valid BBS+ signature on (u^*, s^*, x_1^*, x_2^*). Also observe that $\mathcal{B}$ has queried the BBS+ signature oracle only for $(u_i, s_i, x_{1i}, x_{2i})$ in $\mathbb{S}$. As $(u^*, s^*, x_1^*, x_2^*) \notin \mathbb{S}, \mathcal{B}$ has successfully forged a BBS+ signature on (u^*, s^*, x_1^*, x_2^*) if $\mathcal{B}$ does not abort during the simulation. As the BBS+ signature scheme is unforgeable under chosen message attacks based on the SDH assumption, it follows that the Open-cash scheme is also one-more unforgeable under the SDH assumption.

□

Double-Spender Traceability

The scheme ensures that a malicious user attempting to double-spend the same coin can be traced, as stated in the following theorem.

Theorem 3.3 (Double-Spender Traceability)**.** *Under the assumption that the discrete logarithm problem is hard in the underlying groups, if a malicious user successfully spends the same coin more than once, the bank can recover that user's identity (i.e., the secret exponent corresponding to the user's public key), achieving full traceability of the double-spender.*

Proof. Suppose an adversarial user $\mathcal{A}$ double-spends the same coin in two distinct transactions. We analyze two possible scenarios:

Case 1: Identical Witnesses for K_1, K_2 In this scenario, the adversary uses the same witness (u, s, x_1, x_2) in both transactions, performing proofs of knowledge for

$$K_1 = G_1^u G_2^s \quad \text{and} \quad K_2 = G_1^{x_1} G_2^{x_2}. \tag{3.48}$$

During the *Spend* protocol, the adversary responds to distinct challenges $c_2 \neq c_2'$ with corresponding responses (d_1, d_2) and (d_1', d_2'). From these, the bank (or the shop) recovers u by

$$u \;=\; \frac{d_1 - d_1'}{c_2 - c_2'}, \tag{3.49}$$

thus revealing the user's secret exponent and, consequently, the identity.

Case 2: Different Representations of K_1, K_2 Alternatively, $\mathcal{A}$ might attempt to provide different representations $(u', s') \neq (u, s)$ for K_1 such that

$$K_1 \;=\; G_1^u G_2^s \;=\; G_1^{u'} G_2^{s'}. \tag{3.50}$$

Similarly, distinct representations $(x_1', x_2') \neq (x_1, x_2)$ might be attempted for K_2. However, the security of the zero-knowledge proof

$$\mathrm{PoK}\Big((u, s, x_1, x_2) :\; K_1 = G_1^u G_2^s \;\wedge\; K_2 = G_1^{x_1} G_2^{x_2}\Big) \tag{3.51}$$

ensures that $\mathcal{A}$ cannot produce *both* (u, s) and (u', s') for the same K_1 without effectively solving the discrete logarithm problem. Indeed, if

$$G_1^u G_2^s \;=\; G_1^{u'} G_2^{s'}, \tag{3.52}$$

one can easily derive

$$\log_{G_1}(G_2) \;=\; \frac{u - u'}{s' - s} \bmod p, \tag{3.53}$$

which contradicts the assumed hardness of discrete logarithms.

In either scenario, the bank can recover the malicious user's secret key (and thus

their identity), completing the double-spender traceability proof.

□

3.5 Construction of Open-cash Scheme with BBS+ Observer

In this section, I extend the basic Open-cash scheme from Section 3.4 by introducing an observer, denoted by $\mathcal{O}$, which runs inside a secure execution environment. The main difference from the basic scheme is that the BBS+ signatures and their corresponding ZKPs now incorporate both the user's witnesses (u, s) and the observer's witnesses (o_1, o_2). Instead of the user alone proving possession of (u, s), the modified protocols rely on a joint proof of knowledge involving all four secret values (u, s, o_1, o_2).

Only transactions of the *Withdraw* and *Spend* protocols that have been approved and signed off by both the user and the observer pass the verification checks. Intuitively, the observer's partial control over signatures prevents a dishonest user from independently generating valid spend tokens. Thus, if a user attempts fraudulent behavior, the observer detects and immediately prevents double-spending. Moreover, if the AESP assumption is compromised, this extended scheme gracefully reverts to the basic construction of Section 3.4, preserving all fundamental security properties defined in Section 3.2.

Setup:

This protocol is the same as the one described in Section 3.4.

Open:

The user installs an open-source observer program, created and distributed by the bank, into the secure element of their device. The observer program $\texttt{prog}_{\mathcal{O}}$ internally stores a unique identifier o_1. The user then executes the *Open* protocol with the bank $\mathcal{B}$ as follows:

1. $\mathcal{U}$ randomly selects $u \xleftarrow{\$} \mathbb{Z}_p$ and computes $I_u = g^u$.

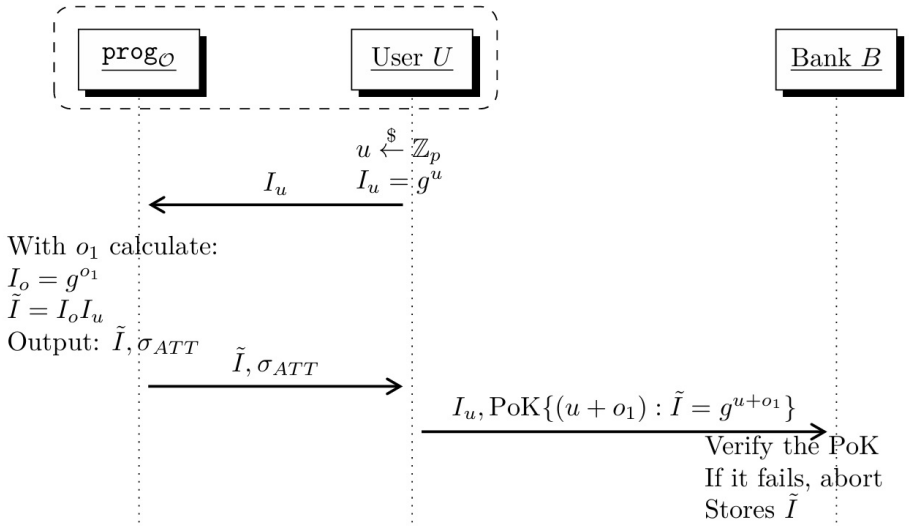

Figure 3.4: Open protocol with observer

2. $\mathcal{U}$ sends I_u to the observer $\mathcal{O}$, along with a proof of knowledge demonstrating knowledge of u:

$$\text{PoK}\{(u) : I_u = g^u\}. \tag{3.54}$$

3. $\mathcal{O}$ computes $I_o = g^{o_1}$, sets $\tilde{I} := I_u I_o$, and returns $\tilde{I}$ to $\mathcal{U}$ along with an attestation signature $(\tilde{I}, \sigma_{\text{ATT}}(\tilde{I}))$, provided the verification is successful.

4. $\mathcal{U}$ provides a non-interactive zero-knowledge proof to $\mathcal{B}$:

$$\text{PoK}\{(u + o_1) : \tilde{I} = g^{u+o_1}\}. \tag{3.55}$$

   This proof is created jointly by $\mathcal{U}$ and $\mathcal{O}$. If attestation verification is required, $\mathcal{U}$ also forwards $\sigma_{\text{ATT}}(\tilde{I})$ to $\mathcal{B}$.

5. If all proofs are successfully verified, $\mathcal{B}$ stores $\tilde{I}$ in its database

6. $\mathcal{U}$ locally stores $(u, \tilde{I})$.

The protocol flow is illustrated in Fig. 3.4.

Withdraw:

This protocol extends the basic *Withdraw* by involving the observer $\mathcal{O}$ (Fig. 3.5). The primary goal is for $\mathcal{O}$ to generate and securely store an additional secret o_2 *per coin*, en-

abling the observer to proactively detect and prevent future double-spending attempts. The protocol proceeds as follows:

1. $\mathcal{U}$ randomly selects $y', s, x_1, x_2 \leftarrow \mathbb{Z}_p$ and computes

$$T_u := h_1^{y'} \, h_2^{u} \, h_3^{s} \, h_4^{x_1} \, h_5^{x_2}. \tag{3.56}$$

2. $\mathcal{O}$ randomly selects $o_2, x_3, x_4 \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$T_o := h_2^{o_1} \, h_3^{o_2} \, h_4^{x_3} \, h_5^{x_4}. \tag{3.57}$$

   Additionally, $\mathcal{O}$ securely stores o_2 along with a coin identifier within its tamper-proof storage for future verification during spending.

3. $\mathcal{U}$ sends T_u to $\mathcal{O}$, accompanied by the following proof of knowledge:

$$\text{PoK}\{(y', u, s, x_1, x_2) : T_u = h_1^{y'} h_2^{u} h_3^{s} h_4^{x_1} h_5^{x_2} \ \wedge \ I_u = g^{u}\}. \tag{3.58}$$

4. Upon successful verification, $\mathcal{O}$ computes $\tilde{T} := T_u \, T_o$ and returns $\tilde{T}$ to $\mathcal{U}$, together with an attestation signature $(\tilde{T}, \sigma_{\text{ATT}}(\tilde{T}))$ to $\mathcal{U}$.

5. $\mathcal{U}$ then sends $\tilde{T}$ to $\mathcal{B}$, jointly providing a proof of knowledge with $\mathcal{O}$

$$\begin{aligned} \text{PoK}\{ \, & (y', u + o_1, s + o_2, x_1 + x_3, x_2 + x_4) : \\ & \tilde{T} = h_1^{y'} \, h_2^{u+o_1} \, h_3^{s+o_2} \, h_4^{x_1+x_3} \, h_5^{x_2+x_4} \\ & \wedge \ \tilde{I} = g^{\,u+o_1}\}. \end{aligned} \tag{3.59}$$

   If attestation verification is required, $\mathcal{U}$ also provides the attestation signature $\sigma_{\text{ATT}}(\tilde{T})$ to $\mathcal{B}$.

6. $\mathcal{B}$ randomly selects $y'', x \xleftarrow{\$} \mathbb{Z}_p^*$ and computes

$$\tilde{A} \ = \ \left(g_1 \, \tilde{T} \, h_1^{y''} \right)^{\frac{1}{x+\gamma}}. \tag{3.60}$$

7. $\mathcal{B}$ returns the signature tuple $(\tilde{A}, x, y'')$ to $\mathcal{U}$.

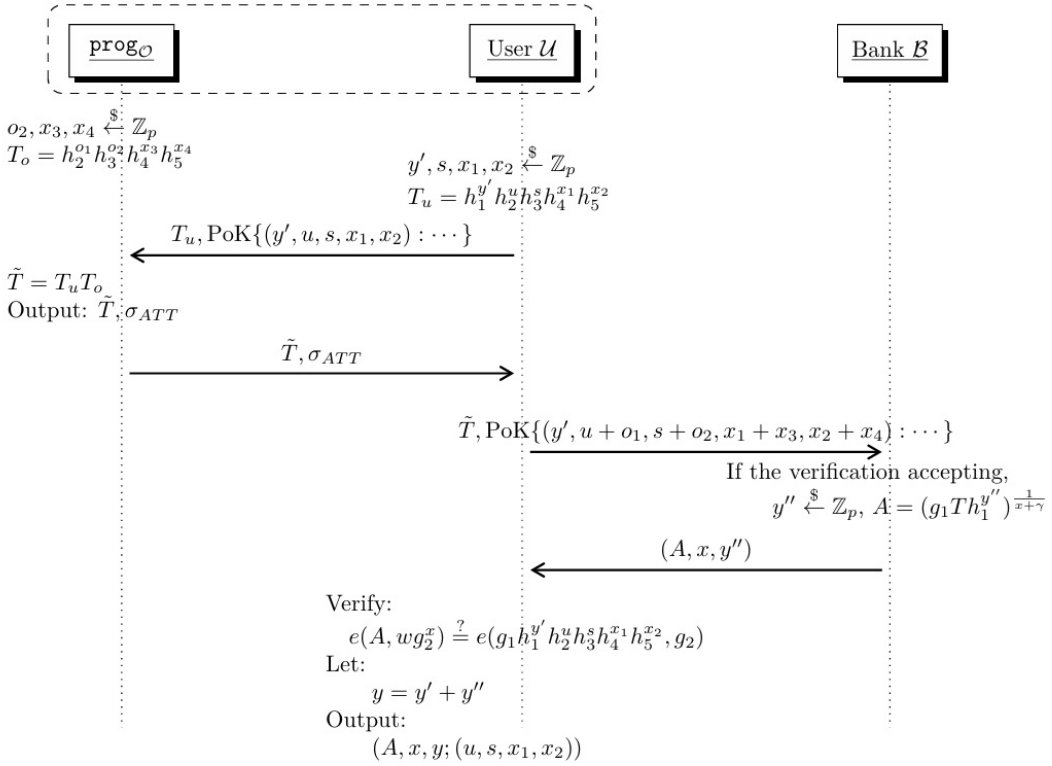

Figure 3.5: Withdraw protocol with observer

8. $\mathcal{U}$ computes $y \coloneqq y' + y''$ and verifies

$$e\big(\tilde{A},\ w\, g_2^{x}\big) \ \stackrel{?}{=}\ e\big(g_1\, \tilde{T}\, h_1^{y''},\ g_2\big). \tag{3.61}$$

9. Finally, the resulting coin held by the user is $(\tilde{A}, x, y;\ (u, s, x_1, x_2))$, and the observer securely retains o_2 in tamper-proof storage, enabling future detection and prevention of double-spending.

Spend

In the extended *Spend* protocol (Fig. 3.6), a user $\mathcal{U}$ spends a coin with the involvement of the observer $\mathcal{O}$. The observer's primary role here is to verify that the coin, uniquely identified by o_2, has not been previously spent. If $\mathcal{O}$ does not find the corresponding o_2 in its secure storage, or detects it has already been deleted, the observer aborts the protocol, indicating an invalid or double-spent coin. Otherwise, upon successful verification, the observer securely deletes o_2 to prevent further reuse.

Each payment transaction in this protocol is uniquely identified by a pair comprising

the shop identifier $\mathrm{ID}_{\mathcal{S}}$ and a transaction timestamp. We denote this pair as the payment message $m = (\mathrm{ID}_{\mathcal{S}}, Time)$.

The protocol steps closely follow those described in Section 3.4, with the observer's involvement detailed as follows:

1. $\mathcal{U}$ calculates $K_1 = G_1^u G_2^s$ and $K_2 = G_1^{x_1} G_2^{x_2}$.

2. $\mathcal{O}$ calculates $K_3 = G_1^{o_1} G_2^{o_2}$ and $K_4 = G_1^{x_3} G_2^{x_4}$ and sends them to $\mathcal{U}$.

3. $\mathcal{U}$ selects a random separator $\alpha \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\tilde{K}_1 = (K_1 K_3) = G_1^{(u+o_1)} G_2^{(s+o_2)} \tag{3.62}$$

$$\tilde{K}_2 = K_2 K_4^{1/\alpha} = G_1^{x_1 + \frac{x_3}{\alpha}} G_2^{x_2 + \frac{x_4}{\alpha}}. \tag{3.63}$$

4. $\mathcal{U}$ selects random $a \xleftarrow{\$} \mathbb{Z}_p$, computes in $\mathbb{Z}_p$ that $b := y + ax$, and sets $\tilde{T} := \tilde{A} h_1^a$.

5. $\mathcal{U}$ executes the following signature of knowledge protocol

$$\begin{aligned} &\mathrm{SPK}\Big\{(x, u, s, o_1, o_2, x_1, x_2, x_3, x_4, a, b) : \\ &\tilde{K}_1 = G_1^{(u+o_1)} G_2^{(s+o_2)} \wedge \tilde{K}_2 = G_1^{x_1 + \frac{x_3}{\alpha}} G_2^{x_2 + \frac{x_4}{\alpha}} \\ &\wedge e(T, g_2)^{-x} e(h_1, g_2)^b e(h_1, w)^a e(h_2, g_2)^{u+o_1} e(h_3, g_2)^{s+o_2} \\ &e(h_4, g_2)^{x_1+x_3} e(h_5, g_2)^{x_2+x_4} = \frac{e(T, w)}{e(g_1, g_2)} \Big\}(m). \end{aligned} \tag{3.64}$$

   (a) $\mathcal{U}$ randomly selects $r_x, r_u, r_a, r_b, r_s, r_{x_1}, r_{x_2} \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$R_1 = G_1^{r_u} G_2^{r_s}, \quad R_2 = G_1^{r_{x_1}} G_2^{r_{x_2}} \tag{3.65}$$

   (b) $\mathcal{O}$ randomly selects $r_{o_1}, r_{o_2}, r_{x_3}, r_{x_4} \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\begin{aligned} R_3 &= G_1^{r_{o_1}} G_2^{r_{o_2}}, R_4 = G_1^{r_{x_4}} G_2^{r_{x_3}} \\ R_6 &= e(h_2, g_2)^{r_{o_1}} \cdot e(h_3, g_2)^{r_{o_2}} \\ &\quad \cdot e(h_4, g_2)^{r_{x_3}} \cdot e(h_5, g_2)^{r_{x_4}} \end{aligned} \tag{3.66}$$

   and sends these values to $\mathcal{U}$.

(c) In order to achieve the unlinkability of views, $\mathcal{U}$ selects another random separator[1] $\epsilon \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$\tilde{R}_1 = R_1 R_3^{\alpha} \tilde{K}_1 \tag{3.67}$$

$$\tilde{R}_2 = R_2 R_4 \tilde{K}_2 K_3^{\epsilon} \tag{3.68}$$

$$\begin{aligned}\tilde{R}_3 = {} & e(T, g_2)^{-r_x} \cdot e(h_2, g_2)^{r_u} \\ & \cdot e(h_3, g_2)^{r_s} \cdot e(h_4, g_2)^{r_{x_1}} \cdot e(h_5, g_2)^{r_{x_2}} \\ & \cdot \cdot e(h_1, g_2)^{r_b} \cdot e(h_1, w)^{r_a} \cdot R_6 \cdot (\tilde{K}_1 \cdot \tilde{K}_2)^{\epsilon}.\end{aligned} \tag{3.69}$$

(d) $\mathcal{U}$ computes $c_1 \leftarrow \mathcal{H}(pp, \tilde{K}_1, \tilde{K}_2, \tilde{T}, \tilde{R}_1, \tilde{R}_2, \tilde{R}_3, m)$.

(e) $\mathcal{U}$ computes $c_1' := \frac{1}{\alpha}(c_1 + \epsilon)$ and sends it to $\mathcal{O}$.

(f) $\mathcal{O}$ looks up o_2 in its secure storage. If o_2 is not found, it aborts. Otherwise, it computes in $\mathbb{Z}_p$

$$\begin{aligned}& z_{o_1} := r_{o_1} + c_1' o_1, \quad z_{o_2} := r_{o_2} + c_1' o_2, \\ & z_{x_3} := r_{x_3} + c_1' x_3, \quad z_{x_4} := r_{x_4} + c_1' x_4.],\end{aligned} \tag{3.70}$$

sends these to $\mathcal{U}$, and securely deletes o_2.

(g) $\mathcal{U}$ computes in $\mathbb{Z}_p$

$$\begin{aligned}z_x &:= r_x + c_1 x, \quad \tilde{z}_u := r_u + c_1 u + \alpha z_{o_1}, \\ z_a &:= r_a + c_1 a, \quad z_b := r_b + c_1 b, \\ \tilde{z}_s &:= r_s + c_1 s + \alpha z_{o_2}, \quad \tilde{z}_{x_1} := r_{x_1} + c_1 x_1 + z_{x_3}, \\ \tilde{z}_{x_2} &:= r_{x_2} + c_1 x_2 + z_{x_4}\end{aligned} \tag{3.71}$$

6. $\mathcal{U}$ sets $\tilde{\sigma} := (\tilde{K}_1, \tilde{K}_2, \tilde{T}, c_1, z_x, \tilde{z}_u, z_a, z_b, \tilde{z}_s, \tilde{z}_{x_1}, \tilde{z}_{x_2})$.

7. If any of the zero-knowledge proofs computed in the previous step fail, it outputs $\tilde{\sigma} := \perp$.

[1]This is a known technique used, for instance, in Brands' e-cash scheme [Bra93] and the blind signature-based Open-cash scheme [HO24].

8. $\mathcal{U}$ sends $(\tilde{\sigma}, m)$ to the shop $\mathcal{S}$.

9. $\mathcal{S}$ first verifies the basic validity of the received elements

$$\tilde{K}_1, \tilde{K}_2 \in \mathbb{G}, \quad \tilde{T} \in \mathbb{G}_1,$$
$$z_x, \tilde{z}_u, z_a, z_b, \tilde{z}_s, \tilde{z}_{x_1}, \tilde{z}_{x_2} \in \mathbb{Z}_p. \tag{3.72}$$

10. $\mathcal{S}$ computes the challenge $c_2 \leftarrow \mathcal{H}(\tilde{\sigma}, m)$ and sends it to $\mathcal{U}$.

11. Upon receiving c_2, $\mathcal{U}$ calculates

$$d_1 = c_2 u + x_1 \tag{3.73}$$
$$d_2 = c_2 s + x_2 \tag{3.74}$$
$$c_2' = \alpha(c_2 + \epsilon) \tag{3.75}$$

and forwards c_2' to $\mathcal{O}$.

12. $\mathcal{O}$ calculates

$$d_3 = c_2' o_1 + x_3 \tag{3.76}$$
$$d_4 = c_2' o_2 + x_4 \tag{3.77}$$

then sends the pair (d_3, d_4) back to $\mathcal{U}$.

13. $\mathcal{U}$ verifies (d_3, d_4) and calculates

$$\tilde{d}_1 = d_1 + d_3/\alpha \tag{3.78}$$
$$\tilde{d}_2 = d_2 + d_4/\alpha \tag{3.79}$$

14. $\mathcal{S}$ computes

$$\tilde{R}_1' := G_1^{z_u} G_2^{z_s} \tilde{K}_1^{-c_2}, \tag{3.80}$$

$$\tilde{R}_2' := G_1^{z_{x_1}} G_2^{z_{x_2}} \tilde{K}_2^{-c_2}, \tag{3.81}$$

$$\begin{aligned}\tilde{R}_3' := {} & e(T, g_2)^{-z_x} \cdot e(h_2, g_2)^{z_u} \cdot e(h_3, g_2)^{z_s} \\ & \cdot e(h_4, g_2)^{z_{x_1}} \cdot e(h_5, g_2)^{z_{x_2}} \\ & \cdot e(h_1, g_2)^{z_b} \cdot e(h_1, w)^{z_a} \cdot \left(\frac{e(g_1, g_2)}{e(T, w)}\right)^{c_1}.\end{aligned} \tag{3.82}$$

15. Finally, $\mathcal{S}$ verifies the following equations;

$$c_1 \stackrel{?}{=} \mathcal{H}(pp, \tilde{K}_1, \tilde{K}_2, \tilde{T}, \tilde{R}_1', \tilde{R}_2', \tilde{R}_3', m) \tag{3.83}$$

$$c_2 \leftarrow \mathcal{H}(\tilde{\sigma}, m) \tag{3.84}$$

$$G_1^{\tilde{d}_1} G_2^{\tilde{d}_2} \stackrel{?}{=} \tilde{K}_1^{c_2} \tilde{K}_2. \tag{3.85}$$

If all verifications succeed, the shop accepts the coin as valid; otherwise, it rejects the transaction.

Deposit:

The Deposit protocol remains identical to Section 3.4. Since the shop only receives $\tilde{\sigma}$ and associated transaction data from the user, it forwards these to the bank following the previously established process. The bank then verifies the coin's validity and ensures that double-spending has not occurred, subsequently crediting the shop's account accordingly.

Finally, I briefly discuss the additional security properties introduced in Section 3.2 that are enabled by the observer's involvement.

Double-Spend Prevention The Open-cash scheme ensures that a coin cannot be spent more than once under the assumption that the AESP is uncompromised. Due to the integration of a tamper-resistant observer (see Section 3.5), this property follows straightforwardly, and I therefore omit a detailed formal proof. Interested readers are referred to my earlier work [HO24] for an explicit proof. Additionally, even if the

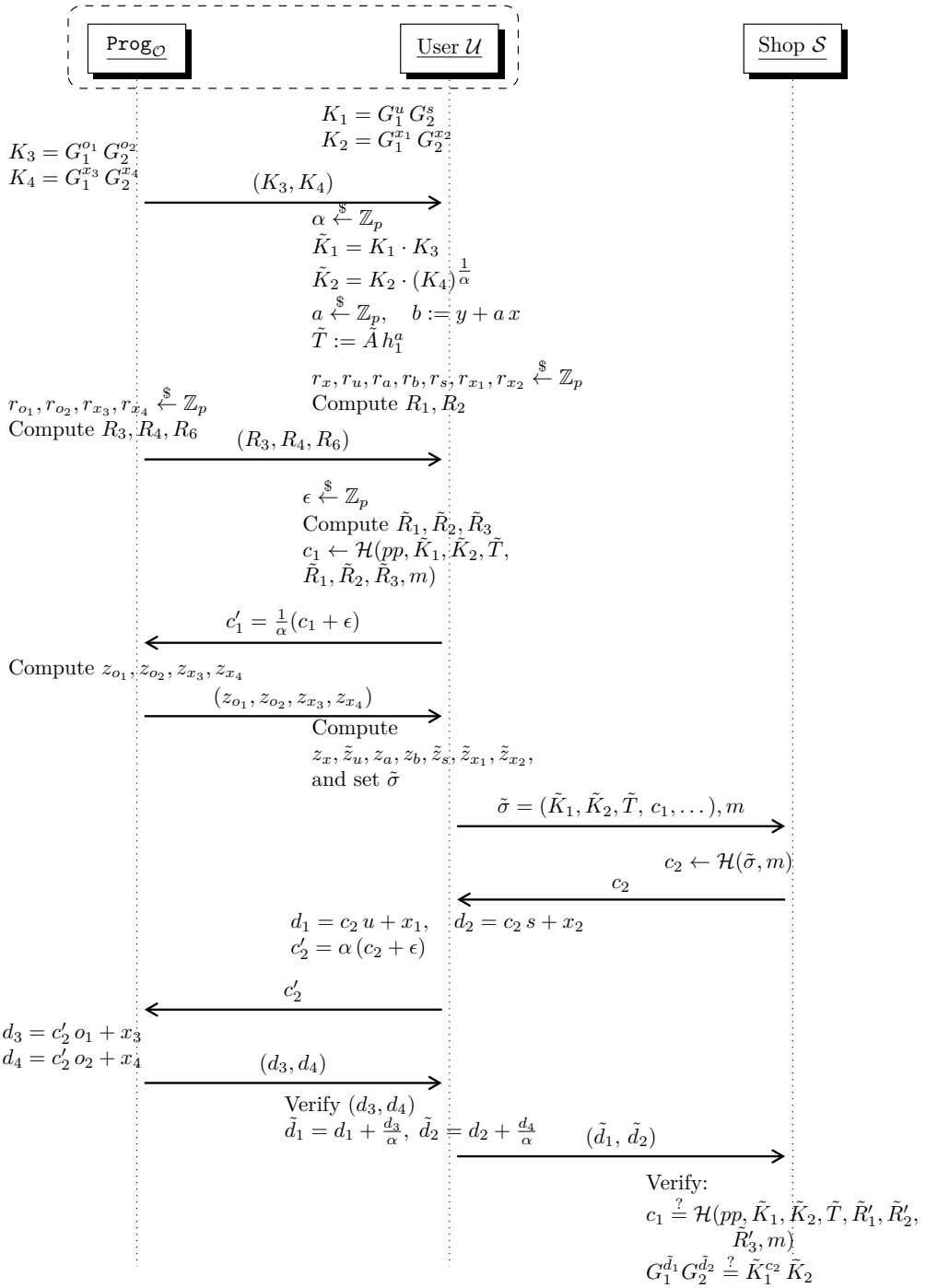

Figure 3.6: Spend protocol with observer

AESP is compromised, the system maintains double-spender traceability, enabling the identification of any user who attempts multiple spends with the same coin once the corresponding transaction transcripts are deposited.

Reversibility An important feature of this scheme is that a verifier cannot distinguish whether a proof originates solely from a single prover or jointly from a user–observer pair. Consequently, if the observer within the tamper-resistant hardware is compromised, resulting in the leakage of secret information, the extended protocol effectively reverts to the basic Open-cash scheme described in Section 3.4. Under such conditions, the scheme still preserves anonymity, one-more unforgeability, and double-spender traceability. Thus, there is no increased risk of users avoiding detection if they attempt double-spending.

Unlinkability of Views The unlinkability property is achieved by carefully randomizing parameters shared with the observer in the *Spend* protocol using two random separators, denoted by α and ϵ. This ensures that the observer's internal view cannot be correlated with the views held by the bank or the shop. In other words, each participant sees only randomized parameters, preventing collusion from revealing sensitive user data or private keys. This randomization technique has previously been employed in Brands' e-cash scheme [Bra93] as well as in my earlier blind-signature-based Open-cash proposal [HO24].

3.6 Conclusion

In this chapter, I presented *Open-cash*, an electronic cash scheme that leverages BBS+ signatures and incorporates the novel concept of *open-source observer* installed and executed within an AESP. The approach not only achieves fundamental properties of anonymity and one-more unforgeability, but also provides double-spender traceability and proactive double-spend prevention, thereby meeting practical regulatory demands by enforcing compliant behavior at the protocol level.

I began by proposing a basic Open-cash construction without an observer, ensuring that even if the AESP assumptions are compromised, the scheme gracefully reverts

to a standard e-cash scheme with guaranteed security. I then extended the design to include an observer that introduces proactive *double-spend prevention*: the observer's partial cryptographic involvement in coin issuance and spending protocols prevents illicit actions *before* they can succeed. Moreover, by distributing the observer program as open-source, banks (or regulatory bodies) maintain transparency, and the AESP-based attestation securely binds the observer program's hash to its outputs, ensuring that an unmodified and publicly inspected observer is indeed running in the user's secure element.

I believe the Open-cash scheme bridges the gap between strong cryptographic anonymity and practical oversight in modern digital payments.

In future work, I plan to extend the current two-prover (user and observer) Spend protocol into a more generalized *multi-prover* design. The existing scheme already allows both the user and the observer to jointly prove their respective witnesses within a single SPK. Generalizing this protocol to accommodate N distinct provers would further expand the applicability of Open-cash to scenarios involving multiple stakeholders. Additionally, while the current Open-cash scheme is based on BBS+ signatures, exploring alternative cryptographic primitives may lead to valuable insights. Investigating the feasibility, efficiency, and practical trade-offs of constructing Open-cash-like protocols with different primitives would be a meaningful direction for future research. Moreover, as the scheme is already built upon BBS+ signatures, I aim to explore seamless integration with other BBS+ based applications, such as verifiable credentials (VCs) and decentralized identifiers (DIDs). Such integration would facilitate advanced interoperability, paving the way for a broader class of secure, privacy-preserving, and compliant payment and identity solutions.

Chapter 4

Fair-Anonymity: A Novel Fairness Notion for Cryptocurrency

We propose a novel scheme named "Fair-Anonymity", which enables authorities to probabilistically trace IDs based on the value of a paid coin of an e-cash system. In the Fair-Anonymity scheme, the probability of anonymity being nullified increases as the transfer amount approaches the threshold, rendering the aforementioned evasion techniques ineffective. Furthermore, even if coins (or transactions) are split, the probability function can be designed so that the sum of probabilities remains consistent with the original transaction, preserving this key property. Fair-Anonymity is also theoretically applicable to any blockchain tokens such as Bitcoin and Ethereum, in which the transfer amount in a transaction is regarded as a coin. The transcript of each execution of the protocol leaves a cryptographic trace on the blockchain, serving as an indelible credential of a legitimate transaction. It provides a flexible tracing mechanism where authorities can trace users with a pre-agreed probability that is automatically determined based on the transaction amount. This probability cannot be manipulated, as the final probability depends solely on the transferred amount, remaining invariant even when transactions are split into smaller denominations. This enables a balance between regulatory requirements, such as anti-money laundering and taxation, and the protection of user privacy.

In the Fair-Anonymity scheme, users can participate by registering their identity with the authorities and obtaining an ID issued by them. The Fair-Anonymity system

supports the decentralization of authorities through cryptographic techniques involving multiple organizations, thereby reducing dependence on a single third party. To construct the Fair-Anonymity scheme, I introduced a new k-out-of-n Committed Oblivious Transfer as a variant of the efficient 2-round OT_n^k protocol proposed by Lai et al. [Lai+18], enabling practical performance.

The probabilistic nature of the Fair-Anonymity protocol effectively disincentivizes high-value illegal transactions without compromising the convenience of honest users. Criminals, even with a small probability, are deterred by the risk of their identities being traced by authorities, whereas legitimate users remain largely unaffected by occasional transaction tracking.

The contributions of this work are as follows:

1. Fairness in Transaction Amounts:

   I introduce a novel concept of "Fairness" as a property where the probability of revealing a user's (payer's) identity is determined solely by the total transaction amount, regardless of how the transaction is divided. This ensures that the probability remains unaffected by multiple low-value transactions, thereby mitigating split attacks. Fairness can be incorporated into existing transactions as accompanying proof information, making it adaptable to various blockchains. The accompanying proof with the transaction can be publicly verified, while only the designated authority can trace the user's identity based on a probability determined by the total transaction amount.

2. Achieving Perfect Fairness with Exponential Saturation Functions:

   I demonstrate that by utilizing Exponential Saturation Functions (ESFs), I can achieve "Perfect Fairness," where identity disclosure is determined exclusively by the total transaction amount. In constructing Fair-Anonymity, the ESF parameters must be approximated with integers, introducing an error ϵ. I identify a trade-off between minimizing this error and reducing the proof size.

3. Modification of Committed Oblivious Transfer (COT):

   I propose improvements to the original efficient k-out-of-n Oblivious Transfer (COT) protocol [Lai+18]. My modification enables public verification of the cor-

rectness of the message without revealing itself, while ensuring that the probability determined by the transaction amount cannot be manipulated.

With this "fair" probabilistic ID tracing scheme, I achieve the following outcomes: (1) Malicious users will fear that even a single illicit transaction could be detected, leading to their identification by authorities. (2) Honest users, on the other hand, will not be negatively impacted by probabilistic tracing, as not all of their transactions will be monitored or linked.

In scenarios where payers use this protocol over a long period, the probability of tracing their ID increases due to the accumulation of transaction amounts. However, this can be mitigated by periodically refreshing the scheme, as described in [Tom+22]. This creates a situation where malicious users are disincentivized from using Fair-Anonymity for transactions, while honest users are encouraged to adopt this scheme. Consequently, Fair-Anonymity can establish a crime-free, transparent economic zone, clearly separated from the illicit economic sphere.

4.1 k-out-of-n Committed Oblivious Transfer

Oblivious Transfer (OT) is a crucial component in constructing protocols that are secure and protect privacy, such as contract signing, private information retrieval, and secure function evaluation. An OT scheme is a two-party protocol between a sender S and a receiver R, where the sender possesses multiple secrets, and the receiver wishes to select and obtain some of them. The receiver acquires the secrets without revealing their choice to the sender, and the sender remains unaware of which secrets the receiver obtained. The first OT scheme was proposed by Rabin [Rab81], and since then, more general forms such as 1-out-of-2 OT (OT_2^1), 1-out-of-n OT(OT_n^1), and general k-out-of-n OT (OT_n^k) have been introduced [CT05; Guo+13; GMS14; NP05; NP99; CZ05; ZW05].

Committed Oblivious Transfer (COT) is an extension of OT that additionally involves commitments. In COT, the sender is committed to the input messages and the receiver is committed to the choice index before the OT protocol is executed. The COT protocol provides additional security guarantees compared to plain OT. The commit-

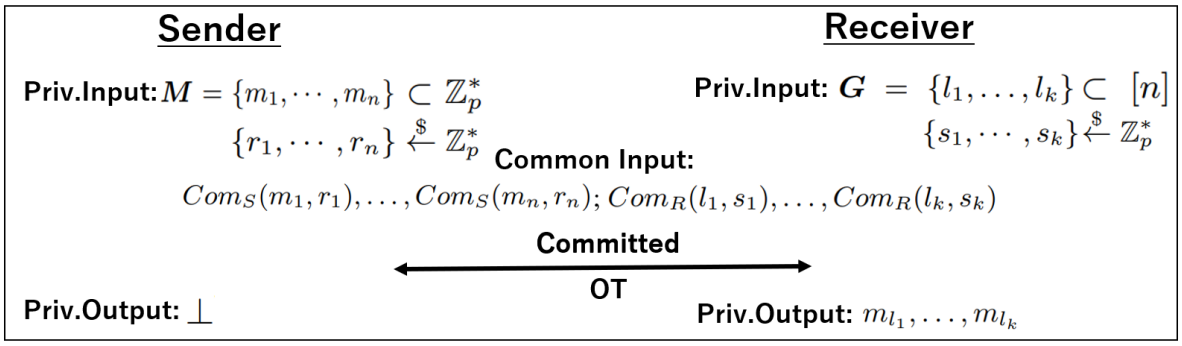

Figure 4.1: Overview of k-out-of-n Commited Oblivious Transfer

ments prevent the sender from changing the messages and the receiver from changing the choice index during the protocol. I introduce the new schemes of k-out-of-n Committed Oblivious Transfer (COT_n^k) with a small modification on the efficient 2-round OT_n^k protocol proposed by Lai et al. [Lai+18].

4.1.1 Definitions of k-out-of-n Committed Oblivious Scheme

In the following text, we denote by $\mathcal{M}$ and $\mathcal{C}$ a set of all messages and a set of all commitments, respectively.

The setup algorithm $ck \leftarrow \mathcal{G}(1^\lambda)$ generates a commitment key ck. The commitment key specifies a message space $\mathcal{M}$ and commitment space $\mathcal{C}$. The commitment algorithm combined with the commitment key ck specifies a function $\text{Com}_{ck} : \mathcal{M} \to \mathcal{C}$.

Suppose we have a sender S and a receiver R. A set of messages $\mathbb{M} = \{m_1, \cdots, m_n\}$ is held by the sender and a set of indices $\mathbb{G} = \{l_1, \cdots, l_k\}$ where $k \leq n$ are chosen by the receiver. We assume that there exists a probabilistic key generation algorithm Gen which takes as input 1^λ and outputs a commitment key ck, where λ is a security parameter, a probabilistic commitment generation algorithm $\text{Com}_{ck} : \mathcal{M} \to \mathcal{C}$ which takes as inputs a message and a commitment key ck and outputs a commitment. Note that the generated commitment key ck is a part of public parameters.

Definition 4.1 (k-out-of-n Committed Oblivious Transfer (COT_n^k))**.** *A COT_n^k protocol is run between the parties S and R. At the beginning there exist public commitments $\text{Com}_S(m_1, r_1), \ldots, \text{Com}_S(m_n, r_n)$, and $\text{Com}_R(l_1, s_1), \ldots, \text{Com}_R(l_k, s_k)$. S inputs $m_1, \ldots, m_n$ and $r_1, \ldots, r_n$, while R inputs $l_1, \ldots, l_k$ and $s_1, \ldots, s_k$. At the end of the*

protocol, R receives $m_{l_1}, \ldots, m_{l_k}$. S learns nothing about $l_1, \ldots, l_k$ while R has no clue about unchosen messages.

The overview of COT_n^k is shown in Fig. 4.1.

Theorem 4.1. *Suppose the Diffie-Hellman Exponent assumption [BBG05] holds, there exist efficient k-out-of-n Committed Oblivious Transfer protocols.*

Furthermore, to construct the Fair-Anonymity protocol (in Section 4.3), I slightly modify the k-out-of-n COT and define k-out-of-n Committed Oblivious Transfer of Commitments ($\mathrm{C^2OT}_n^k$). In $\mathrm{C^2OT}_n^k$, the sender sends messages in plain text, and the receiver receives the corresponding ciphertexts.

Definition 4.2 (k-out-of-n Committed Oblivious Transfer of Commitments ($\mathrm{C^2OT}_n^k$)). *A $\mathrm{C^2OT}_n^k$ protocol is run between the parties S and R. At the beginning, both parties take a generator g_c and commitments $\mathrm{Com}_S(m_1, r_1), \ldots, \mathrm{Com}_S(m_n, r_n)$, and $\mathrm{Com}_R(l_1, s_1), \ldots, \mathrm{Com}_R(l_k, s_k)$ as public inputs. S inputs $m_1, \ldots, m_n$ and $r_1, \ldots, r_n$, while R inputs $l_1, \ldots, l_k$ and $s_1, \ldots, s_k$. At the end of the protocol, R receives ${g_c}^{m_{l_1}}, \ldots, {g_c}^{m_{l_k}}$. S learns nothing about $l_1, \ldots, l_k$ while R has no clue about unchosen messages.*

It becomes verifiable that the original messages of the ciphertexts received by the receiver are equal to the ones sent by the sender. This enables the public verifier in the Fair-Anonymity protocol to verify that the received ciphertexts correspond to the original messages without directly knowing the original messages.

4.1.2 Construction of COT_n^k scheme

In this subsection, I present the concrete construction of the COT_n^k scheme.

Overview of technique

Let $\mathbb{BG} = (\mathbb{G}, \mathbb{G}_T, e, p)$ be a bilinear group with two generators $g, h \in \mathbb{G}$. α is known by the system generator only. In this setting, the token $\mathrm{P}(\mathbb{G}) = g^{\overline{\mathrm{s}}{}^{\overline{(\alpha+l_1)(\alpha+l_2)\cdots(\alpha+l_k)}}}$ and each secret m_i is encrypted using a key $e\left(g^{\frac{1}{\alpha+i}}, h\right)^r$, that is $\tilde{c}_i = e\left(g^{\frac{1}{\alpha+i}}, h\right)^r \cdot m_i$. If $i \in \mathbb{G}, h^{\frac{(\alpha+l_1)(\alpha+l_2)\cdots(\alpha+l_k)}{(\alpha+i)}}$ is computable from the system parameters. Thus, the decryptor

can retrieve the encryption (decryption) key $e\left(g^{\frac{1}{\alpha+i}}, h\right)^r$ together with $\mathrm{P}(\mathbb{G})^r$ and its private key s. If $i \notin \mathbb{G}$, the value of $h^{\frac{(\alpha+l_1)(\alpha+l_2)\cdots(\alpha+l_k)}{(\alpha+i)}}$ cannot be computed. Therefore, the decryptor is unable to retrieve the encryption key and get the corresponding secret.

Construction

A trusted third party establishes the system by choosing a security parameter λ and a random α as the system secret key, and generates the system parameter $SP = (\mathbb{BG}, g, h, g_1, g_2, \cdots, g_n, h_1, h_2, \cdots, h_n)$, which is public known.

In the first round, the receiver chooses a random $s \in \mathbb{Z}_p^*$ as its secret key and a set $\mathbb{G} = \{l_1, l_2, \cdots, l_k\}$. It then uses the Aggregation algorithm to compute a token $\mathrm{P}(\mathbb{G}) = g^{\frac{s}{(\alpha+l_1)(\alpha+l_2)\cdots(\alpha+l_k)}}$, and computes a proof information $\Sigma = h^{\frac{(\alpha+l_1)(\alpha+l_2)\cdots(\alpha+l_k)\cdot\alpha^{n-k}}{s}}$ for its choice set $\mathbb{G}$. In the second round, upon receiving a request from a receiver, the sender first tests whether $e(\mathrm{P}(\mathbb{G}), \Sigma) = e\left(g, h^{\alpha^{n-k}}\right)$ and aborts if no. Otherwise, S picks a random $r \in \mathbb{Z}_p^*$ and computes a ciphertext CT for the secrets as $\tilde{c}_0 = \mathrm{P}(\mathbb{G})^r = g^{\frac{rs}{(\alpha+l_1)(\alpha+l_2)\cdots(\alpha+l_k)}}$ together with, for each $i = 1, 2, \cdots, n : \tilde{c}_i = e\left(g^{\frac{1}{\alpha+i}}, h\right)^r \cdot m_i$. Once receiving the encrypted secrets CT from the sender, the receiver, for each $i \in \mathbb{G}$, computes $m_i = \tilde{c}_i \cdot e\left(\tilde{c}_0, h^{\frac{(\alpha+l_1)\cdots(\alpha+l_n)}{(\alpha+i)}}\right)$. After the decryption, R gets only k secrets with indexes in $\mathbb{G}$ from S. After that, R outputs new commitments so that the verification for all of the received messages can be executed using membership proof.

The concrete COT_n^k scheme is illustrated as follows:

- **Inputs:**
  - System parameter SP. A trusted third party runs the Setup algorithm as follows. Given a security parameter λ, this algorithm generates a bilinear group $\mathbb{BG} = (\mathbb{G}, \mathbb{G}_T, e, p)$ with two generators $g, h \in \mathbb{G}$. Then it randomly chooses $\alpha \in \mathbb{Z}_p^*$ as the system secret key and computes $g_i = g^{\frac{1}{\alpha+i}}, h_i = h^{\alpha^i}$ for all $i \in [n]$. The system parameter SP consists of $(\mathbb{BG}, g, h, g_1, g_2, \cdots, g_n, h_1, h_2, \cdots, h_n)$.
  - S holds a set of secrets $\mathbb{M} = \{m_1, \cdots, m_n\} \subset \mathbb{Z}_p^*$ and random numbers $\{r_1, \cdots, r_n\} \xleftarrow{\$} \mathbb{Z}_p^*$.
  - R holds his choice set $\mathbb{G} = \{l_1, \ldots, l_k\} \subset [n]$ and random numbers $\{s_1, \cdots, s_k\} \in \mathbb{Z}_p^*$.

– Input Common Commitments:
$Com_S(m_1, r_1), \ldots, Com_S(m_n, r_n)$ and $Com_R(l_1, s_1), \ldots, Com_R(l_k, s_k)$ are common inputs [1].

- **Execute OT_n^k Protocol:**

1. $R \rightarrow S$: Given a choice set $\mathbb{G} = \{l_1, \cdots, l_k\}$ and the system parameters SP, R picks a random $s \in \mathbb{Z}_p^*$ as his secret key sk and uses the Aggregation algorithm [DPP07] to compute $\mathrm{P}(\mathbb{G})$ together with Σ where

$$\mathrm{P}(\mathbb{G}) = g^{\frac{s}{(\alpha+l_1)(\alpha+l_2)\cdots(\alpha+l_k)}}, \tag{4.1}$$

$$\Sigma = h^{\frac{(\alpha+l_1)(\alpha+l_2)\cdots(\alpha+l_k)\cdot\alpha^{n-k}}{s}}. \tag{4.2}$$

2. $S \rightarrow R$: S runs the Encrypt algorithm and generates commitments as follows. Given a set $T = (\mathrm{P}(\mathbb{G}), \Sigma)$, a set of secrets $\mathbb{M} = \{m_1, \cdots, m_n\}$ and the system parameter SP, it first performs the verification algorithm as: $e(\mathrm{P}(\mathbb{G}), \Sigma) = e\left(g, h^{\alpha^{n-k}}\right)$. If the equation does not hold, it aborts. Otherwise, it accepts $|\mathbb{G}| = k$. Then for a random parameter $r \xleftarrow{\$} \mathbb{Z}_p^*$, it computes the ciphertext set $CT = \{\tilde{c}_i\}_{i=1,\ldots,n}$ for the messages as

$$\tilde{c}_0 = \mathrm{P}(\mathbb{G})^r = g^{\frac{rs}{(\alpha+l_1)(\alpha+l_2)\cdots(\alpha+l_k)}} \tag{4.3}$$

together with, for each $i \in [n]$:

$$\tilde{c}_i = e\left(g^{\frac{1}{\alpha+i}}, h\right)^r \cdot m_i \tag{4.4}$$

$$\tag{4.5}$$

3. R decrypts the received ciphertexts as follows. Given the ciphertext $CT = \{\tilde{c}_0, \tilde{c}_1, \cdots, \tilde{c}_n\}$, a choice set $\mathbb{G} = \{l_1, \cdots, l_k\}$, a secret key s and the system

[1] S and R should perform the PoK for S : $((m_i, r_i); Com_S(m_i, r_i))$ for all $i \in [n]$ and the PoK for R : $((l_i, s_i); Com_R(l_i, s_i))$ for all $i \in [k]$ respectively

parameter SP, only for each $i \in \mathbb{G}, R$ can compute

$$m_i = \tilde{c}_i \cdot e\left(\tilde{c}_0, h^{\frac{(\alpha+l_1)(\alpha+l_2)\cdots(\alpha+l_k)}{(\alpha+i)}}\right)^{-\frac{1}{s}} \tag{4.6}$$

$$\tag{4.7}$$

- **Outputs and Verification:** R verifies that the values of the received messages are the same as the values of committed ones.

4.2 Fairness

In this section, I define a novel concept of *Fairness* and show that a specific function satisfies the definition of the fairness property. This property is extremely important in that a Fair-Anonymity scheme constructed using a probability distribution function satisfying this property enables authorities to trace the sender's ID with a probability that depends only on the transfer amount, regardless of the division of coins (splitting of transfer transactions).

4.2.1 Definition of Fairness

Let $\mathbb{C}$ and $\mathbb{U}$ be the sets of all coins and all users, respectively. For any coins $c \in \mathbb{C}$, we denote by $c.v$ the value of the coin c. A payment system is said to be *fair* if the following condition is satisfied.

Definition 4.3 (ϵ-Fairness)**.** *Suppose we have a probabilistic extractor $\mathcal{E} : \mathbb{C} \to \mathbb{U}$ or $\perp$ that extracts from any coin c the spender's user id $u \in \mathbb{U}$ when successful or nothing $\perp$ otherwise.*

*Then, for all $\epsilon \in \mathbb{R}_{\geq 0}$, we say that a payment system satisfies ϵ-**fairness** if for all coins $c_1, c_2, c_3 \in \mathbb{C}$ it holds:*

$$c_1.v = c_2.v + c_3.v \Rightarrow |\Pr[\mathcal{E}(c_1) \neq \perp] - \Pr[\mathcal{E}(c_2) \neq \perp \vee \mathcal{E}(c_3) \neq \perp]| \leq \epsilon \tag{4.8}$$

When $\epsilon = 0$, we say that a payment system satisfies perfect fairness.

The intuition of this definition is the following. Any division of coins or payments does not affect the probabilistic traceability with at most ϵ fluctuation. ϵ-Fairness is about ensuring that the process of identifying the spender from a coin is not affected by the division or combination of coins, within a certain tolerance level represented by ϵ. The smaller the ϵ, the fairer the system is considered to be.

4.2.2 Exponential Saturation Function

Next, I introduce the exponential saturation function which satisfies the condition of the fairness property. The form of the function is

$$p(x) = 1 - e^{-\frac{x}{K}} \tag{4.9}$$

where $K \in \mathbb{R}$ is a rate constant.

Lemma 4.1. *Suppose we have a payment system with a probabilistic extractor $\mathcal{E} : \mathbb{C} \to \mathbb{U}$ or $\perp$ as in Definition 4.3 such that its probability is determined by exponential saturation function of the input coin value. More concretely, we define a probability function $p : \mathbb{R}_{\geq 0} \to [0, 1]$ as*

$$p(c.v) = \Pr[\mathcal{E}(c) \neq \perp] = 1 - e^{-\frac{c.v}{K}} \tag{4.10}$$

for some rate constant $K \in \mathbb{R}_{\geq 0}$.

Then, for any $n > 0$ division of coin c to $c_1, \ldots, c_n$ such that $c.v = c_1.v + \cdots + c_n.v$, the payment system satisfies perfect fairness for all $c.v, K \in \mathbb{R}_{\geq 0}$.

Proof. By using probability of complementary event that $p(X \cup Y) = 1-(1-p(X))(1-p(Y))$ for independent variables X, Y. We have

$$\begin{aligned} p(c.v) &= 1 - e^{-\frac{c.v}{K}} \\ &= 1 - \prod_{i=1}^{n} (1 - p(c_i.v)) \\ \Leftrightarrow 1 - p(c.v) &= \prod_{i=1}^{n} (1 - p(c_i.v)) \\ \log(1 - p(c.v)) &= \sum_{i=1}^{n} \log (1 - p(c_i.v)) \end{aligned} \tag{4.11}$$

By substituing p with exponential sturation function as:

$$p(c_i.v) = 1 - e^{-\frac{c_i.v}{K}}, \quad p(c.v) = 1 - e^{-\frac{c.v}{K}}$$
$$\log(e^{-\frac{c.v}{K}}) = \sum_{i=1}^{n} \log\left(e^{-\frac{c_i.v}{K}}\right)$$
$$\therefore c.v = \sum_{i=1}^{n} c_i.v \tag{4.12}$$

□

4.3 Fair-Anonymity

In this section, we propose the Fair-Anonymity scheme constructed by combining the construction of COT_n^k with the exponential saturation function satisfying the fairness property.

4.3.1 Overview of Fair-Anonymity Protocols

The Fair-Anonymity protocol consists of System Setup, Registration Protocol, Execution Protocol, and Tracing Protocol.

- **System Setup:** First, in the setup phase, the authority generates a set of IDs using a salt known only to itself. Here, the authority can be a single organization or a decentralized system that requires consensus among multiple organizations using cryptographic techniques. Since the set of IDs is generated using a salt known only to the authority, third parties cannot compute them.

- **Registration Protocol:** Users who want to use the Fair-Anonymity protocol interact with the authority to prove their identity and obtain an ID. This is equivalent to opening a bank account. This can be done in a privacy-preserving manner for users, such as using Verifiable Credentials instead of real identities.

- **Execution Protocol:** In the execution protocol, the user interacts online with a verifier (on the blockchain). The authority is offline and not involved in this interaction. First, the user and the verifier execute COT, where the user is the

sender and the verifier is the receiver. The user selects a set of messages consisting of either the encryption of their ID or random values, and keeps them secret. The ratio of the ID's ciphertext to random values is specified by a probability distribution function that satisfies fairness, such as an exponential saturation function (introduced in Section 4.2) with the total amount as a variable (if the transfer amount is large, the ratio of the ID's ciphertext increases according to the probability function, which is an increasing function for x $\geq$ 0). The verifier selects k labels to receive. Then, the user sends k out of the n messages specified by the verifier using COT. Due to the properties of COT, neither the message values nor the specified label values can be changed from the initially chosen ones. If there are no issues during the protocol execution, the verifier publishes the received values (ID ciphertext or random values) as part of the transcript on the blockchain with a signature. I emphasize that the verifier and those who can see the transcript can only know the values received by the verifier, but cannot distinguish whether the values are ciphertexts or mere random values, thus preserving the user's anonymity.

- **Tracing Protocol:** In the tracing protocol, the authority can trace a specific transcript on the blockchain at any time. Since a specific transfer transaction using the Fair-Anonymity protocol includes the sender's ID ciphertext with the pre-agreed probability according to the transfer amount, the authority can identify the user by searching a narrow set of IDs using the salt known only to the authority (if it is an ID ciphertext and not a random value). Those who do not know the salt must perform an exhaustive search on the set $\mathbb{Z}_p^*$, and if the set is sufficiently large, anonymity is not compromised.

4.3.2 System Setup

Firstly, Authority $\mathcal{A}$ generates a subset $\mathbb{U} = \{u_i\}_{i=1,\ldots,N} \subset \mathbb{Z}_p^*$, using a value of salt known only to $\mathcal{A}$. Assuming the DDH assumption holds, it is impossible to calculate the user ID from g^u for some group g for those who do not know the salt, while $\mathcal{A}$, knowing the salt, can perform a brute-force search in the $\mathbb{U}$ space. Note that the size of the set $\mathbb{U}$ is $|\mathbb{U}| \ll p$. Next, $\mathcal{A}$ generates a set of Pedersen commitments $\mathbb{A} =$

$\{Com_A(u_i, s_i)\}_{i=1,\ldots,N}$ from the set of user IDs $\mathbb{U}$ as follows:

$$Com_A(u_i, s_i) = g_T^{u_i} h_T^{s_i}, \text{ where } \quad g_T, h_T \quad \in \mathbb{G}_T \quad \text{and} \quad s_i \xleftarrow{\$} \mathbb{Z}_p^* \tag{4.13}$$

This set of commitments enables users, in the Fair-Anonymity protocol to be constructed next, to provide a zero-knowledge proof that they know their ID is in the correct ID set.

4.3.3 Registration Protocol

The Registration protocol is executed between $\mathcal{A}$ and User $\mathcal{U}$ interactively. First, $\mathcal{U}$ shows his certificate of identity to $\mathcal{A}$. $\mathcal{A}$ verifies it. After the verification, $\mathcal{A}$ picks up an ID $u \leftarrow \mathbb{U}$ for him and finds the corresponding commitment $Com_A(u_\rho, s_\rho)$ of $u = u_\rho$ at the ρ-th position in the entire set of $\mathbb{A}$. Then, $\mathcal{A}$ sends $\mathcal{U}$ the triple of (u, ρ, r_ρ).

4.3.4 Execution Protocol

Next, I give the protocol of executing Fair-Anonymity between a user $\mathcal{U}$ and a public verifier $\mathcal{V}$. The overview of this protocol is shown in Fig. 4.2. For simplicity, the protocol is constructed using $\mathrm{C^2OT}_n^k|_{k=1}$. Note that the protocol can similarly be constructed with higher efficiency for the general case where $k > l$.

Consider $\mathcal{U}$, already registered with an authority $\mathcal{A}$, executes the execution protocol with a public verifier $\mathcal{V}$ for his coin c of value $c.v$.

1. First, the pair of two integers $l, n \in \mathbb{N}$ are determined based on the probability distribution $p(v)$, which depends on the coin value $v = c.v$,

$$n, l \leftarrow \mathbb{N} \quad \text{s.t.} \quad \frac{l}{n} = p(v) + \epsilon = 1 - e^{-\frac{v}{K}} + \epsilon. \tag{4.14}$$

   where $\epsilon \in \mathbb{R}$ represents the small fluctuation due to approximating real numbers with rational numbers. Note that the probability function is one of the exponential saturation functions that satisfy the fairness property (shown in Section 4.2).

2. $\mathcal{U}$ and $\mathcal{V}$ start the $\mathrm{C^2OT}_n^1$ protocol, where Sender is $\mathcal{U}$ and Receiver is $\mathcal{R}$ in this case, as follows:

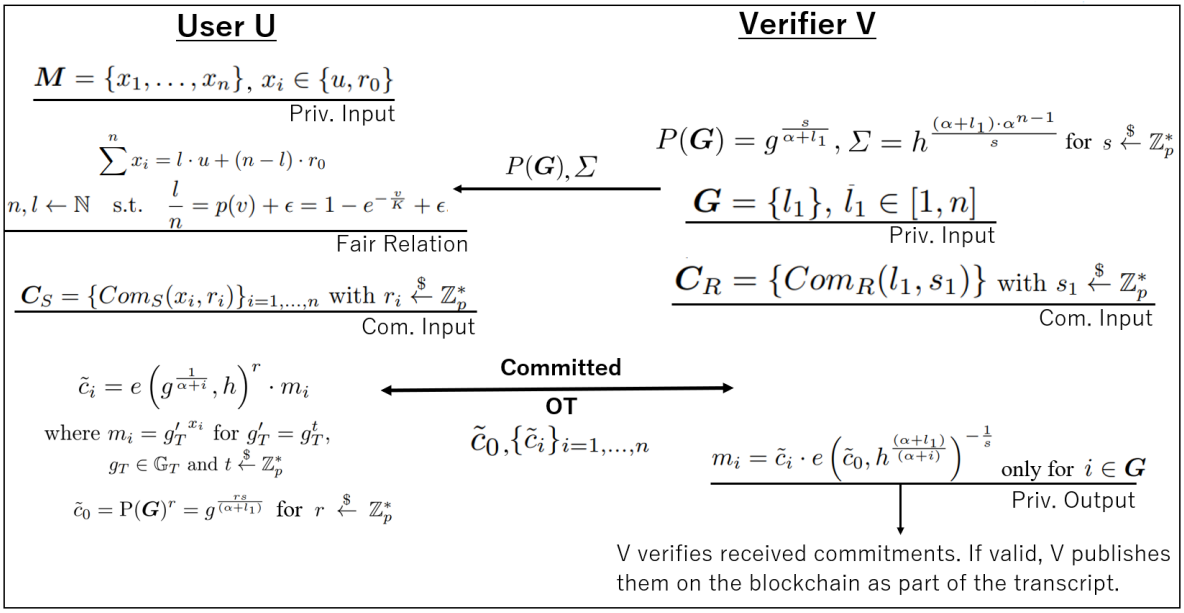

Figure 4.2: Overview of Execution Protocol of Fair-Anonymity

(a) **Input**: First, $\mathcal{U}$ chooses the set of n messages $\mathbb{M} = \{x_1, \ldots, x_n\}$ where $x_i \in \{u, r_0\}$ s.t.

$$\sum_{i=1}^{n} x_i = l \cdot u + (n-l) \cdot r_0 \tag{4.15}$$

where $u \in \mathbb{U}$ is the $\mathcal{U}$'s ID and $r_0 \in \mathbb{Z}_p^*$ is a randomly chosen number. [2] For his chosen messages, $\mathcal{U}$ calculates the corresponding commitments $\mathbb{C}_S = \{Com_S(x_i, r_i)\}_{i=1,\ldots,n}$ with $r_i \xleftarrow{\$} \mathbb{Z}_p^*$

Next, $\mathcal{V}$ chooses his choice set $\mathbb{G} = \{l_1\}$, $l_1 \in [1, n]$ (because we consider now the case $k = 1$) and calculates the commitment $\mathbb{C}_R = \{Com_R(l_1, s_1)\}$ with $s_1 \xleftarrow{\$} \mathbb{Z}_p^*$.

The two sets of the commitments $\mathbb{C}_R$ and $\mathbb{C}_S$ are common inputs.

(b) **Execute**:

- $\mathcal{V} \rightarrow \mathcal{U}$: Given a choice set $\mathbb{G} = \{l_1\}$ and the system parameters SP,$\mathcal{V}$ picks a random $s \xleftarrow{\$} \mathbb{Z}_p^*$ as his secret key and uses the Aggregation Algorithm to compute $\mathrm{P}(\mathbb{G})$ together with Σ where

$$P(\mathbb{G}) = g^{\frac{s}{\alpha + l_1}}, \quad \Sigma = h^{\frac{(\alpha+l_1)\cdot\alpha^{n-1}}{s}} \tag{4.16}$$

[2] We assumed $|\mathbb{U}| \ll p$ and hence $\Pr[r_0 \in \mathbb{U}] < \mathrm{negl}(\lambda)$.

- $\mathcal{U} \rightarrow \mathcal{V}$: $\mathcal{U}$ runs the Encrypt algorithm and generates commitments as follows. Given a set $T = (\mathrm{P}(\mathbb{G}), \Sigma)$, a set of secrets $\mathbb{M} = \{x_1, \cdots, x_n\}$ and the system parameter SP, it first performs the verification algorithm as: $e(\mathrm{P}(\mathbb{G}), \Sigma) = e\left(g, h^{\alpha^{n-k}}\right)$. If the equation does not hold, it aborts. Otherwise, it accepts $|\mathbb{G}| = k$. Then for a random parameter $r \xleftarrow{\$} \mathbb{Z}_p^*$, it computes the ciphertext set $\tilde{C} = \{\tilde{c}_i\}_{i=1,\ldots,n}$ for the messages as

$$\tilde{c}_0 = \mathrm{P}(\mathbb{G})^r = g^{\frac{rs}{(\alpha+l_1)}} \tag{4.17}$$

together with, for each $i \in [n]$:

$$\tilde{c}_i = e\left(g^{\frac{1}{\alpha+i}}, h\right)^r \cdot m_i \tag{4.18}$$

$$\text{where } m_i = {g'_T}^{x_i} \text{ for } g'_T = g_T^t,\ g_T \in \mathbb{G}_T \text{ and } t \xleftarrow{\$} \mathbb{Z}_p^*. \tag{4.19}$$

[3] Using Verifiable Encryption ,$\mathcal{U}$ performs the PoK for the $\tilde{c}_i =_{eq} Com_S(x_i)$ for all $i \in [n]$ to $\mathcal{V}$, outputting the proof set $\{\pi_i\}_{i=1,\ldots,n}$.

- $\mathcal{V}$ decrypts the received ciphertexts as follows. Given the ciphertexts $\tilde{C} = \{\tilde{c}_0, \tilde{c}_1, \cdots, \tilde{c}_n\}$, the choice set $\mathbb{G} = \{l_1\}$, the secret key s and the system parameter SP, only for the $i \in \mathbb{G}$, here just for the l_1, $\mathcal{V}$ can decrypt

$$m_i = \tilde{c}_i \cdot e\left(\tilde{c}_0, h^{\frac{(\alpha+l_1)}{(\alpha+i)}}\right)^{-\frac{1}{s}}. \tag{4.20}$$

(c) **Verify**:

$\mathcal{V}$ verifies the received proof set $\{\pi_i\}_{i=1,\ldots,n}$. If the verification fails $\mathcal{V}$ aborts.

3. Afther the execution of $\mathrm{C^2OT}_n^k$, $\mathcal{U}$ proves to $\mathcal{V}$ that the two relations $\mathscr{R}_1$ and $\mathscr{R}_2$ defined in the following Section 4.4 hold - for $\mathscr{R}_1$ all of the $\{x_i\}_{i=1,\ldots,n}$ in the commitments $\mathbb{C}_S$ are indeed selected from either u or r_0 (see the equation (4.21)

[3]Note that in this point the construction of $\mathrm{C^2OT}_n^k$ protocol is slightly different from that of the COT_n^k protocol, changing the message part encrypted by ElGamal Encryption to $g_q^{x_i}$. This enables only $\mathcal{A}$, knowing the salt, to trace the user's ID.

), and for $\mathscr{R}_2$ the equation (4.22) for the exponential saturation function holds.

4. At the end, if $\mathcal{U}$ clears all verifications, $\mathcal{V}$ publishes the results of those proofs as a transcript on the blockchain. As a result, the $\mathcal{U}$'s coin can be considered "fair" with the transcript.

4.3.5 Fair-Tracing

The Authority $\mathcal{A}$ can, at any time, refer to the transcripts of users published on the blockchain and combine these with the secret value of salt known only to $\mathcal{A}$, to probabilistically trace the ID of the user who made the coin payment. However, the actual probability of $\mathcal{A}$ being able to identify the user ID is given by $p(v) \pm \epsilon$.

4.4 Security Notions of Fair-Anonymity

In this section, I define the required security notions of Fair-Anonymity - *Completeness*, *Soundness*, *Anonymity*, and *Fair Traceability*. First, we define two cryptographic relations that Fair-Anonymity must satisfy, using four Proofs of Knowledge (PoKs). Next, we define the four security notions above. Finally, we prove that the construction of Fair-Anonymity in Section 4.3 satisfies these security notions.

First, we define two relations. In the Fair-Anonymity scheme, the probability of an authority $\mathcal{A}$ being able to trace a User's ID is determined by the value of the User's coin. During the one-time System Setup, $\mathcal{A}$ generates the set of commitments for all user IDs, $\mathbb{A} = \{Com_A(u_i)\}_{i=1,\ldots,N}$. In the Registration protocol, a user $\mathcal{U}$ presents their real identity to $\mathcal{A}$, and then sends $\mathcal{U}$ their user ID u, and the label ρ indicating the position of the commitment corresponding to u in the set $\mathbb{A}$ together with r_ρ. In the Execution phase, $\mathcal{U}$ executes probabilistic message transmission to a public verifier $\mathcal{V}$ using COT_n^k (the message being information about u or a uniform random number). In the $\mathrm{C^2OT}_n^k$, $\mathcal{U}$ chooses n messages $\{x_1, \ldots, x_n\}$ and calculates the corresponding commitment set $\mathbb{C}_S = \{Com_S(x_i)\}_{i=1,\ldots,n}$ as common input. Each message is u itself or a uniform random number r_0. After the execution of $\mathrm{C^2OT}_n^k$, $\mathcal{U}$ proves that the following two relations $\mathscr{R}_1$ and $\mathscr{R}_2$ defined below hold. $\mathscr{R}_1$ is the relation that all $\{x_i\}_{i=1,\ldots,n}$ in the commitments $\mathbb{C}_S$ are indeed values of either u or r_0. $\mathscr{R}_2$ denotes the relation for

the commitments $\mathbb{C}_S$ of n messages $\{x_i\}_{i=1,\ldots,n}$, where l of these messages have values equal to u, and the remaining $n-l$ messages are valued at r_0.

1. **Relation $\mathscr{R}_1$:** For the $\mathcal{U}$'s witness (u, r^*, ρ) and a fresh commitment $c^* = Com_{\mathcal{U}}(u, r^*) = g^u h^{r^*}$, where r^* is a fresh random number, there exists $\rho \in [N]$ such that $Com_{\mathcal{U}}(u, r^*) =_\omega Com_A(m_\rho, r_\rho)$, i.e.,

$$\begin{aligned}(u, r^*, r_\rho, r_\sigma, \rho, \sigma) \in \mathscr{R}_1 \Leftrightarrow \pi \leftarrow PoK\big((u, r^*, r_\rho, r_\sigma, \rho, \sigma); c^* = Com_{\mathcal{U}}(u, r^*) \\ \wedge {}^{\exists}\rho \in [N] \ s.t. \ c_\rho = Com_A(u, r_\rho) \\ \wedge {}^{\exists}\sigma \in [n] \ s.t. \ c_\sigma = Com_S(u, r_\sigma)\big) \\ \text{is an accepting proof.} \qquad (4.21)\end{aligned}$$

2. **Relation $\mathscr{R}_2$:** In $\mathrm{C^2OT}_n^k$ for the commitments $\mathbb{C}_S$ for massages $\{x_1, \ldots, x_n\}$, the condition $\#\{i | c^* = c_i, i \in [n]\} = l (> 0)$ should be satisfied. For $c^* = Com_{\mathcal{U}}(u, r^*)$ and $c^\dagger = Com_{\mathcal{U}}(r_0, r^\dagger)$, $\mathscr{R}_2$ is defined as:

$$\begin{aligned}(u, r^*, r_0, r^\dagger, r_1, \ldots, r_n) \in \mathscr{R}_2 \Leftrightarrow \pi \leftarrow PoK\big((u, r^*, r_0, r^\dagger, r_1, \ldots, r_n); \\ c_i =_\omega c^* \text{ or } c^\dagger \quad \text{for all } i \in [n] \\ \wedge \prod_{i=1}^{n} Com(x_i, r_i) = (c^*)^l (c^\dagger)^{n-l} Com(0, v)\big) \qquad (4.22)\end{aligned}$$

where $v = \sum_{i=1}^{n} r_i - l r^* - (n-l) r^\dagger$ and π is an accepting proof[4].

For the proofs of the relations, the following four proofs of knowledge of the relation between a witness $w \in \mathbb{W}$ and a statement $x \in \mathbb{X}$ are required, where $\mathbb{X}$ and $\mathbb{W}$ are the sets of witnesses and statements, respectively.

1. PoK-1: $((u, r^*, \rho); c^* \in_\omega \mathbb{A})$
2. PoK-2: $((u, r^*, \sigma); c^* \in_\omega \mathbb{C}_S)$
3. PoK-3: $\big((u, r_0), (r_1, \ldots, r_n); \ c_i = \mathrm{Com}(u, r_i) \text{ or } \mathrm{Com}(r_0, r_i) \text{ for all } i \in [n]\big)$
4. PoK-4: $\big((u, r_0), r^*, r^\dagger, (r_1, \ldots, r_n); \prod_{i \in [n]} c_i = (c^*)^n (c^\dagger)^{n-l} Com(0, v)\big)$
   where $v = \sum_{i=1}^{n} r_i - l \cdot r^* - (n-l) \cdot r^\dagger$

[4]The relation of $c_i =_\omega c^*$ or $c^\dagger$ can be proven with OR-Proofs [FHJ20; CDS94]

For the proof of $\mathscr{R}_1$, the PoKs of (1)-(3) are required while the proof of $\mathscr{R}_2$ requires (1),(2),(4). Let us denote by $\mathscr{R} = (\mathscr{R}_1 \wedge \mathscr{R}_2)$.

Next, we define the required security notions of the Fair-Anonymity.

Definition 4.4 (Completeness)**.** *Completeness refers to the fact that a prover can always provide a valid proof except for negligible probability for a statement with a witness. More formally, given a statement* $x = (c^*, c^\dagger, \mathbb{C}_S)$ *and a witness* $w = (u, r^*, r_0, r^\dagger, r_1, \ldots, r_n, \rho, \sigma)$*, for every honest user* $\mathcal{U}$ *and honest verifier* $\mathcal{V}$*, and for every security parameter* $\lambda > 0$*, the following holds:*

$$\Pr\left[\mathcal{V}(x, \pi) = 1 \mid \pi \leftarrow \mathcal{U}(x, w), (x, w) \in \mathscr{R}\right] \geq 1 - \text{negl}\,. \tag{4.23}$$

where the probability is taken over the random coin-flips by $\mathcal{U}$*.*

Definition 4.5 (Soundness)**.** *Soundness guarantees that the prover can give a proof that verifies for a false statement only with negligible probability. More formally, given a statement* $x = (c^*, c^\dagger, \mathbb{C}_S)$ *, for any PPT adversaries* $\mathcal{A}$ *and honest verifier* $\mathcal{V}$*, and for every security parameter* $\lambda > 0$*, the following holds:*

$$\Pr\left[\mathcal{V}(x, \pi') = 1 \mid \pi' \leftarrow \mathcal{A}(x)\right] < \text{negl}\,. \tag{4.24}$$

where the probability is taken over the random coin-flips by $\mathcal{A}$*.*

Definition 4.6 (Anonymity)**.** *Anonymity guarantees that the outcome of the execution of the protocol does not non-negligibly increase the probability of identifying the user's ID* u *from the coin* c*. More formally, given a statement* $x = (c^*, c^\dagger, \mathbb{C}_S)$ *and a witness* $w = (u, r^*, r_0, r^\dagger, r_1, \ldots, r_n, \rho, \sigma)$*, for all PPT adversaries* $\mathcal{A}$*, and for all coin* $c \in \mathbb{C}$ *and for every security parameter* $\lambda > 0$*, it holds:*

$$\left|\Pr[u \leftarrow \mathcal{A}(c, x, \pi) \mid \pi \leftarrow \mathcal{U}(x, w)] - \Pr[u \leftarrow \mathcal{A}(c, x)]\right| \leq negl(\lambda) \tag{4.25}$$

where the probability is taken over the random coin-flips by $\mathcal{U}$ *and* $\mathcal{A}$*.*

Definition 4.7 (Fair Traceability)**.** *Fair Traceability guarantees Authority to trace the user's id* $u \in \mathbb{U}$ *involoved in a coin* $c \in \mathbb{C}$ *with an accepting fairness proof* $(x, \pi) \in \mathbb{X} \times \not\succ$

with pre-agreed fair probability $p(c.v)$ determined by the coin value $c.v$. More formally, let $\mathbb{K}$ be the set of all possible secret keys. Let $p : \mathbb{R}_{\geq 0} \to [0, 1]$ be a probability function satisfing ϵ-Fairness in Definition 4.3. There exists an efficient probabilistic extractor $\mathcal{E} : \mathbb{K} \times \mathbb{C} \times \mathbb{X} \times \not> \to \mathbb{U}$, which takes a secret key $sk \in \mathbb{K}$, a valid coin $c \in \mathbb{C}$, and an accepting statement-proof pair $(x, w) \in \mathscr{R}$ as inputs and outputs a user id $u \in \mathbb{U}$. Given a statement $x = (c^*, c^\dagger, \mathbb{C}_S)$ and a witness $w = (u, r^*, r_0, r^\dagger, r_1, \ldots, r_n, \rho, \sigma)$, for every honest user $\mathcal{U}$ and for the extractor $\mathcal{E}$, it holds:

$$|\Pr[\mathcal{E}(sk, c, x, \pi) = u \mid \pi \leftarrow \mathcal{U}(w, x)] - p(c.v)| < \epsilon \tag{4.26}$$

Finally, we claim that the construction of Fair-Anonymity in Section 4.3 satisfies these security notions.

Theorem 4.2 (Fair-Anonymity)**.** *The construction described in Section 4.3 satisfies Completeness (def. 4.4), Soundness (def. 4.5), Anonymity (def. 4.6) and Fair Traceability (def. 4.7).*

The proof of this theorem is rather straightforward from the above discussions. Here, I will give only the proof sketch.

Proof. I give the proof scketch that the construction of Fair-Anonymity satisfies the four security notions of Completeness, Soundness, Anonymity, and Fair Traceability as follows. Completeness follows directly from that of the Membership Proof (One-out-of-many Proof / OR-Proof) in the PoK-1 - PoK-4 , which implies $\mathcal{V}$ accepts if the all of the four proofs are accepting. In the same way, Soundness is implied from the soundness of PoK-1 to PoK-4. Adversary who breaks the Soundness (def. 4.5) has to break at least one of the soundness PoK-1 to PoK-4. Thus, this probability is negligible in λ Anonymity means that the probability of user information leaking from the published transcript (c, x, π) is negligible. From the Zero-knowledge property of PoK-1 to PoK-4, ensuring that the probability of distinguishing the witnesses is negligible. Noting that u is contained in the witness information, Anonymity is directly implied. Lastly, regarding Fair Traceability, the probability that u is transmitted via $\mathrm{C^2OT}_n^k$ approximates the function $p(x)$ (4.9), which satisfies the fairness property based on the (n, l) pair in equations (4.14) - (4.15) in the Execution protocol. Indeed, there is

an error due to approximating $p(x)$, defined over the real numbers $\mathbb{R}$, with integer pairs $n, l \in \mathbb{N}$, but this value is upper-bounded by $\epsilon \leq \frac{1}{2n}$ which decreases as n increases. $\square$

Chapter 5

Conclusion

The field of digital currencies is continuously evolving, balancing critical demands for user anonymity and the imperative to prevent illicit activities. This thesis contributes to this domain by presenting two novel cryptographic frameworks—Open-cash and Fair-Anonymity—that provide innovative solutions addressing both theoretical and practical aspects of privacy, transparency, and regulatory compliance.

My first major contribution, *Open-cash*, introduces the innovative concept of open-source observers embedded within secure electronic cash systems, potentially applicable to Central Bank Digital Currencies (CBDCs). The proposed observers leverage anonymous attested execution environments and advanced cryptographic techniques, such as blind signatures and zero-knowledge proofs based on BBS+ signatures, achieving a robust balance between strong privacy guarantees and necessary regulatory transparency. The theoretical innovations, particularly the novel primitive of Blind Indirect Attestation, represent a significant step forward in cryptographic security, enabling verifiable oversight without compromising user anonymity or trust. This approach effectively addresses societal concerns about centralized surveillance while maintaining necessary regulatory oversight.

The second contribution, *Fair-Anonymity*, provides a complementary solution tailored specifically for decentralized public blockchain environments like Bitcoin. This study introduces a fundamentally new fairness notion, which probabilistically determines transaction traceability solely based on transaction amounts, thereby eliminating vulnerabilities arising from transaction splitting. The Fair-Anonymity protocol

innovatively integrates a committed k-out-of-n Oblivious Transfer with modern zero-knowledge proof systems, resulting in a cryptographically rigorous and practically applicable method that elegantly balances legitimate user privacy with essential regulatory transparency. This ensures strong anonymity for regular, low-value usage while enabling effective scrutiny of potentially illicit high-value transactions, fostering a safer and more transparent cryptocurrency ecosystem.

Together, these contributions not only advance theoretical understanding within cryptographic research but also provide tangible frameworks with direct implications for real-world digital currency implementation. By resolving critical tensions between privacy and regulation, the proposed solutions offer pathways to safer, more transparent, and trustworthy digital currencies, significantly contributing to broader societal goals of financial inclusion, user autonomy, and regulatory compliance.

Future Work

While this thesis lays significant theoretical foundations, several avenues for future research remain open. Practical implementation and comprehensive benchmarking of these cryptographic frameworks in realistic environments are essential next steps. Furthermore, extending the proposed notions and protocols to integrate with emerging blockchain-based financial instruments and exploring their interoperability with existing regulatory frameworks could offer substantial additional societal value.

Bibliography

[ASM06] Man Ho Au, Willy Susilo, and Yi Mu. "Constant-Size Dynamic k-TAA". en. In: *Security and Cryptography for Networks*. Springer, 2006, pp. 111–125. ISBN: 978-3-540-38081-8. DOI: 10.1007/11832072_8.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. "Hierarchical Identity Based Encryption with Constant Size Ciphertext". In: *Advances in Cryptology – EUROCRYPT 2005*. Vol. 3494. Lecture Notes in Computer Science. 2005, pp. 440–456.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. "Short Group Signatures". en. In: *Advances in Cryptology – CRYPTO 2004*. Vol. 3152. Series Title: Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, pp. 41–55. DOI: 10.1007/978-3-540-28628-8_3.

[BGK93] Ernie Brickell, Peter Gemmell, and David Kravitz. "Trustee-based Tracing Extensions to Anonymous Cash and the Making of Anonymous Change". en. In: (1993).

[BL10] Ernie Brickell and Jiangtao Li. "Enhanced Privacy ID from Bilinear Pairing for Hardware Authentication and Attestation". en. In: *2010 IEEE Second International Conference on Social Computing*. Minneapolis, MN, USA: IEEE, Aug. 2010, pp. 768–775. ISBN: 978-1-4244-8439-3. DOI: 10.1109/SocialCom.2010.118. URL: http://ieeexplore.ieee.org/document/5591478/ (visited on 06/16/2024).

[BL12] Foteini Baldimtsi and Anna Lysyanskaya. *On the Security of One-Witness Blind Signature Schemes*. 2012.

[BL13] Foteini Baldimtsi and Anna Lysyanskaya. "On the Security of One-Witness Blind Signature Schemes". In: *Advances in Cryptology - ASIACRYPT 2013*. Vol. 8270. Lncs. Springer, 2013, pp. 82–99.

[Bla+11] Olivier Blazy, Sébastien Canard, Georg Fuchsbauer, Aline Gouget, Hervé Sibert, and Jacques Traoré. "Achieving Optimal Anonymity in Transferable E-Cash with a Judge". en. In: *Progress in Cryptology – AFRICACRYPT 2011*. Vol. 6737. Lecture Notes in Computer Science. 2011, pp. 206–223. (Visited on 04/25/2024).

[Bra93] Stefan Brands. "An Efficient Offline Electronic Cash System Based On The Representation Problem". In: *CWI*. Technical Report CS-R9323 (1993), p. 77.

[Bra94] Stefan Brands. "Untraceable Off-line Cash in Wallet with Observers". In: *Advances in Cryptology — CRYPTO' 93*. Vol. 773. Springer, 1994, pp. 302–318.

[CC04] Ernie Brickell Camenisch Jan and Liqun Chen. "Direct Anonymous Attestation". In: Oct. 2004.

[CDL16] Jan Camenisch, Manu Drijvers, and Anja Lehmann. "Anonymous Attestation Using the Strong Diffie Hellman Assumption Revisited". en. In: *Trust and Trustworthy Computing*. Springer International Publishing, 2016, pp. 1–20.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. "Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols". en. In: *Advances in Cryptology — CRYPTO ' 94*. Lecture Notes in Computer Science. 1994, pp. 174–187.

[CFN90] David Chaum, Amos Fiat, and Moni Naor. "Untraceable Electronic Cash". In: *Advances in Cryptology — CRYPTO' 88*. Ed. by Shafi Goldwasser. Lecture Notes in Computer Science. event-place: New York, NY. Springer, 1990, pp. 319–327. ISBN: 978-0-387-34799-8.

[Cha+90] David Chaum, Bert den Boer, Eugène van Heyst, Stig Mjølsnes, and Adri Steenbeek. "Efficient Offline Electronic Checks". In: *Advances in Cryptology — EUROCRYPT' 89*. Springer, 1990, pp. 294–301.

[Cha83] David Chaum. "Blind Signatures for Untraceable Payments". In: *Advances in Cryptology*. Springer US, 1983, pp. 199–203.

[Cha85] David Chaum. "Security without Identification: Transaction Systems to Make Big Brother Obsolete". In: *Communications of the ACM* 28.10 (Oct. 1985), pp. 1030–1044.

[CLM07] Jan Camenisch, Anna Lysyanskaya, and Mira Meyerovich. "Endorsed E-Cash". In: ISSN: 2375-1207. May 2007, pp. 101–115. (Visited on 04/25/2024).

[CP93] David Chaum and Torben Pryds Pedersen. "Wallet Databases with Observers". en. In: *Advances in Cryptology — CRYPTO' 92*. Vol. 740. Springer, 1993, pp. 89–105. ISBN: 978-3-540-57340-1.

[CS97] Jan Camenisch and Markus Stadler. "Efficient group signature schemes for large groups". en. In: *Advances in Cryptology — CRYPTO '97*. Springer, 1997, pp. 410–424.

[CT05] Cheng-Kang Chu and Wen-Guey Tzeng. "Efficient k-Out-of-n Oblivious Transfer Schemes with Adaptive and Non-adaptive Queries". In: *Public Key Cryptography - PKC 2005*. Lecture Notes in Computer Science. 2005.

[CZ05] Zhide Chen and Hong Zhu. "General Public Key m-Out-of-n Oblivious Transfer". en. In: *Computational and Information Science*. Lecture Notes in Computer Science. 2005, pp. 888–894.

[DPP07] Cécile Delerablée, Pascal Paillier, and David Pointcheval. "Fully Collusion Secure Dynamic Broadcast Encryption with Constant-Size Ciphertexts or Decryption Keys". In: *Pairing-Based Cryptography – Pairing 2007*. Springer Berlin Heidelberg, 2007, pp. 39–59.

[Fer94a] Niels Ferguson. "Extensions of Single-term Coins". In: *Advances in Cryptology — CRYPTO' 93*. Lecture Notes in Computer Science. Springer, 1994, pp. 292–301.

[Fer94b] Niels Ferguson. “Single Term Off-Line Coins”. In: *Advances in Cryptology — EUROCRYPT’ 93*. Lecture Notes in Computer Science. Springer, 1994, pp. 318–328.

[FHJ20] Marc Fischlin, Patrick Harasser, and Christian Janson. “Signatures from Sequential-OR Proofs”. In: *Advances in Cryptology – EUROCRYPT 2020*. Lecture Notes in Computer Science. Springer International Publishing, 2020.

[GK15] Jens Groth and Markulf Kohlweiss. “One-Out-of-Many Proofs: Or How to Leak a Secret and Spend a Coin”. In: *Advances in Cryptology - EUROCRYPT 2015*. Ed. by Elisabeth Oswald and Marc Fischlin. Lecture Notes in Computer Science. Springer, 2015, pp. 253–280. ISBN: 978-3-662-46803-6.

[GMS14] Fuchun Guo, Yi Mu, and Willy Susilo. “Subset Membership Encryption and Its Applications to Oblivious Transfer”. In: *IEEE Transactions on Information Forensics and Security* 9.7 (July 2014). Conference Name: IEEE Transactions on Information Forensics and Security, pp. 1098–1107. (Visited on 01/08/2024).

[Guo+13] Fuchun Guo, Yi Mu, Willy Susilo, and Vijay Varadharajan. “Membership Encryption and Its Applications”. en. In: *Information Security and Privacy*. Lecture Notes in Computer Science. Springer, 2013, pp. 219–234.

[HO24] Taishi Higuchi and Akira Otsuka. “Electronic Cash with Open-Source Observers”. In: *Financial Cryptography and Data Security. FC 2023 International Workshops*. Springer Nature Switzerland, 2024, pp. 286–302.

[HO25a] Higuchi, Taishi and Otsuka, Akira. “Fair-Anonymity: A Novel Fairness Notion for Cryptocurrency”. In: Computer Science & Information Technology, 2025.

[HO25b] Higuchi, Taishi and Otsuka, Akira. “Open-cash: an electronic cash scheme with open-source observers based on BBS signatures”. In: *IEEE Access* (2025).

[Kog] Dima Kogan. “Lecture 5: Proofs of Knowledge, Schnorr’s protocol, NIZK”. en. In: ().

[Lai+18] Jianchang Lai, Yi Mu, Fuchun Guo, Rongmao Chen, and Sha Ma. "Efficient k-out-of-n oblivious transfer scheme with the ideal communication cost". In: (2018).

[NP05] Moni Naor and Benny Pinkas. "Computationally Secure Oblivious Transfer". en. In: *Journal of Cryptology* 18.1 (Jan. 2005), pp. 1–35.

[NP99] Moni Naor and Benny Pinkas. "Oblivious transfer and polynomial evaluation". In: ACM, May 1999, pp. 245–254. ISBN: 978-1-58113-067-6.

[OO92] Tatsuaki Okamoto and Kazuo Ohta. "Universal Electronic Cash". In: *Advances in Cryptology — CRYPTO' 91*. Lecture Notes in Computer Science. Springer, 1992, pp. 324–337.

[PS96] David Pointcheval and Jacques Stern. "Provably Secure Blind Signature Schemes". In: *Advances in Cryptology — ASIACRYPT '96*. Vol. 1163. Springer, 1996, pp. 252–265.

[PST17] Rafael Pass, Elaine Shi, and Florian Tramèr. "Formal Abstractions for Attested Execution Secure Processors". en. In: *Advances in Cryptology – EUROCRYPT 2017*. Springer International Publishing, 2017.

[Rab81] Michael O. Rabin. *How To Exchange Secrets with Oblivious Transfer*. Harvard University Technical Report 81. 1981.

[Sch90] C. P. Schnorr. "Efficient Identification and Signatures for Smart Cards". In: *Advances in Cryptology — CRYPTO' 89 Proceedings*. Lecture Notes in Computer Science. Springer, 1990, pp. 239–252.

[Sed+21] Johannes Sedlmeir, Reilly Smethurst, Alexander Rieger, and Gilbert Fridgen. "Digital Identities and Verifiable Credentials". en. In: *Business & Information Systems Engineering* 63.5 (Oct. 2021), pp. 603–613. ISSN: 1867-0202. URL: `https://doi.org/10.1007/s12599-021-00722-y` (visited on 12/22/2024).

[THO22] Taisei Takahashi, Taishi Higuchi, and Akira Otsuka. "VeloCash: Anonymous Decentralized Probabilistic Micropayments With Transferability". In: *IEEE Access* 10 (Jan. 2022), pp. 93701–93730. (Visited on 01/14/2024).

[Tom+22] Alin Tomescu, Adithya Bhat, Benny Applebaum, Ittai Abraham, Guy Gueta, Benny Pinkas, and Avishay Yanai. *UTT: Decentralized Ecash with Accountable Privacy*. Publication info: Preprint. MINOR revision. 2022. URL: `https://eprint.iacr.org/2022/452` (visited on 09/10/2024).

[ZW05] Jianhong Zhang and Yumin Wang. "Two provably secure k-out-of-n oblivious transfer schemes". In: *Applied Mathematics and Computation* (Oct. 2005), pp. 1211–1220.