博 士 論 文

Probabilistic Micropayments in Decentralized Blockchain

Taisei TAKAHASHI

高橋 大成

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

2022 年 9 月

Abstract

Starting with Bitcoin, proposed in 2008, many decentralized blockchain schemes have been proposed and implemented. Blockchain has strong potential not only as a cryptocurrency, but also with smart contracts, an automated contract enforcement system. However, blockchain has two structural challenges. One is fast payments, and the second is throughput. In this thesis, we assume tamper-proof hardware and solve the three challenges.

Secure Offline Payments in Bitcoin We have achieved fast payments on the blockchain. Double-spending attacks on fast payments are one of the fatal architectural problems in Cryptocurrencies. The prior study proposed an offline fast payment scheme that relies on tamper-proof wallets produced by trustworthy manufacturers. With the wallets, the payee can immediately trust the transactions generated by the wallets without waiting for their registration to the blockchain. They proposed a protocol that makes use of a fragment of the main blockchain to prove to the wallets the legitimacy of preloaded coins. One drawback is that their protocol requires a trusted online time-stamp server to prove that the fragment is from honest miners. Otherwise, it sacrifices usability. To eliminate such an online trustee, we took the opposite approach that the payee (not the wallets) verifies the legitimacy of preloaded coins during offline payment. Consequently, our result shows that, with lightweight tamper-proof wallets, completely decentralized offline payment is possible without any modification to the existing Bitcoin network.

Decentralized Probabilistic Micropayments with Transferability Micropayments are one of the challenges in cryptocurrencies. The problems in realizing micropayments in the blockchain are the low throughput and the high blockchain transaction fee. As a solution, decentralized probabilistic micropayment has been proposed. The winning amount is registered in the blockchain, and the tickets are issued to be won with probability, which allows us to aggregate approximately $1/p$ transactions into one. Unfortunately, existing solutions do not allow for ticket transferability, and the smaller p, the more difficult it is to use them in the real world. We propose a novel decentralized probabilistic micropayment Transferable Scheme. It allows tickets to be transferable among users. By allowing tickets to be transferable, we can make them smaller. We also propose a novel Proportional Fee Scheme. This is a scheme where each time a ticket is transferred, a portion of the blockchain transaction fee will be charged. With the proportional fee scheme, users will have the advantage of sending money with a smaller fee than they would generally send through the blockchain. For example, sending one dollar requires only ten cents.

Anonymous Decentralized Probabilistic Micropayments with Transferability We propose VeloCash, Decentralized Probabilistic Micropayments with Transferability, which preserves anonymity. As a tamper-proof hardware assumption, VeloCash uses Attested Execution Secure Processors (AESP), a formal abstraction of secure processors with attested execution functionality and Direct Anonymous Attestation (DAA) to achieve anonymity for sending and receiving tickets. Even under anonymity, VeloCash can detect double-spending attacks perfectly and revoke the adversary's device.

Publication list

This thesis is based on the below publications.

Peer Reviewed Papers

[**THO22**] Taisei Takahashi, Taishi Higuchi and Akira Otsuka. "VeloCash: Anonymous Decentralized Probabilistic Micropayments with Transferability" in *IEEE Access*, 2022.

Peer Reviewed Proceedings of International Conferences

[**TO21**] Taisei Takahashi and Akira Otsuka. "Probabilistic Micropayments with Transferability". In *European Symposium on Research in Computer Security - ESORICS 2021*, Lecture Notes in Computer Science, pages 390–406, Cham, 2021. Springer International Publishing.

[**TO20**] Taisei Takahashi and Akira Otsuka. "Short Paper: Secure Offline Payments in Bitcoin". In *Workshop on Trusted Smart Contracts In Association with Financial Cryptography and Data Security*, Lecture Notes in Computer Science, pages 12–20, Cham, 2020. Springer International Publishing.

Acknowledgements

Firstly, I appreciate my supervisor, Professor Akira Otsuka. Fortunately, when I started studying blockchain five years ago, he was willing to accept that he would supervise me. He led me to the frontier in the blockchain. He has mentored me for these five years. Without his excellent mentorship, I would not have gone to a Ph.D. program, and I would not have been able to complete this work. I sincerely appreciate him for everything he has done for me. I am incredibly grateful to Professor Seiko Arita, Professor Kouichi Sakurai, and Associate Professor Kazumasa Omote for their constructive suggestions and valuable advice. I was able to take the first step into cryptography from Prof. Arita's lecture on cryptography. I am thankful to the members of Otsuka Laboratory. They have given me many cooperations, notices, and support throughout my laboratory life. In particular, they taught us various aspects of blockchain and cryptography. I want to show my appreciation to my managers and colleagues for their understanding and cooperation. Mainly, I sincerely appreciate Mr. Akira Nagata and Ms. Yumiko Baba in Serverworks Co., Ltd. They gave me the precious opportunity of researching blockchain at the Institute of Information Security. I could not fulfill this work without their excellent cooperation. Furthermore, I kindly thank my family. They always encourage and believe me. I sincerely appreciate them for their affection while studying and staying at home. Finally, I appreciate anonymous reviewers for their valuable comments.

List of Figures

Contents

Chapter 1

Introduction

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

1.1 Blockchain

Blockchain technologies, the basis of cryptocurrencies, was born in 2008 with Bitcoin, proposed by an anonymous person or group, Satoshi Nakamoto [Nak09]. Cryptocurrencies and blockchain have attracted significant attention because of their *Decentralized* properties.

The traditional financial system is a *Centralized* system based on the trust of specific institutions, such as governments, banks, and regulators. Central banks issue paper money, and electronic payment systems, such as credit cards, are also under the authorities' control. In contrast, blockchain has enabled payments that do not require the existence of a specific institution. Under the traditional financial system, a great deal of operational cost is required to maintain the system; thus, the blockchain is expected to simplify and reduce the cost.

Furthermore, blockchain can be used not only as a cryptocurrency but also as a *Smart Contract* that takes advantage of its property of "sharing the same state globally", which was realized by Ethereum [Woo] in 2013. Ethereum allows the execution of smart contract codes written in Turing-complete programming language that runs on the blockchain.

However, challenges exist for blockchain to have the same performance as the current

financial system. In this thesis, we address the biggest challenges that blockchain has: 1. Fast Payments, 2. High Throughput, 3. Anonymity with Transparency.

1.2 Contributions

1.2.1 Secure Offline Payments in Bitcoin

Karame et al. [KAC12] have analyzed double-spending attacks of fast payments in Bitcoin. This attack scheme allows an adversary to use more coins than he has for fast payment due to the property of Bitcoin, which takes a certain amount of time to be confirmed. Generally, 6-blocks, or roughly one hour before the transaction is registered and confirmed in the Bitcoin network, are considered secure for the attack. Fast payment, i.e., a scheme that allows the return of goods at the same time of payment, is needed in Bitcoin.

Dmitrienko et al. [DNY17] have proposed a secure fast payment scheme in Bitcoin, assuming tamper-proof hardware. However, their proposed method requires either a trusted online timestamp server or a limit on the duration of time coins can be used.

We propose a secure fast payment scheme in Bitcoin, following Dmitrienko et al. Unlike Dmitrienko et al., our scheme does not require an online timestamping server or a time limit on coins.

1.2.2 Decentralized Probabilistic Micropayments with Transferability

Due to the low throughput of the blockchain, micro payments, e.g., less than $1, are challenging to achieve.

Existing studies have proposed *Decentralized Probabilistic Micropayments*, where winning money is registered on the blockchain, lottery tickets are issued, and the tickets are used as currency. If the winning amount of money β is registered on the blockchain and the

probability of winning the ticket is p, the ticket is used as a currency with an expected value of $\beta \cdot p$. Unfortunately, existing research shows that tickets can be sent only once from the ticket issuer. Also, the smaller the winning probability p, the higher the throughput of the blockchain, but the less practical it becomes.

We propose that *Decentralized Probabilistic Micropayments with Transferability* that enables the transferability of ticket sending among payees as well as the issuer. Our proposed method increases the throughput of the blockchain depending on the number of spending; however, unlike existing research, it does not reduce the utility. In addition, a $1 transfer is feasible with a 10 cent fee, making it possible to settle payments with a smaller fee than on the blockchain.

1.2.3 Anonymous Decentralized Probabilistic Micropayments with Transferability

Under our proposed transferable decentralized probabilistic micropayments, payments are not anonymous. From the user's perspective, it is desirable that the spending be anonymous since the fact that the payment is not anonymous means that an individual's income is known.

We develop the transferable scheme and propose Decentralized Probabilistic Micropayments with Transferability, which preserves *Anonymity*, named VeloCash. Furthermore, even under anonymity, we ensure "transparency" that perfectly detects double-spending attacks and revokes the adversary.

1.3 Organization

In Chapter 2, we describe preliminaries. In Chapter 3, we describe the first work, "Secure Offline Payments in Bitcoin". It realizes secure fast payment on Bitcoin. In Chapter 4, we state the second work, "Decentralized Probabilistic Micropayments with Transferabil-

ity". It realizes transferable decentralized probabilistic micropayments that increase the blockchain's throughput and enable micropayments. In Chapter 5, we mention the third work, "VeloCash: Anonymous Decentralized Probabilistic Micropayments with Transferability". It realized transferable decentralized probabilistic micropayment, which preserves anonymity with transparency. In Chapter 6, we conclude this thesis.

Chapter 2

Preliminaries

This chapter describes the security notions of blockchain, Direct Anonymous Attestation (DAA), and Attested Execution Secure Processors (AESP). Primarily, DAA and AESP are described in Chapter 5.

2.1 Security Notions of Blockchain

Garay et al. [GKL15] have analyzed Bitcoin's backbone protocol. They formalize and prove three fundamental properties: *common prefix property*, *chain quality property*, and *chain growth property*.

Informally, *common prefix property* means that the probability that k blocks from the end of an honest user's local chain are not prefixed to another user's chain decreases exponentially with the security parameter. See for the precise statement Theorem 15 of [GKL15].

2.2 Probabilistic Micropayments

The idea of Probabilistic Micropayments has been proposed by Wheeler [Whe97] and Rivest [Riv97]. Since small payments would be costly if settled each time, they proposed a lottery-style protocol where the ticket issuer deposits a large amount of money in the bank, and the winner

could receive the money if they won. The lottery tickets can be used as currency, and the value per ticket is regarded as the ticket's expected value. In this scheme, the existence of a bank is mandatory, and participants are limited to people who have a relationship with the bank.

MICROPAY [Ps15] and DAM [CGL+17] have been proposed as Decentralized Probabilistic Micropayments using blockchain. Since both have a large overhead of supporting sequential micropayments, Almashaqbeh et al. have proposed MicroCash [ABC20] which is a light-weight protocol for non-interactive and sequential payments.

2.3 Direct Anonymous Attestation (DAA)

Direct Anonymous Attestation (DAA) remote authentication scheme for trusted hardware modules has been proposed by Brickell et al. [BCC04]. DAA has been adopted by the Trusted Computing Group (TCG).

When a verifier communicates with a user in possession of a trusted hardware module, DAA allows the verifier to verify that the output from the user is from the hardware module. In addition, since DAA preserves anonymity, the verifier can only know that the output is from the trusted hardware module and can not identify the hardware.

DAA can be seen as group signatures [BMW03] but differs from group signatures in that DAA does not have an *opening* algorithm that allows the group manager to obtain the identity of the signer from the signature. Instead of having *opening* function, DAA has a so-called "revocation function". Suppose a particular hardware module has been broken and its secret key has been compromised; the secret key is placed on the revocation list. When a verifier receives the signature, he can verify whether it is signed with the secret key on the revocation list. The verification can be done by the value $K = B^f$, where f is the secret key, K and B are the values in the signature, where B is the generator of an algebraic group and computing the discrete algorithms are hard.

There are multiple DAA schemes have been proposed [SRC15, GHS11, BCL09, CPS10, Che10]. In this paper, we adopt *Enhanced Privacy ID (EPID)* scheme proposed by Brickell and Li [BL10, BL12]. EPID is a scheme proposed by Intel Corporation and is already in use in the real world, embedded in chipsets such as Intel SGX. EPID is compliant with International Standards Organization standard ISO/IEC 20008, 20009 and approved by the Trusted Computing Group (TCG) as the recommended scheme. Intel has made EPID an open-source to processor manufacturer under the Apache 2 license. In 2015, Microchip and Atmel announced that they had licensed the EPID technology [Cora, Corb].

2.4 Verifiable Delay Functions (VDF)

A Verifiable Delay Function (VDF) [BBBF18] is the function that requires sequential steps to evaluate yet outputs a unique value that can be efficiently and publicly verifiable.

VDF could be used to build a randomness beacon [Rab83], an ideal function that regularly outputs unpredictable random values, using a permissionless blockchain like Bitcoin and Ethereum.

Chapter 3

Secure Offline Payments in Bitcoin

3.1 Introduction

Double-spending attacks on fast payments [KAC12] are one of the fatal architectural problems in Cryptocurrencies. Double spending refers to the payment where the same coin is spent twice in a way that the receiving party cannot notice the invalidity of the payment. We study an offline immediate payment scheme based on blockchain secure against the double-spending attacks on fast payment assuming the security of tamper-proof wallets. Dmitrienko et al. [DNY17] pointed out that Bitcoin requires clients to be online to perform transactions and a certain amount of time to verify them, and also offline payments raise non-trivial challenges, as the payee has no means to verify transactions. Even online, fast payments are shown to be vulnerable to double-spending attacks [KAC12]. Offline immediate payments are long demanding in cryptocurrencies. In practice, without changing the ongoing systems, "fast payment" is widely used such that the payee could accept the transaction immediately by checking the signature and the payer's balance to confirm that the payer has enough money to spend. However, Karame et al. [KAC12] pointed out that, for such a fast payment scheme, the double-spending attack is possible if a malicious payer makes use of the race condition of the double-spending transactions that reach the payees with

different timing in the peer-to-peer network. Dmitrienko proposed a solution for immediate and offline payments that relies on tamper-proof wallets manufactured by a trustworthy manufacturer and deploys a time-based transaction confirmation mechanism. They use a fragment of the main blockchain to prove the legitimacy of pre-loaded coins to the wallets. Their scheme solved the double-spending problem, as Karame et al. suggested. The wallet loses its credibility if a malicious user succeeds to pre-load illegal coins by intention. Thus, it is enormously important to establish secure coin pre-loading to the wallet. In order to achieve this, Dmitrienko proposes an interesting protocol that proves the fact that a pre-loading payment to the wallet existed using a subchain, a fragment of the main blockchain. They considered that a subchain is a part of the mainchain if and only if the total PoW to mine the blocks in the subchain is greater than some predetermined lower bound and the average time to produce the subchain is less than some constant time period. Thus, to verify the proof inside the tamper-proof wallet, objectively measuring the total computation time consumed to generate the subchain is necessary. (A) A trivial solution to this is to assume a trusted time-stamp server, which supplies a time-stamp to a block every time the corresponding block is created. (B) Another solution proposed in [DNY17] is to set the expiration time of the pre-loading coins. This also convinces the wallet that the subchain is produced within some bounded time period. Apparently, the construction (A) with the trusted time-stamp server requires an additional trusted third party. The construction (B) is decentralized, but it sacrifices usability to a great extent.

3.1.1 Related work

Other studies of interest include Teechan, an offline payment channel proposed by Lind et al. [LNE+19]. It assumes tamper-proof wallets on both sides and achieves offline payments. However, it has to deposit some funds, which corresponds to our pre-loading, into the 2-of-2 multisig address between the payee and the payer sufficiently before the offline payment occurs. Thus, the setting differs from ours.

In the theoretical aspect, Garay et al. [GKL15] formulated the basic properties of the blockchain in Bitcoin [Nak09] such as "common prefix" and "chain quality," assuming that the hashing power of an adversary controlling a fraction of the parties is strictly less than 1/2. Further extensions are made for variable difficulty [GKL15], a security analysis in the "semi-synchronous" network model [PSS17].

We propose a novel offline payment scheme alternative to Dmitrienko's protocol. The advantage of our scheme is it is fully decentralized; that is, it does not assume any external trusted time-stamp server. Further, we do not need to set any expiration time for the pre-loaded coins.

3.2 Security Properties

- $\mathbb{C}$: a blockchain.
- B_i: i-th block in $\mathbb{C}$ is a triple $\langle \mathsf{H}(B_{i-1}), \boldsymbol{x}_i, nonce \rangle$.
- τ: a transaction of a form $\mathsf{Sign}(\mathsf{sk}_A; A \to B, value)$.
- $\boldsymbol{x}_i$: a root of Merkle Tree for a set of transactions $\{\tau_1, \tau_2, \ldots\}$ in B_i.
- $\mathcal{U}$: a set of users.
- $\mathcal{H}$: a set of honest users, $\mathcal{H} \subseteq \mathcal{U}$.
- $X, Y \in \mathcal{U}$: (typically, X as a payer, Y as a payee).

To analyze the security of our scheme, we follow the notions introduced by Garay et al. [GKL15]. All players are bounded interactive turing machines and are all synchronized and messages are exchanged in a discrete-time frame called round. $\{EXEC^{t,n}_{\Pi,\mathcal{A},\mathcal{Z}}(z)\}_{z\in\{0,1\}^*}$ denotes the random variable ensemble that determines the output of the environment $\mathcal{Z}$ on input z for a protocol Π with adversary $\mathcal{A}$. $VIEW^{t,n}_{\Pi,\mathcal{A},\mathcal{Z}}(z)$ denotes the concatenated view of all parties after the completion of an execution $EXEC^{t,n}_{\Pi,\mathcal{A},\mathcal{Z}}(z)$. A "flat" model is

assumed where all parties execute exactly q mining trials (hash queries) per round. Then, each mining trial by various players is modeled as a Bernoulli distribution with different parameters, and the deviation from the expected probability is estimated. We denoted by κ the length of hash function output, by η parameter determining block to round translation, and by f probability at least one honest party succeeds in finding a POW in a round. Garay et al. [GKL15] showed that any consecutive rounds S of length $|S| > \eta\kappa$ in the blockchain protocol is a "typical execution" with probability $1 - e^{-\Omega(\kappa)}$ where honest and adversarial mining trials succeed as expected within a bounded probability fluctuation of at most ϵ.

3.3 Secure Offline Payments in Bitcoin

Our immediate payment protocol construction basically follows the Dmitrienko's protocol [DNY17] but without the existence of the time-stamp server. The main difference in the protocol is that the correctness of the coin-preloading to the tamper-proof wallet is verified by the payee in our construction, not the payer as in Dmitrienko's protocol. Similarly to their scheme [DNY17], our construction also assumes the tamper-resistant wallet to incorporate overspending prevention. The tamper-proof wallet has a secret key sk_T which was created by the wallet manufacturer T. The wallet also has a variable *balance* (≥ 0). It increases in the coin preloading phase and decreases in the offline payment phase. Our construction has 3 phases: (i) online coin preloading, (ii) offline payment, and (iii) coin redemption and wallet revocation. The details are as follows.

Online Coin Preloading

The flow diagram is shown in Figure 3.1. In the coin preloading phase, the payer X requests a new account w from the wallet (Step 1), then create the escrow transaction τ_l transferring b_l bitcoins from her account x to w, then commit it to the networks (Step 2). As soon as τ_l is verified and integrated into the Bitcoin network in a block, say B_i, X takes B_i (Step

3), and provides τ_l and the witness of the membership proof $\widetilde{\tau}_l$[1] to W (Step 4). W sets its *balance* to b_l and stores τ_l, $\widetilde{\tau}_l$, and replies status (Step 5)[2]. For simplicity, we assume one-time coin preloading for every account w such that once an amount b_l is preloaded to w, the wallet W never accepts preloading transaction to w anymore and only makes payments while $balance \geq 0$. It is not hard to extend it to multiple coin preloading.

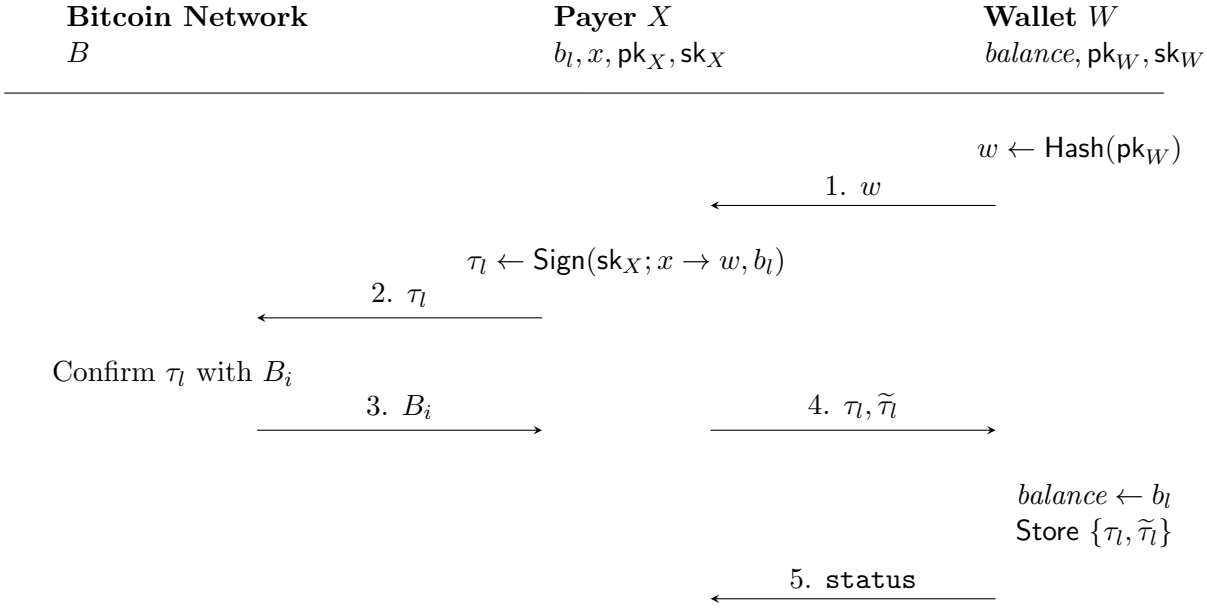


Figure 3.1: Online Pre-loading protocol

Offline Payment

The flow diagram is shown in Figure 3.2. In the offline payment phase, the payee Y sends the public key pk_Y and the requested amount b_o to the payer X, which are immediately forwarded to W (Step 1). W checks the balance and if $balance \geq b_o$, it decreases *balance* by b_o and generates a transaction $\tau_o = \mathsf{Sign}(sk_w; w \rightarrow y, b_o)$. Further, W generates a $\mathsf{proof} = \mathsf{Sign}(\mathsf{sk}_T; \tau_o, \tau_l, \widetilde{\tau}_l)$ that shows τ_o was created within the tamper-proof wallet by signing with sk_T. The resulting τ_o, proof and cert_T, a trustworthy vendor certificate, are sent to Y (Step 3). Y accepts the transaction if τ_l is $\mathsf{confirmed}$ and τ_o is valid and issued by

[1] $\widetilde{\tau}_l$ is a set of hash values such that $\mathsf{member}(B, \tau, \widetilde{\tau}) = 1$.

[2] The wallet does not check the validity of coin preloading transaction τ_l. Payments made from $\mathsf{unconfirmed}$ τ_l will be rejected by payees.

a tamper-proof wallet. More formally, Y accepts τ_o and *proof* if and only if

$$\begin{cases} \mathsf{cert}_T \text{ is trustworthy} \\ \mathsf{Verify}(\mathsf{pk}_T; \mathsf{proof}) = 1 \\ \tau_o \in (\mathcal{V}_{\mathbb{C}_Y} \cap \mathsf{Sign}(sk_W; w \to \cdot, b_o)) \\ \tau_l \in \left(\mathbb{C}_Y^{\lceil k} \cap \mathsf{Sign}(\cdot; \cdot \to w, b_l)\right) \end{cases} \tag{3.1}$$

If all checks succeed, Y stores τ_o, *proof* and replies to W with status (Step4).

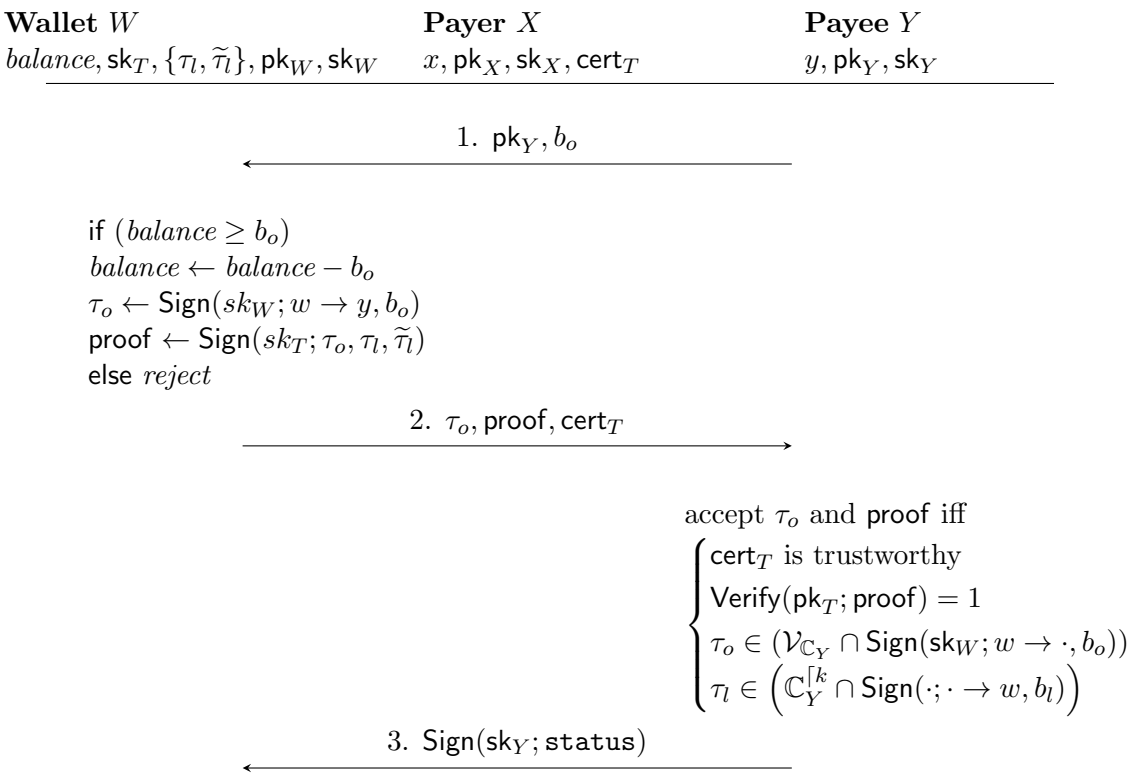


Figure 3.2: Offline payment protocol

Coin Redemption and Wallet Revocation

The flow diagram is shown in Figure 3.3. When Y gets online, Y proceeds to the coin redemption and wallet revocation phase. Y broadcasts τ_o to the Bitcoin network in order to redeem the coins received from W (Step 1). Next, the Bitcoin network verifies τ_o and

integrates it into the blockchain. The payee Y observes the Bitcoin network and periodically updates its local chain $\mathbb{C}'_Y$ reflecting the newly mined blocks (Step 2). Y waits until τ_o is $\mathsf{confirmed}$ or τ_o becomes $\mathsf{invalid}$, $\tau_o \notin \mathcal{V}_{\mathbb{C}'_Y}$. If τ_o is $\mathsf{invalid}$, Y initiates revocation by creating a revocation transaction $\tau_r = \mathsf{Sign}(\mathsf{sk}_Y; \mathsf{proof}, \mathsf{cancel}\ \tau_o)$ and send it to Insurer Z (Step 3). Z investigates τ_r and, to compensate Y for the damage of b_o issues τ_Z and then commits to the Bitcoin network (Step 4).

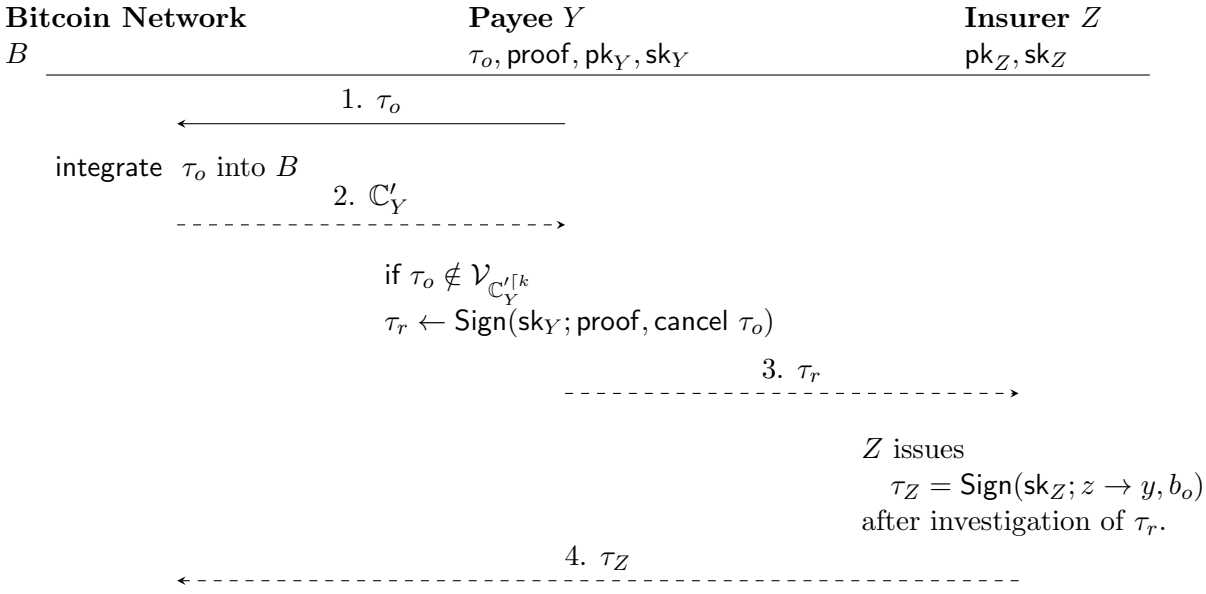


Figure 3.3: Coin redemption and double-spending wallet revocation protocol

3.4 Security Model

Definition 1. *τ is said to be* valid *with respect to a blockchain $\mathbb{C}$ if and only if τ satisfies*[3] *all the pre-agreed requirements*[4] *with respect to the blockchain $\mathbb{C}$. Further, we denote by $\mathcal{V}_{\mathbb{C}}$*

[3]Dmitrienko [DNY17] have introduced a slightly different term $\mathsf{CheckSyntaxT}$ for transaction validation. $\mathsf{CheckSyntaxT}$ refers to the syntactical conformance of transactions to the requirements in the blockchain. On the other hand, valid refers to all the requirements for integrating into the blockchain. Those are the syntactical conformance of transactions, the correctness of the payer's signature and further requirements such as "the payer's account must exist", and "balance after the transaction must not be negative."

[4]In Bitcoin, the requirements (e.g., Bitcoin Improvement Proposals) are subject to change. New proposals will be incorporated into the requirements after agreed upon among the majority of miners through the voting process in the blockchain.

a set of all possible valid *transactions with respect to* $\mathbb{C}$ *such that*

$$\mathcal{V}_{\mathbb{C}} = \{\tau \mid \tau \textit{ is } \mathsf{valid} \textit{ with respect to } \mathbb{C}\}.$$

valid transactions are correctly formed and not over-spent more than the balance of the payer's account as far as regarding the given blockchain $\mathbb{C}$ as the mainchain. Once valid transactions are broadcasted to the Bitcoin network by peers, they may be integrated into the mainchain if the given chain $\mathbb{C}$ is shared by honest miners, and thus the transaction is valid for those honest miners. Note that this is not guaranteed. Because the given chain $\mathbb{C}$ may contradict the localchains of other honest miners. Furthermore, the set of valid transactions will change as chains evolve.

Definition 2. *A transaction* τ *is said to be* confirmed[5] *if and only if for every honest player* $X \in \mathcal{H}$ *the transaction* τ *can be found on his localchain* $\mathbb{C}_X$*, that is,* $\tau \in \mathbb{C}_X$ *for all* $X \in \mathcal{H}$*.*

Once a transaction τ is confirmed, the transaction must be in the mainchain $\mathbb{C}$ where $\mathbb{C}$ is the longest prefix of all localchains held by honest players such that $\mathbb{C} \preceq \mathbb{C}_X$ for all $X \in \mathcal{H}$. where $\preceq$ is a prefix relation [GKL15]. We consider confirmed transactions are valid. That is, τ is confirmed $\implies \tau \in \mathcal{V}_{\mathbb{C}_X}$ for all $X \in \mathcal{H}$.

Definition 3. *Given a security parameter* k*,* τ_o *is said to be* verified *by a proof of a form*

[5] In practice, a transaction is *confirmed* after the block that contains the transaction has at least six blocks built on top. This is the situation where our notion of confirmed is satisfied with high probability.

$\mathsf{Sign}(\mathsf{sk}_T; \tau_o, \tau_l)$ *with respect to a localchain* $\mathbb{C}$ *if and only if*

$$\begin{cases} \mathit{cert}_T \text{ is issued by a trustworthy provider} \\ \mathsf{Verify}(\mathsf{pk}_T; \mathsf{proof}) = 1 \\ \tau_o \text{ is a transaction of a form } \mathsf{Sign}(\mathsf{sk}_W; w \to \cdot, b_o) \\ \tau_l \text{ is a transaction of a form } \mathsf{Sign}(\cdot; \cdot \to w, b_l) \\ \tau_o \in \mathcal{V}_{\mathbb{C}} \\ \tau_l \in \mathbb{C}^{\lceil k} \end{cases} \tag{3.2}$$

where cert_T *is a certificate issued to a public key* pk_T *which is generated within a tamper-proof wallet* W *at the production time.* $(\mathsf{pk}_W, \mathsf{sk}_W)$ *is a key pair generated by* W*, and* w *is an account related to* pk_W*.*

In the offline payment, a payee Y verifies the received transaction τ_o by a proof $\mathsf{Sign}(\mathsf{sk}_T; \tau_o, \tau_l)$ with his localchain $\mathbb{C}_Y$. If all of the above conditions are satisfied, Y is convinced that the coin-preloading transaction to the payer's wallet τ_l is $\mathsf{confirmed}$ by all honest players with overwhelming probability in k. As far as the tamper-proof wallet honestly produces the payment transaction τ_o, Y can believe that τ_o is not an overspending transaction and will be $\mathsf{confirmed}$ later on. To see whether τ_l satisfies the last condition $\tau_l \in \mathbb{C}^{\lceil k}$, in a naive construction, the payee must keep the whole set of transactions previously registered in the blockchain $\mathbb{C}$. For efficiency purposes, we assume that all transactions in a block B are kept in the form of a Merkle Tree, and B only keeps the root hash value of the tree. Let $\widetilde{\tau}_l$ be a witness of the membership proof or a set of all sibling hash values in every branch in the path from the root to the leaf τ_l. Using the witness $\widetilde{\tau}_l$, the membership proof can be

efficiently proved since there exists a function $\mathsf{member}(\cdot)$ such that

$$\mathsf{member}(B, \tau, \widetilde{\tau}) = \begin{cases} 1 & \text{if } \tau \in B \\ 0 & \text{otherwise} \end{cases}$$

We replace the proof with $\mathsf{Sign}(\mathsf{sk}_T; \tau_o, \tau_l, \widetilde{\tau_l})$ where offline payee needs efficiency.

Theorem 1. *We assume there exists a tamper-proof wallet W. Given a security parameter k and an offline transaction τ_o accepted by a payee Y such that τ_o is* $\mathsf{verified}$ *by a* $\mathsf{proof} = \mathsf{Sign}(\mathsf{sk}_w; \tau_o, \tau_l)$ *with respect to the payee's localchain $\mathbb{C}_Y$, the probability that the transaction τ_o, later on, changes its state to* $\mathsf{invalid}$ *is negligible in k. That is, with $\mathbb{C}_Y^{\lceil k} \preceq \mathbb{C}'_Y$, we have*

$$\Pr\left[\tau_o \notin \mathcal{V}_{\mathbb{C}'_Y}^{\lceil k} \mid \tau_o \text{ is } \mathsf{verified} \text{ by } \mathsf{proof} \text{ with respect to } \mathbb{C}_Y\right] < \mathsf{negl}(k). \tag{3.3}$$

Proof. Given τ_o is $\mathsf{verified}$ by proof with respect to $\mathbb{C}_Y$, Equation (3.2) holds. Under the tamper-proof assumption, the tamper-proof wallet W never overspend, exceeding the preloaded balance b_l. Therefore, if the preloading transaction τ_l is $\mathsf{confirmed}$, that is,

$$\tau_l \in \mathbb{C}_X \text{ for all } X \in \mathcal{H}, \tag{3.4}$$

then the offline transaction τ_o cannot become $\mathsf{invalid}$ with respect to $\mathbb{C}'_Y$. Since τ_l is already in a localchain of Y, that is, $\tau_l \in \mathbb{C}_Y^{\lceil k}$, for τ_l not to be $\mathsf{confirmed}$, there must exist an honest player Z with a localchain $\mathbb{C}_Z$ such that $\tau_l \notin \mathbb{C}_Z$. The common prefix property states that all localchain $\mathbb{C}$ held by honest players must satisfy

$$\mathbb{C}^{\lceil k} \preceq \mathbb{C}_Z \tag{3.5}$$

with negligible error probability in k. Substituting $\mathbb{C} \to \mathbb{C}_Y$, $\tau_l \in \mathbb{C}_Y^{\lceil k}$ contradicts with $\tau_l \notin \mathbb{C}_Z$. Thus, τ_l must be $\mathsf{confirmed}$ with arbitrarily high probability $1 - \mathsf{negl}(k)$ with the

security parameter k. Hence the theorem. □ □

In the case $\tau_o \notin \mathcal{V}_{\mathbb{C}_Y'^{\lceil k}}$ where the offline transaction τ_o is turned out to be $\mathsf{invalid}$ later on with respect to the payee's evolved localchain $C_Y'(\succeq \mathbb{C}_Y^{\lceil k})$, this must be the case where the tamper-proof wallet assumption is broken, and τ_o is found to be an overspent transaction. Even in this case, the preloading transaction τ_l must still be $\mathsf{confirmed}$ with $1 - \mathsf{negl}(k)$. Therefore, the $\mathsf{proof} = \mathsf{Sign}(\mathsf{sk}_w; \tau_o, \tau_l)$ still satisfies the following 5 of all 6 conditions in (3.2) for all honest player $X \in \mathcal{H}$ with $1 - \mathsf{negl}(k)$.

$$\begin{cases} \mathsf{cert}_T \text{ is issued by a trustworthy provider} \\ \mathsf{Verify}(\mathsf{pk}_T; \mathsf{proof}) = 1 \\ \tau_o \text{ is a transaction of a form } \mathsf{Sign}(\mathsf{sk}_W; w \to \cdot, b_o) \\ \tau_l \text{ is a transaction of a form } \mathsf{Sign}(\cdot; \cdot \to w, b_l) \\ \tau_l \in \mathbb{C}_X{}^{\lceil k} \end{cases} \tag{3.6}$$

This fact convinces all honest players. Given τ_o is $\mathsf{invalid}$ with respect to $\mathbb{C}_Y'$, that is, $\tau_o \notin \mathcal{V}_{\mathbb{C}_Y'^{\lceil k}}$, the redeeming transaction $\tau_r = \mathsf{Sign}(\mathsf{sk}_Y; \mathsf{proof}, \mathsf{cancel}\ \tau_o)$ becomes valid with respect to $\mathbb{C}_X \succeq \mathbb{C}_Y'^{\lceil k}$ for all $X \in \mathcal{H}$ with arbitrarily high probability $1 - \mathsf{negl}(k)$. The trustworthy provider of the tamper-proof wallet or an insurance company might compensate Y for the damage of b_o after τ_r is $\mathsf{confirmed}$.

3.5 Conclusion

In this chapter, we have shown that, with light-weight tamper-proof wallets, completely decentralized offline payment is possible without any modification to the existing Bitcoin network. Our protocol requires the coin pre-loading transaction to be confirmed, and its block is delivered to every possible payee before the first offline payment is made. This should be the best possible for the Bitcoin network.

Based on the work of Dmitrienko et al. [DNY17], we constructed a scheme on Bitcoin, that is, Proof of Work type blockchain, using the "common-prefix property" to obtain transactions agreement among all nodes on the blockchain. Since the "common prefix property" has also been proposed for Proof of Stake type blockchains [KRDO17, DPS19], our scheme can also be used in Proof of Stake type blockchains.

Chapter 4

Decentralized Probabilistic Micropayments with Transferability

4.1 Introduction

Micropayments are minimal payments, e.g., less than $1, and can be used in a wide range of applications, such as per-page billing in e-books and delivering content billed per minute. However, it is challenging to realize micropayments in the blockchain.

The problems in realizing micropayments in the blockchain are the low throughput and the high blockchain transaction fee. Since the capacity of each block is fixed, miners give priority to transactions that can generate high fees and put off micropayment transactions with low fees. In addition, the blockchain transaction fees do not depend on the amount of money to be transferred. Thus, the blockchain transaction fees can be relatively small for high-value transfers but high for micropayments.

The above problems can be solved by Layer-two [GMSR+20]. Instead of registering all transactions in the blockchain, Layer-two aggregates small transactions into a few larger ones, increasing transaction throughput and reducing transaction fees. Decentralized probabilistic micropayments [ABC20, TO21] have been proposed as one of the methods for Layer-two.

It is a lottery-based scheme, the amount of required payments is locked in an escrow, and micropayments are issued as lottery tickets. Let the winning amount be β, and the winning probability is p, the expected value per lottery ticket is $p \cdot \beta$, and the ticket is used as currency. Probabilistic micropayments allow us to aggregate the entire transactions by approximately p. For example, if $10,000$ transactions are to be processed by a probabilistic micropayments scheme, only $10,000 \cdot p$ transactions will be registered in the blockchain.

Almashaqbeh et al. have proposed MicroCash [ABC20] which is a lightweight protocol for non-interactive and sequential payments. The disadvantage of MicroCash is that the game theory guarantees safety against double-spending attacks. Thus, the penalty escrow, which is confiscated after the double-spending attack is discovered, is expensive. As an example, when $m = 5$ and $B_{\text{escrow}} = 2000$, the penalty escrow is $B_{\text{penalty}} = 477.6$. In addition, tickets can only be sent once by the ticket issuer; in other words, the tickets can not be *transferable*.

As MicroCash, when safety is constructed using only a game-theoretic approach, considering penalty escrow, the number of honest users who can receive the ticket, u, is realistically constrained to about 5. If we make u large, we need to make the penalty escrow large in proportion to u. As an alternative plan, if we assume that the users can not commit malicious activity, such as a tamper-proof assumption, u can be large without penalty escrow. However, the smaller p is, the higher the gambling potential becomes and the less the payee can use it for actual economic transactions. If many tickets with a minimal winning probability are sent and do not win, the honest users can not make any income. This is because if the ticket can not be transferable, the payee will not earn any income unless the ticket they received wins. The smaller p, the more the opportunity to get an income is lost.

If the ticket is *transferable*, p can be reduced. The payees do not lose anything since the ticket can be used to pay others even if the ticket is not won. However, it is challenging to achieve transferability with existing solutions. Since the ticket is transferable, the double-spending attacks can be performed by the issuer and all users. Requiring game-theoretically guaranteed penalty escrow for all users is practically undesirable because of high collateral

costs. Suppose the ticket transfer is limited to a tamper-proof device, malicious activities that deviate from the protocol can be prevented, and transferability can be achieved without high penalty escrow.

4.2 Contribution

We propose a novel decentralized probabilistic micropayments, *Transferable Scheme*, which allows tickets to be transferable among users.

The contribution of the chapter is twofold:

- Novel probabilistic micropayment techniques:

  As far as we know, all of the ever-proposed probabilistic schemes are based on lottery tickets where only a small fraction of payees will win the lottery and receive multi-fold awards. In contrast, our probabilistic micropayment [TO21] utilizes transferability, where every payer has to pay a transaction fee proportional to the paid amount, say 10%, and the fees are accumulated inside the transferred ticket. Then, only the winner of the lottery ticket will take all of the accumulated transaction fees as the lottery award. In addition, non-winners will always gain expected revenue with far less speculativity. The value of the ticket will diminish exponentially as transferred until the expected velocity. Most of the payments fall into the range of micropayments.

- Imperfect tamper-proof assumption with game-theoretic upper-bound for the adversary's utility:

  We assume tamper-proof wallets which prevent double-spending before it happens. Our double-spending detection techniques are shown to detect perfectly when the double-spent ticket is about to be registered in the blockchain (fork detection) and detect probabilistically when received at the payee (collision detection). With these detection techniques, we can eliminate the need for penalty escrow (required in the

previous works [ABC20]) and force adversaries to weigh the cost of breaking a tamper-proof wallet against the maximum expected value that the adversary can obtain from the attack.

4.3 Background

4.3.1 Payment Channels and Networks

The payment channel establishes a private, peer-to-peer transmission protocol. Based on predefined rules, two parties can agree to update their state and transfer money by exchanging authenticated state transitions in a so-called 'off-chain' fashion.

In order to conduct a transaction on *Payment Channel*, two parties must first register a shared 2-of-2 multi-sig escrow fund in the blockchain and establish the channel. The payment channel enables the two parties to perform transactions through private communications. After the sending and receiving are completed in the channel, the final fixed value is registered in the blockchain. Only two transactions are registered in the blockchain per channel, escrow fund transaction and final fixed value. A payer can send money to a user who has not established a channel with the payer through the *Payment Network* between users who have established a channel. For example, suppose Alice sends 0.1 coins to Charlie, who has not established a channel with Alice. First, Alice sends 0.1 coins to Bob, with whom Alice has a channel. Next, Bob sends 0.1 coins to Charlie, whom Bob has a channel.

Unfortunately, the payment channel and the network have the disadvantage of high collateral cost [MBB+19]. Each time a channel is established, escrow is required between two parties. Also, the longer the payment network path, the more reserves are required and locked. Since the reserves can not be used during the locktime periods, the reserves represent a lost opportunity. Furthermore, in a payment network, a fee is charged for each pass through the nodes. It is impractical to adopt a payment network for micropayments since it is undesirable to incur the cost for each node.

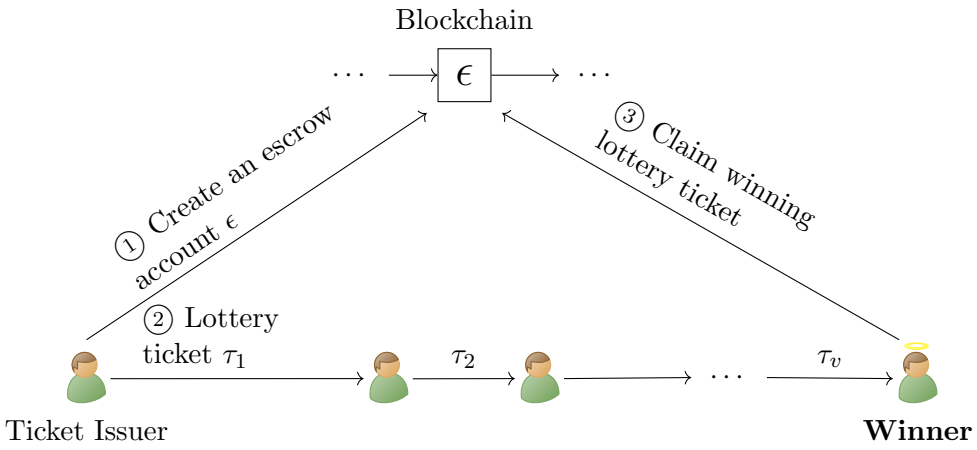


Figure 4.1: Overall Design

4.4 Ticket Transfer Protocol

This section presents the design of our transferable scheme. We start with an outline of the lottery ticket transaction, followed by a detailed description of each part.

4.4.1 Outline

The outline of the system is shown in Figure 4.1.

Step 1, The issuer issues a smart contract escrow account ϵ and registers and confirms that ϵ has been registered in the blockchain. **Step 2**, The issuer issues the ticket τ for probabilistic micropayments and sends it to a user. The payee verifies that the ticket came from a legitimate wallet and that the escrow account is properly registered in the blockchain. If there is no problem, the user receives the ticket and returns the service or product to the payer. Then, the payee signs the ticket with his wallet and sends it to another user. **Step 3**, If the ticket received meets the requirements for winning, the ticket is sent to the escrow account ϵ.

The sequence of procedures in this scheme, such as ticket issuance and payment with the ticket, is done using a tamper-proof wallet.

Tamper-proof wallet

The premise is that all users participating in the transferable scheme have tamper-proof hardware wallets.

The wallet consists of a tamper-proof device manufactured by a trusted manufacturer. It does not accept any unauthorized operation that deviates from the protocol, such as double-spent tickets.

There are two keys in the wallet. One is a key for personal use key pairs $(\mathsf{sk}_{W_X}, \mathsf{pk}_{W_X})$ for sending and receiving the ticket; we denote the hash value of pk_{W_X} be the "address" associated with the wallet owner. The other is a secret key sk_T used to prove that the ticket was created and sent from a legitimate wallet. Additionally, the wallet owner possesses a certificate cert_T corresponding to the secret key sk_T.

4.4.2 Escrow Setup

The flow diagram is shown in Figure 3.1.

The issuer X requests a new account w_X from the wallet (Step 1), then create the escrow transaction τ_l transferring β coins from the account x to the wallet address w_X and commit it to the networks (Step 2). As soon as τ_l is verified and integrated into the Blockchain network in a block, say B_i, X takes B_i (Step 3), and provides τ_l and B_i to W_X (Step 4). W create the escrow account ϵ. Then, sends it to X with status (Step 5). Finally, X sends τ_0 and ϵ to the Blockchain network. [1]

4.4.3 Payment with Lottery Ticket

The flow diagram is shown in Figure 4.2. In the payment with lottery ticket phase, the payee Y sends pk_{W_Y} (Step 1). The wallet W_X creates a ticket τ_1 and signs it with the secret key sk_{W_X}, and signs the ticket τ_1 with the wallet manufacturer's secret key sk_T. The wallet

[1] The wallet does not check the validity of the escrow transaction τ_0 and ϵ. Payees will reject the ticket which is not transferred from ϵ.

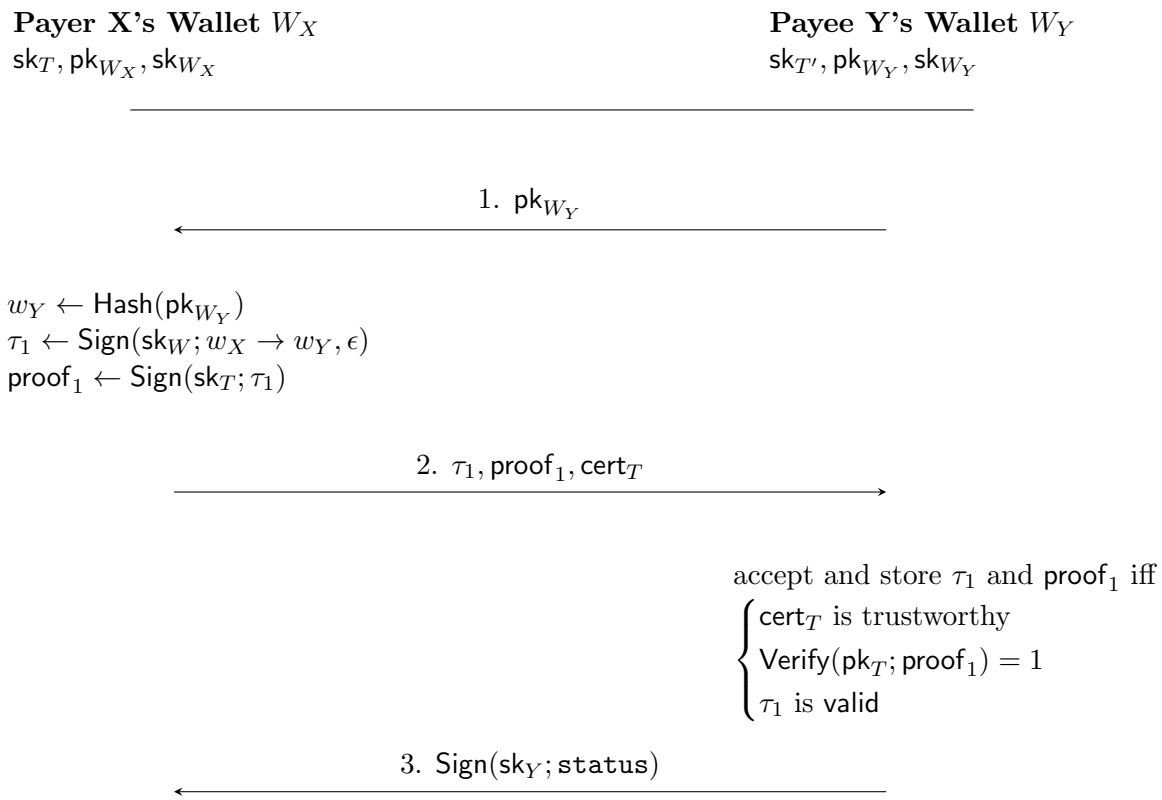


Figure 4.2: Payment with Lottery Tickets

W_X sends ticket τ_1, proof_1, and cert_T to the payee' wallet W_Y (Step 2). If all checks succeed, Y stores τ_1, proof_1, and replies to W_X with the status (Step 3). If the payee Y wants to send the received ticket to another user, the same procedure is followed from Step 1.

4.4.4 Ticket Winning and Revocation

The flow diagram is shown in 4.3. If $\tau \in \mathsf{win}$, Y sends τ and proof to the contract account ϵ (Step 1). If τ is both valid and $\mathsf{eligible}$, the escrow account ϵ signs the escrow transaction τ_0 with w_Y as the destination. The payee Y observes the blockchain network and periodically updates its local chain and confirms τ_0 is valid (Step 2).

If τ is one of the double-spent tickets created by a double-spending attack, the contract account ϵ shows that τ is double-spend one (Step 3). [2] Y initiates revocation by creating a

[2]Double-spending attacks can be perfectly detected and the adversary's address is discovered. See Section 4.7.

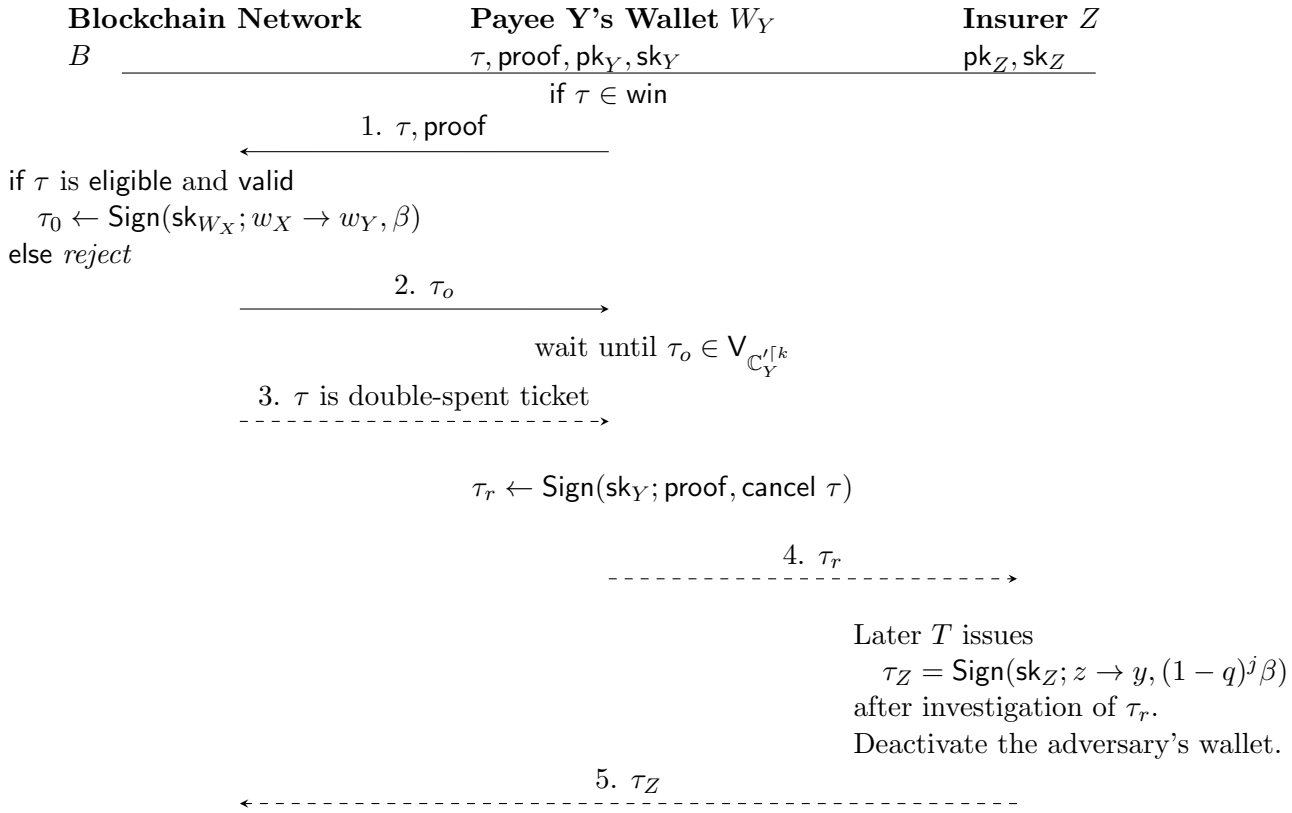


Figure 4.3: Ticket redemption and double-spending wallet revocation protocol

revocation transaction $\tau_r = \mathsf{Sign}(\mathsf{sk}_Y; \mathsf{proof}, \mathsf{cancel}\ \tau)$ and send it to Insurer Z (Step 4). Z investigate τ_r and in order to compensate Y for the damage, issues τ_Z then committed to the Bitcoin network (Step 5). [3]

4.5 Ticket Winning Condition

- $\mathbb{C}$: a blockchain.
- $\mathcal{U}$: a set of users.
- $X, Y \in \mathcal{U}$: (typically, X as a payer, Y as a payee).
- l: the number of double-spent (duplicated) tickets by an adversary
- ϵ: escrow account which has several fields: $(\beta, h_0, \tau_0, p, \mu)$
  - β: the lottery winning amount
  - h_0: the block height containing the escrow account
  - τ_0: escrow creation transaction
  - p: the probability for determining of winning a ticket
  - μ: the fixed number to calculate the winning ticket ($\in \mathbb{N}$)
- τ: a lottery ticket which has several fields : $(A, B, \tau_{pre}, \sigma)$
  - A: a sender
  - B: a receiver
  - τ_{pre}: a reference to a previous ticket or to an escrow account ϵ
  - σ: signature by a sender
- Φ: the cost of breaking a tamper-proof hardware wallet

[3] The compensated amount is the same as the return when received the ticket. See Section 4.6 for the value of a ticket when it is in transfer.

- γ: the blockchain transaction fee

4.5.1 Structure of the ticket

This section describes the structure of the lottery ticket and the design of the ticket winning method. If the ticket is transferable, a blockchain transaction fee is charged when the ticket is won and registered in the blockchain. We introduce a scheme where users who send and receive the ticket share the blockchain transaction fee little by little.

Definition 4. *A lottery tickets τ consists of a sixfold:*

$$(A, B, \tau_{pre}, \sigma_W, \sigma_T, \mathsf{cert}_T) \tag{4.1}$$

where A and B are accounts of a sender and a receiver, respectively. τ_{pre} is a reference to a previous ticket or to an escrow account ϵ. A pair of signatures, σ_W and σ_T, is a multi-signature, where σ_W is signed with a signing key tied with a sender's account and σ_T is signed with a tamper-proof device's signing key to proving that the signing device is trusted verifiable with a certificate cert_T issued by a trusted manufacturer. We denote by σ_A to denote a signature signed by A. The escrow account ϵ further contains $(\beta, h_0, \tau_0, p, \mu)$ to specify the parameters of the transferable transaction, where β is the ticket winning amount, and h_0 is the block height to specify particular VDF values. τ_0 is the escrow creation transaction. p is the probability for determining of winning a ticket. μ is a fixed value used to determine the winning ticket.

For readability, we write a ticket τ as:

$$\tau = (A \to B, \tau_{\text{pre}})_X \,. \tag{4.2}$$

We define $|\tau|$ the "number of generations" of τ, which is the length of the sequence from ϵ to τ. For example, $|\tau| = n$ if there exists a sequence $\tau_1, \ldots, \tau_{n-1}$ such that $\epsilon \prec \tau_1 \prec \tau_2 \prec$

$\cdots \prec \tau_{n-1} \prec \tau$. We define $|\tau| = \infty$ if no such sequence exists [4]. To write compactly, we denote by τ_i the i-th generation of τ.

Definition 5 (Transferred transaction)**.** *Two tickets* $\tau_i = (A \to B, \tau_{pre})_X$ *and* $\tau_{i+1} = \left(A' \to B', \tau'_{pre}\right)_{X'}$ *are said to be* transferred *if and only if following properties satisfies:*

$$\begin{cases} \mathsf{Hash}(\tau_i) = \tau'_{pre} \\ A = X,\ B = A' = X' \\ \mathit{cert}_{T'_X} \text{ is trustworthy} \\ \text{multi-signature } \sigma_{W'_X} \text{ and } \sigma_{T'_X} \text{ are valid} \end{cases} \tag{4.3}$$

Then, we write $\tau_i \prec \tau_{i+1}$.

We write $\tau_i \lll \tau_{i+n}$ if there exists a sequence of ordered lottery tickets $\tau'_1 \prec \ldots \prec \tau'_n$ for $n \geq 1$ and they satisfy $\tau_i \prec \tau'_1$ and $\tau'_n \prec \tau_{i+n}$. In the case where τ has no previous lottery tickets, the ticket is called a 'genesis' ticket. For the genesis tickets τ_1 tied to an escrow account ϵ, we specially denote by $\epsilon \prec \tau_1$ so that a lottery tickets are simply written as:

$$\epsilon \prec \tau_1 \prec \tau_2 \prec \ldots \prec \tau_n. \tag{4.4}$$

Definition 6. *A lottery tickets* τ *is said to be* valid *with respect to a blockchain* $\mathbb{C}$ *for some security parameter* k *if and only if there exists an escrow account* ϵ *and a sequence of transactions* $\tau_1, \ldots, \tau_n$ *such that*

$$\epsilon \in \mathbb{C}^{\lceil k} \quad \textit{and} \quad \epsilon \prec \tau_1 \prec \ldots \prec \tau_n \prec \tau. \tag{4.5}$$

$\mathbb{C}^{\lceil k}$ denotes the set of blocks that are k or more blocks before the beginning of the blockchain. This notion is borrowed from Garay et al [GKL15].

[4] For practical purposes, we assume that the height of τ can only be measured when all tickets in the sequence from ϵ to τ are given. Even if such a sequence exists, the height of τ is considered to be ∞ unless the entire sequence is specifically presented.

4.5.2 Ticket Winning Condition

This section describes the design of the ticket winnings.

Definition 7. *$\tau_{i,v}$ is said to be* win *if and only if the following properties satisfies:*

$$\mathsf{win} = \left\{ \tau_v \;\middle|\; \mathrm{VDF}\left(h_0 + v \cdot \mu\right) < D, v \in \mathbb{N} \right\} \tag{4.6}$$

where v is the number of generations of τ and μ is the fixed number specified in the escrow account ϵ.

h_0 is registered in ϵ, which specifies the block height at which ϵ would be registered. The probability p is calculated using a simple Verifiable Delay Function (VDF) [BBBF18]. The calculation can be done after a certain period of time has elapsed from when the ticket is transferred according to the number of generations. For example, if a ticket with $h_0 = 100$, $\mu = 5$, and $v = 3$ is received, the VDF value will be known when the block height of 115 is confirmed.

As described in the next Section 4.6, even though the ticket meets the requirements win, the ticket may be used as payment instead of getting the winning amount β. If a ticket $\tau \in \mathsf{win}$ has already been transferred, the user with the most recent ownership can get the winning amount β.

Definition 8. *τ_v is said to be* eligible *if and only if the following properties satisfies:*

$$\mathsf{eligible} = \left\{ \tau_v \;\middle|\; {}^{\exists}\tau_{v'} \in \mathsf{win} \wedge \tau_{v'} \lll \tau_v \right\} \tag{4.7}$$

eligible ticket will be considered as the final winning ticket. Thus, the user who has the eligible ticket can get β from the escrow account ϵ.

4.6 Proportional Fee Scheme

In this section, we consider the blockchain transaction fee to transfer the winning amount to the winner's address and the value of the ticket in the transfer process.

In our transferable scheme, it is not beneficial for the issuer to bear the blockchain transaction fee. Since when the issuer bears the blockchain transaction fee, the amount available for payment is $\beta - \gamma$, which does not provide any advantage for the issuer to use the transferable scheme.

We propose a novel *Proportional Fee Scheme*. The process is depicted in Figure 4.4. This scheme is where each time a payer transfers a ticket; the payer bears the fee based on the number of generations of the ticket. When a payer sends τ_j to the payee, in return, the payee gives goods or services worth $(1-q)^j\beta$.

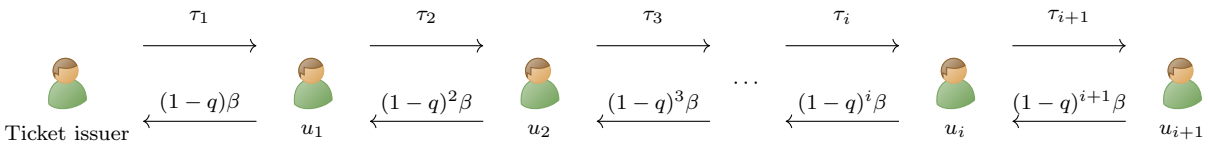


Figure 4.4: Proportional Fee Scheme

Definition 9 (Proportional fee scheme)**.** *Let q be the lottery ticket transaction fee rate. Suppose a payer sends a ticket τ_i, and in return, the payee gives goods or services worth $(1-q)^i\beta$ to the sender. The fees borne by the payment is $(1-q)^{i-1}q\beta$.*

Specifically, the fee for each payment is $\tau_{i-1} - \tau_i = (1-q)^{i-1}q\beta$, and the profit (income − expenditure) when $\tau_i = \mathsf{eligible}$ is $\beta - (\tau_i + \gamma) = (1-(1-q)^i)\,\beta - \gamma$ where γ is the blockchain transaction fee.

Suppose the ticket satisfies the win condition before the accumulated fees exceed the blockchain transaction fee γ. In this case, the user may decide whether to send it to the blockchain network and get β or transfer the ticket to another user as payment. Specifically, the user can profit from the eligible ticket by getting the winning amount β under the following condition:

$$\left(1-(1-q)^i\right)\beta > \gamma. \tag{4.8}$$

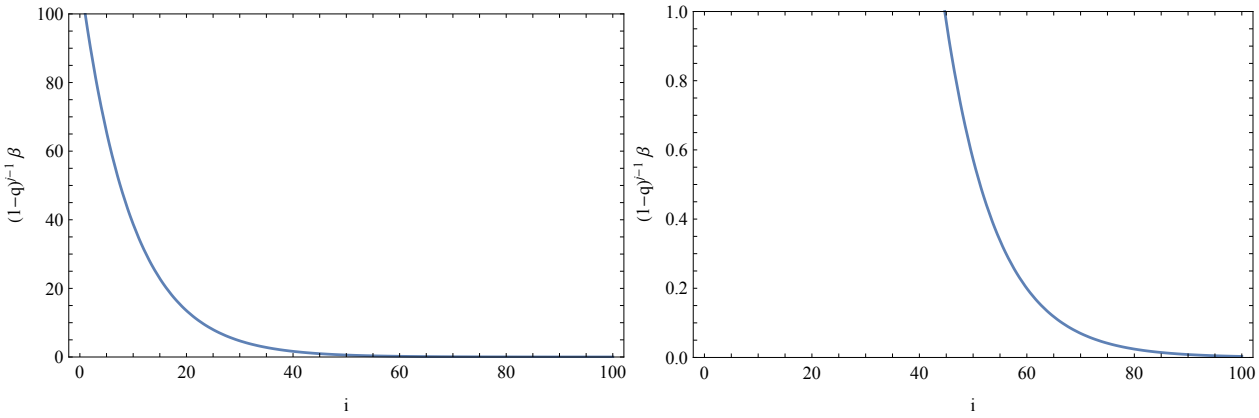
If the ticket satisfies the win condition is transferred to another user, the ticket is distributed as eligible and can be sent to the blockchain in any subsequent generation. Naturally, the ticket will be sent to the blockchain network in the generation that satisfies the equation 4.8.

This scheme has the advantage that the payment fee can be smaller than the blockchain transaction fee. The average transaction fee for cryptocurrencies, especially Bitcoin, is approximate $2 [YCH].

In our transferable scheme, let $\beta = \$100$, $p = \frac{1}{100}$ and $q = \frac{1}{10}$, the ticket value per generation is depicted in Figure 4.5. As we can see from Figure 4.5b, the value of the ticket falls below $1 from approximately $i = 50$. Figure 4.6 shows the frequency of the fee, and we can see that there are more than 50 transactions whose value is less than $1. Since the fee per payment is roughly $q = \frac{1}{10}$, the fee for a $1 transfer is about 10 cents.

Both the existing Lottery scheme and our Transferable scheme can aggregate blockchain transactions by the winning probability p. The difference is that our transferable scheme does not increase the gambling potential, even making the winning probability p smaller. In the existing scheme, the smaller p is, the lower the probability that the payee will win the ticket, which makes the income more unstable for the payees. In our transferable scheme, even if the ticket is not winning, the payee can use it for payment by paying a smaller fee than the blockchain transaction fee.

There is a concern that the sizeable winning amount β decreases the velocity of the ticket. This is because if there is a large gap between the winning amount β and the value of the ticket, the profit of winning $\beta - \gamma$ will be more significant. Therefore, it is best for recipients to decide whether to use the ticket for payment after confirming their winnings, which causes the velocity of the ticket to be slow. The solution is not to make the winning amount β too high. In addition, if we set the winning amount β to a value almost equal to



(a) < 100 (b) < 1

Figure 4.5: Ticket value per generation when $\beta = \$100$, $p = \frac{1}{100}$ and $q = \frac{1}{10}$. Figure 4.5a shows the value of a ticket, depending on the number of generations i of the ticket. As shown in Figure 4.5b, at roughly 50 generations or more, the value of the ticket is less than \$1, making it micropayment.

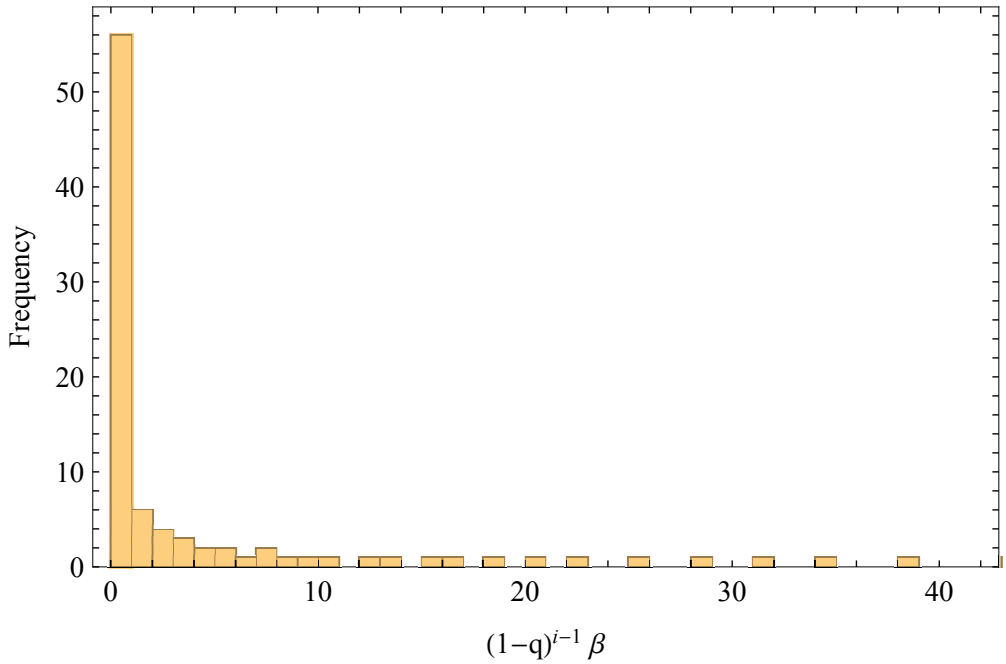


Figure 4.6: Frequency of the ticket values, same as Figure 4.5, when $\beta = \$100$, $p = \frac{1}{100}$ and $q = \frac{1}{10}$. Roughly more than 50 generations are worth less than \$1.

the blockchain transaction fee γ, the profit of winning will be very small; thus, the velocity of the ticket will not be affected.

4.7 Security Properties

As far as the tamper-resistant assumption holds, double-spending attacks can not be performed theoretically. However, in reality, the tamper-proof hardware wallet could be broken, leading to double-spending attacks. Thus, instead of requiring a penalty escrow, we design security from the perspective of whether the utility an adversary can gain from the attack exceeds the cost of breaking the tamper-resistant hardware.

Definition 10 (κ-tamper proof)**.** *A device is called κ-tamper proof if it satisfies the following conditions:*

1. *tamper-proof hardware is the hardware that prevents an adversary from stealing and changing stored data.*

2. *the device is either completely broken/tampered or working perfectly with probability κ and $(1-\kappa)$, respectively* [5].

3. *broken/tampered is a state in which all confidential information inside the device, including the secret key, has been leaked to the adversary.*

We assume each device is in a state either completely broken/tampered or working perfectly. They occur with probabilities κ and $(1-\kappa)$, respectively. As long as the behavior is observed from outside, it is not possible to distinguish between a device that is operating correctly and a device that adversary control the correct device who have an access to its internal key.

[5] In reality, the adversaries are biased, but we assume it can not be distinguishable from a legitimate user from outside.

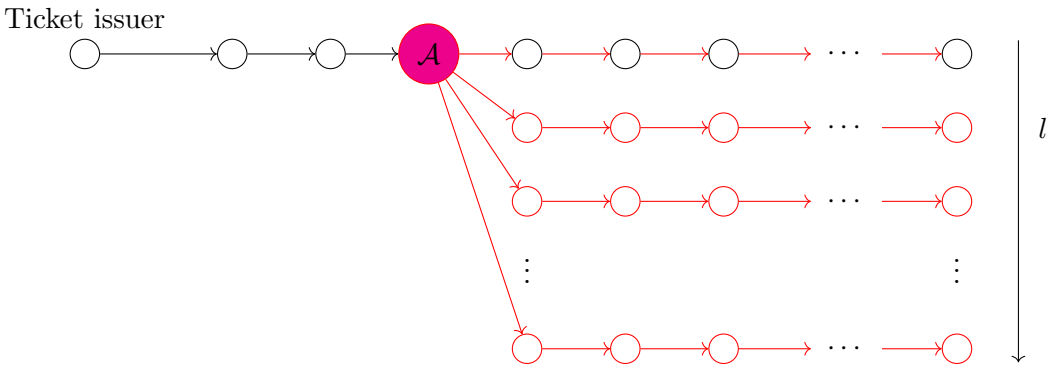


Figure 4.7: Double-spending Attack

Double-spending Attack

Double-spending attacks are an attack that makes a profit by duplicating and double-spending the received ticket. In our transferable scheme, we assume that the adversary breaks the κ-tamper proof and receives or issues tickets, then transfers with different addresses (wallets). We call the tickets created by the double-spending attacks "duplicated."

In our transferable scheme, we assume an adversary can gain profit up to $l \cdot \beta$ where l is the number of duplicated tickets. This is illustrated in Figure 4.7.

4.7.1 Detection Methods

This section describes how to detect the double-spending attacks and find the adversary's address. We introduce two methods to detect the attack and find the adversary's address perfectly. When the adversary's address is found, we assume that the address is broadcasted to all users, and then the adversary's address is rejected by all users. An adversary can not profit unless the maximum expected utility he can gain from a single attack exceeds the cost of breaking the κ-tamper proof wallet.

Note that the following two detection methods require that the receiver be online at the time of receipt. Conversely, the payer does not need to be online.

Definition 11 (Fork of transferred transactions)**.** *Given two series of transactions initiated with the same escrow account* $\tau = \{\epsilon \prec \tau_1 \prec \cdots \prec \tau_n\}$*, and* $\tilde{\tau} = \{\tilde{\epsilon} \prec \tilde{\tau}_1 \prec \cdots \prec \tilde{\tau}_{n'}\}$*, the series of transactions are said to be* Fork *if and only if it satisfies:* $^{\exists}k \in \mathbb{Z}_{\geq 0}$*,*

$$\begin{cases} \epsilon = \tilde{\epsilon} \\ \tau^{\lceil k} \lessdot \tilde{\tau} \ \wedge \ \tilde{\tau}^{\lceil k} \lessdot \tau \\ \tau^{\lceil k-1} \not\lessdot \tilde{\tau} \ \vee \ \tilde{\tau}^{\lceil k-1} \not\lessdot \tau. \end{cases} \tag{4.9}$$

Assume the users monitor the blockchain, and after the eligible ticket is registered in the blockchain, the users check the eligible ticket against the ticket they have received.

Theorem 2 (Fork Detection)**.** *Given two series of transactions* $(\tau, \tilde{\tau}) \in$ Fork*, there exists an efficient fork detection algorithm that outputs the double-spending transactions.*

Proof. Assume there exists two series of transactions τ and $\tilde{\tau}$ such that $(\tau, \tilde{\tau}) \in$ Fork, and τ is eligible and registered in the blockchain, the user who has $\tilde{\tau}$ reports a double-spending attack. Put the two transactions τ, $\tilde{\tau}$ as $\tau = \{\epsilon \prec \tau_1 \prec \cdots \prec \tau_n\}$, and $\tilde{\tau} = \{\tilde{\epsilon} \prec \tilde{\tau}_1 \prec \cdots \prec \tilde{\tau}_{n'}\}$. From the Definition 11, there exists $k \geq 0$ that satisfies the condition (4.9). We assume $n \geq n'$ without loss of generality. Then, $\tau^* = \{\epsilon \prec \tau_1 \prec \cdots \prec \tau_{n-k}\}$ is the longest common prefix, and τ_{n-k+1} and $\tilde{\tau}_{n-k+1}$ are the double spending transactions. □

Definition 12 (Collision of transferred transactions)**.** *Assume that each of the u users has* $\alpha\,(\geq 2)$ *addresses. If an adversary sends at least two duplicated tickets to any one of u users, the 'collision' occurs.*

We adopt a *round scheme* so that the adversary can not profit when the collision is detected. We divide the ticket-sending procedure into three rounds. The procedure is illustrated in Figure 4.8. **Round** 1) The adversary sends the tickets to the honest payees. The payee checks the received tickets for the collisions. **Round** 2) If the payee finds the collision, broadcasts τ and $\tilde{\tau}$ to the honest users. **Round** 3) If the collision is not detected,

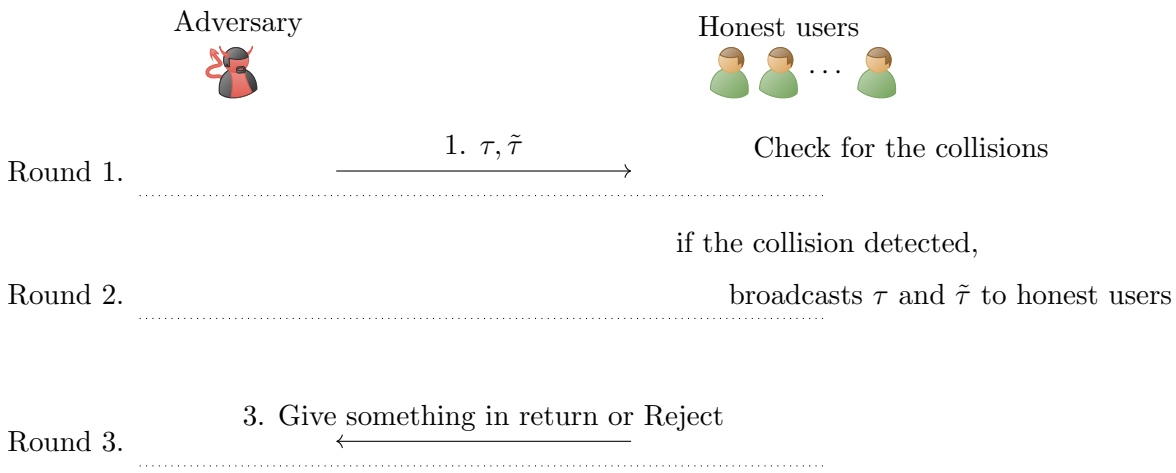


Figure 4.8: Collision detection round

the payee gives products or services to the payer in return. If the collision is detected, the adversary's address is rejected and will never be accepted by all honest users.

Theorem 3 (Collision Detection)**.** *Let u be the number of users who participate in transfer scheme. By collision detection and round scheme, the expected utility of double-spending attack $\mathbb{E}_d$ is upper-bounded by the following inequality:*

$$\mathbb{E}_d \leq \sqrt{\frac{u}{e}}\beta. \tag{4.10}$$

Proof. As stated in the Definition 12, we assume a uniform distribution where each user has α addresses[6]. This must be the case where the adversary chooses uniformly l different addresses from the total of αu addresses. By the round scheme, the adversary can not profit if a single user address is chosen more than once.

Let $p(l; u)$ be the probability that at least one user address is chosen more than once. This probability is described as follows:

$$p(l; u) \approx 1 - e^{-\frac{l^2}{2u}}. \tag{4.11}$$

[6]In reality, the number of addresses each user has is considered more likely to follow an exponential distribution. It is an unfavourable assumption that all user have the same number of addresses α.

Assume that the adversary double-spent l tickets with a maximum value of β per ticket. The adversary's expected utility value is

$$\mathbb{E}_d < \max_l \Big\{ l\beta \cdot (1 - p(l;u)) \Big\}. \tag{4.12}$$

Thus, $\mathbb{E}_d$ is at most $\sqrt{\frac{u}{e}}\beta$ when $l = \sqrt{u}$. □

In our transferable scheme, the double-spending attack is perfectly detected, and the address used in the attack will be rejected by all users. Therefore, it is not profitable for the adversary unless the cost of breaking a single tamper-proof wallet exceeds the maximum expected value gained by the attack. Specifically, the adversary can not profit under the following conditions:

$$\sqrt{\frac{u}{e}}\beta < \Phi \tag{4.13}$$

where Φ is the cost of breaking κ-tamper proof wallet.

As an example, consider the maximum expected utility value $\mathbb{E}_d$ with $u = 1,000,000$ and $\beta = \$100$. Applying the equation 4.10 produces $\mathbb{E}_d \lessapprox \$60,700$.

4.8 Conclusion

In this chapter, we introduce the first transferable decentralized probabilistic micropayment scheme and the proportional fee scheme. The feature of our scheme is that the ticket is transferable. Therefore, the ticket winning probability can be much smaller than the existing methods. Thus we can aggregate a larger number of transactions into one and can increase the blockchain throughput. Also, the proportional fee scheme can make the transaction fee smaller via the lottery ticket than on the blockchain.

Our scheme only assumes a tamper-proof device, and the ticket transfer protocol is simple, requiring only a digital signature. The tamper-proof assumptions can be achieved by SE (Secure Elements) such as SIM cards, which are widely used in Smartphones. Since

the computational resources of SE are limited, the concern arises that it is not impractical to perform all operations in the SE. In our scheme, the operations to be performed in the SE can be limited to the prevention of double-spending. On the other hand, operations that are not related to double-spending prevention can be executed in the regular application area. Therefore, since the use of SE's computational resources can be minimized, it would be feasible to realize our scheme on mobile devices such as smartphones. Specifically, the operations to be performed in the SE are checking whether a ticket is valid, creating a key pair, and signing at the time of money transfer. On the regular application side, the operations are performed to avoid duplicated tickets (e.g., collision and fork detection) and check for winning tickets.

We consider that our transferable scheme is not a singular way of transferable lottery tickets but a system similar to the circulation of paper and coins issued by central banks. We will use blockchain to achieve this. We believe that our scheme can be applied not only to micropayments but also to high-value payment transactions.

4.8.1 Open problem

We proposed a scheme to increase the blockchain's throughput by a factor of v (100× as an example). Our scheme increases the throughput by v, but the size of the ticket also increases by v linearly. The size of the EPID signature is 171 bytes [BL12, BL10], and if the receiving address is a hash value with the digest length of 512 bits, the aggregation ticket size is roughly 29.9 KB when $v = 100$. Although $v = 100$ is negligibly small as a verification cost for the blockchain, it is desirable to avoid increasing the size in proportion to v in practice.

In order to prevent the ticket size from increasing in proportion to the number of times the ticket is sent, schemes such as aggregate signatures or one-way accumulators may be used.

Chapter 5

VeloCash: Anonymous Decentralized Probabilistic Micropayments with Transferability

5.1 Introduction

In the previous Chapter 4, we have proposed a transferable decentralized probabilistic micropayments scheme. In this chapter, we propose Decentralized Probabilistic Micropayments with Transferability which preserves *Anonymity*, named VeloCash.

Similar to Chapter 4, we realize the lottery scheme and the proportional fee scheme under anonymous conditions. We also realize "anonymity with transparency" by fully detecting double-spending attacks and revoking the adversary.

5.2 Contribution

The contribution of the chapter is threefold:

- Extended anonymity notions for blockhchain-based decentralized transferable payment schemes:

  All of the previous anonymity notions [CG08, BFQ21] assume the existence of the bank as the central authority. Thus, applying those anonymity notions to the blockchain-based decentralized payment schemes is not straightforward. In this chapter, we introduced the generalized anonymity notions of transferred electronic cash schemes to cover both centralized and decentralized payment schemes.

- Revokable anonymity extension for attested execution secure processors:

  Attested Execution Secure Processors (AESP) [PST17] is the abstraction of tamper-proof secure processors, which enforces every installed program to attach an attested signature as a proof to show that the output is the result of the execution of the program. In this chapter, we propose a new mechanism to revoke tampered AESP's utilizing the idea of key extractor when double-spending is detected. In order not to be too abstracted, our key extractor is defined over Enhanced Privacy Identifier (EPID) [BL10], but our technique can be applied to any Direct Anonymous Attestation (DAA) scheme [BCC04] with Fiat-Shamir proof of knowledge for NP relation.

- Secure construction of probabilistic anonymous payments with transferability:

  We proposed VeloCash, an anonymous probabilistic micropayment scheme with transferability. The construction satisfies all the security and anonymity notions claimed in the theorems.

5.3 Preliminary

As in Chapter 4, the transferable payment scheme is realized using tamper-proof devices. In this study, which realizes the anonymity of the transferable payment scheme, we also assume tamper-proof devices. More specifically, we use Attested Execution Secure Processors

(AESP), that abstracts "attested execution" secure processors. In the "attested execution" of AESP, the output of the installed program is digitally signed with the secret key in the AESP. The signature enables verification that the output is indeed from a legitimate AESP.

Direct Anonymous Attestation (DAA) has been used for the attested signature. DAA can be seen as group signatures [BMW03] but differs from group signatures in that DAA does not have an *opening* algorithm that allows the group manager to obtain the identity of the signer from the signature. Instead of having *opening* function, DAA has a so-called "revocation function". Suppose a particular hardware module has been broken and its secret key has been compromised; the secret key is placed on the revocation list. When a verifier receives the signature, he can verify whether it is signed with the secret key on the revocation list.

In this study, we use AESP to send and receive tickets, and DAA is used for attested signatures and for sending tickets. There are three reasons to adopt DAA: 1) For anonymization of attested signature. 2) To extract the adversary's secret if the double-spending attacks are performed 3) To revoke the extracted adversary's secret key. In particular, extracting and revoking the adversary's secret key is a strong motivation for adopting DAA. We adopt Enhanced Privacy ID (EPID), a DAA scheme that uses Fiat-Shamir proof of knowledge for NP relation. Thus, since the Extractor can be configured, it is possible to extract and revoke the adversary's secret key from the double-spending tickets with the same commitment but different challenges.

A possible alternative to anonymization other than DAA is using ring signatures. Traceable ring signatures [FS07, Fuj11] limit the number of times a secret key can sign transactions, making it possible to detect double-spending attacks. However, ring signatures grow proportionally to the size of the anonymity set. There is an inevitable trade-off between anonymity and signature size while DAA signature size is constant.

5.3.1 Direct Anonymous Attestation (DAA)

Direct Anonymous Attestation (DAA) remote authentication scheme for trusted hardware module has been proposed by Brickell et al. [BCC04]. DAA has been adopted by the Trusted Computing Group (TCG).

There are multiple DAA schemes have been proposed [SRC15, GHS11, BCL09, CPS10, Che10]. In this paper, we adopt *Enhanced Privacy ID (EPID)* scheme proposed by Brickell and Li [BL10, BL12]. EPID is a scheme proposed by Intel Corporation and is already in use in the real world, embedded in chipsets such as Intel SGX. EPID is compliant with International Standards Organization standard ISO/IEC 20008, 20009 and approved by the Trusted Computing Group (TCG) as the recommended scheme. Intel has made EPID an open-source to processor manufacturers under the Apache 2 license. In 2015, Microchip and Atmel announced that they had licensed the EPID technology [Cora, Corb].

5.3.2 Specification of EPID

EPID is one of the DAA schemes which preserves anonymity proposed by Brickell and Li [BL10, BL12]. There are two types of revocation in EPID, the secret key based revocation, and the signature based revocation. In the secret key based revocation, put the secret key into the secret (private) key based revocation list $\mathsf{Priv\text{-}RL}$. In the signature based revocation, put the signature into the revocation list $\mathsf{Sig\text{-}RL}$. The verification process will invalidate the signatures with the keys in the corresponding revocation lists in both revocation schemes.

There are four entities in EPID: an issuer $\mathcal{I}$, a revocation manager $\mathcal{R}$, platforms[1] $\mathcal{P}_i$, and verifiers $\mathcal{V}$. The scheme consists of five polynomial-time algorithms:

$$\Pi_{\text{EPID}} = (\mathsf{Setup}, \mathsf{Join}, \mathsf{Sign}, \mathsf{Com}, \mathsf{Verify}, \mathsf{Revoke})\,. \tag{5.1}$$

[1]In the original definition by Brickwell and Li [BL10, BL12], platforms are the secure hardware-based signing entities such as SGX in Intel processors. Pass et al.[PST17] refers to the signing entities to produce attested signatures as ideal functionality $\mathcal{G}_{\text{att}}$, and they used the term 'platform' for the entities who is allowed to invoke functionalities in $\mathcal{G}_{\text{att}}$.

Setup :

$$(\mathsf{gpk}, \mathsf{isk}) \leftarrow \mathsf{Setup}(1^\lambda) \tag{5.2}$$

Issuer $\mathcal{I}$ takes the security parameter 1^λ as input and outputs a group public key gpk and an issuing private key isk.

Join :

$$\langle \perp, \mathsf{sk}_i \rangle \leftarrow \mathsf{Join}_{\mathcal{I},\mathcal{P}_i} \langle (\mathsf{gpk}, \mathsf{isk}), \mathsf{gpk} \rangle \tag{5.3}$$

Issuer $\mathcal{I}$ is given gpk and isk. $\mathcal{P}_i$ is given gpk and outputs a secret key sk_i.

Sign :

$$\sigma/\perp \leftarrow \mathsf{Sign}(\mathsf{gpk}, \mathsf{sk}, m, \mathsf{Sig\text{-}RL}) \tag{5.4}$$

The above is a probabilistic signature algorithm which on input $\mathsf{gpk}, \mathsf{sk}$, a message m and a signature based revocation list $\mathsf{Sig\text{-}RL}$ outputs a signature σ, or $\perp$ if sk has been revoked in $\mathsf{Sig\text{-}RL}$.

Here, we define the deterministic version of the same signature algorithm:

$$\sigma/\perp \leftarrow \mathsf{Sign}(\mathsf{gpk}, \mathsf{sk}, m, \mathsf{Sig\text{-}RL}; \mathsf{r}) \tag{5.5}$$

where r is a randomness. Thus, it always outputs the same signature σ if all inputs are the same.

For simplicity, we sometimes omit public parameters from expressions such that $\mathsf{Sign}_{\mathsf{sk}}(m)$ for probabilistic signature algorithms and $\mathsf{Sign}_{\mathsf{sk}}(m; \mathsf{r})$ for deterministic signature algorithms, respectively.

Com :

$$(x, \mathtt{com}) \leftarrow \mathsf{Com}(\mathsf{gpk}, \mathsf{sk}; \mathsf{r}) \tag{5.6}$$

The probabilistic commitment generation algorithm on input $\mathsf{gpk}, \mathsf{sk}$ and r outputs

x and $\mathtt{com}$. This is not defined in Brickwell and Li [BL10, BL12]. We introduce this function for technical reasons described later in Definition 20. Assuming the signature scheme uses Fiat-Shamir proof of knowledge for NP relation $(x, \omega) \in R_\lambda$, there exists a key extractor, as we will see in Definition 20. When used together with the deterministic signature algorithm $\mathsf{Sign}_{\mathsf{sk}}(m; \mathsf{r})$ with the same randomness r, we can utilize these functions to identify double-spenders. Sometimes we also omit public parameters and write as $\mathtt{com}_{\mathsf{sk}}(\mathsf{r})$ for simplicity[2].

Verify :

$$\{0, 1\} \leftarrow \mathsf{Verify}(\mathsf{gpk}, m, \mathsf{Priv\text{-}RL}, \mathsf{Sig\text{-}RL}, \sigma)$$

On input gpk, a secret key based revocation list Priv-RL, a signature based revocation list Sig-RL, a message m and a signature σ, the function outputs either 1 if the signature is valid and 0 for invalid. Verify outputs 0 (invalid) when either case that σ is not a valid signature on m or that σ has been revoked.

Further, EPID[BL10, BL12] defined the two revocation algorithms: secret key based revocation and signature based revocation, where original DAA[BCC04] only defines the former. Our scheme does not utilize the signature based revocation, and hence our construction does not depend on the EPID signature based revocation. The two revocation algorithms are defined as follows:

Revoke - secret key based revocation

$$\mathsf{Priv\text{-}RL} \leftarrow \mathsf{Revoke}(\mathsf{gpk}, \mathsf{Priv\text{-}RL}, \mathsf{sk}) \tag{5.7}$$

Given gpk, Priv-RL, and sk, $\mathcal{R}$ updates Priv-RL by adding sk into Priv-RL.

[2] com satisfies the hiding and biding properties defined in Damgård [Dam99]

Revoke - Signature based revocation

$$\begin{array}{c} \mathsf{Sig\text{-}RL} \\ \leftarrow \mathsf{Revoke}(\mathsf{gpk}, \mathsf{Priv\text{-}RL}, \mathsf{Sig\text{-}RL}, m, \sigma) \end{array} \tag{5.8}$$

Given $\mathsf{gpk}, \mathsf{Priv\text{-}RL}, \mathsf{Sig\text{-}RL}, m$, and σ, $\mathcal{R}$ updates $\mathsf{Sig\text{-}RL}$ by adding σ into $\mathsf{Sig\text{-}RL}$ after verifying σ.

Security Definition of EPID

An EPID scheme is secure if it satisfies the following three requirements: correctness, anonymity, unforgeability [BCC04, BL10].

The correctness requires that every signature a platform generates is valid except when the platform is revoked by the secret key based revocation or the signature based revocation.

Theorem 4. *(Theorem 4 of [BL09]) The EPID scheme is correct.*

The EPID scheme is anonymous if the adversary can not determine from the signature the secret key used to generate the signature.

Theorem 5. *(Theorem 5 of [BL09]) An EPID scheme is anonymous in the random oracle model under the decisional Diffie-Hellman assumption in G_3.*

The EPID scheme is unforgeable if the adversary can not forge a valid signature with all secret keys known to the adversary that has been revoked.

Theorem 6. *(Theorem 6 of [BL09]) The EPID scheme is unforgeable in the random oracle model under the strong Diffie-Hellman assumption in (G_1, G_2).*

See Appendix A for the construction of EPID and Appendix B for more detailed definitions.

5.4 Anonymous Ticket Transfer Protocol

The results in the preceding section show that the cost of breaking a κ-tamper proof wallet must exceed the expected utility of an adversary. In this section, we follow the formal abstraction of attested execution secure processors [PST17] with adequate tamper-resistance (whose breaking cost exceeds Φ).

All previously proposed anonymous and transferable electronic cash schemes[OO92, CP93, CG08, BCFK15, BFQ21] assume that:

1. the existence of central authority (bank), and
2. only the bank can detect double-spending.

Our decentralized blockchain-based transferable payment scheme is described in the previous section; however, every player is eligible to set up an escrow account and initiate offline transferable payments; hence no central authority (bank) exists. Further, our collision-detection and fork-detection techniques enable every recipient of transferable payments to detect double-spending and publish the evidence on the blockchain. That is, every player can detect double-spending potentially. Finally, to capture the anonymity in these decentralized settings, we will define generalized anonymity notions in the following subsections.

Algorithms (CG08)

A conventional transferable e-cash system generally involves two types of players: a bank $\mathcal{B}$ and a user $\mathcal{U}$. Whereas a blockchain-based transferable e-cash system has no banks, a blockchain $\mathbb{C}$ takes the role of a bank $\mathcal{B}$. $\mathbb{C}$ can be regarded as $\mathcal{B}$ that holds no secret information and publishes all of its views and the deposited coin list $\mathcal{L}$.

A coin is represented by an identifier id and some values π needed to prove its validity.

- $\mathsf{ParamGen}(1^\lambda)$ is a probabilistic algorithm that outputs the parameters of the system param (including the security parameter λ). We assume all functions take param as their inputs unless otherwise specified.

Note: param may contain the genesis block of the blockchain $\mathbb{C}$. For schemes assuming DAA-based anonymous signature schemes with tamper-proof devices, $\mathsf{ParamGen}$ generates the DAA public key and secret key pair $(\mathsf{gpk}, \mathsf{gsk})$ and param contains gpk for the verification of anonymous signatures.

- $\mathsf{BKeyGen}(\mathsf{param})$ (resp. $\mathsf{UKeyGen}(\mathsf{param})$) is a probabilistic algorithm executed by $\mathcal{B}$(resp. $\mathcal{U}$) that outputs the key pair $(\mathsf{sk}_{\mathcal{B}}, \mathsf{pk}_{\mathcal{B}})$ (resp. $(\mathsf{sk}_{\mathcal{U}}, \mathsf{pk}_{\mathcal{U}})$).

  Remark. *Blockchain-based transferable e-cash systems have the blockchain* $\mathbb{C}$ *in place of the bank* B*, and* $\mathbb{C}$ *has no secret information. Thus,* $\mathsf{BKeyGen}(\mathsf{param})$ *outputs nothing such that* $(\mathsf{sk}_{\mathcal{B}}, \mathsf{pk}_{\mathcal{B}}) = (\perp, \perp)$.

  Remark. *For schemes assuming DAA-based anonymous signature schemes with tamper-proof device,* $\mathsf{UKeyGen}(\mathsf{param})$ *invokes* $\mathsf{Join}(\mathsf{gpk})$ *protocol with the manufacture who holds* gsk*, and stores* sk_i *securely in the tamper-proof device within a platform* $\mathcal{P}_i$*. Such systems share the group public key* gpk *and no individual public key* pk_i *for the platform* $\mathcal{P}_i$ *exist. Hence,* $(\mathsf{sk}_{\mathcal{U}}, \mathsf{pk}_{\mathcal{U}}) = (\mathsf{sk}_i, \perp)$.

- $\mathsf{Withdraw}\left(\mathcal{B}\left(\mathsf{sk}_{\mathcal{B}}, \mathsf{pk}_{\mathcal{B}}, \mathsf{pk}_{\mathcal{U}}\right), \mathcal{U}\left(\mathsf{sk}_{\mathcal{U}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{pk}_{\mathcal{B}}\right)\right)$ is an interactive protocol where $\mathcal{U}$ withdraws from $\mathcal{B}$ one coin. At the end, $\mathcal{U}$ either gets a coin $C = (\mathsf{id}, \pi)$ and outputs OK, or outputs $\perp$. The output of $\mathcal{B}$ is either its view $\mathcal{V}_{\mathcal{B}}^{\mathrm{W}}$ of the protocol (including $\mathsf{pk}_{\mathcal{U}}$), or $\perp$.

  Remark. $\mathcal{V}_{\mathcal{B}}^{\mathrm{W}}$ *including* $\mathsf{pk}_{\mathcal{U}}$ *is published on the blockchain* $\mathbb{C}$ *in blockchain-based systems. The same amount of the coin* C *is funded in an address* ϵ *on the blockchain* $\mathbb{C}$*, where* ϵ *is related to the public key* $\mathsf{pk}_{\mathcal{U}}$ *and spendable using the secret key* $\mathsf{sk}_{\mathcal{U}}$.

- $\mathsf{Spend}\left(\mathcal{U}_j(\mathsf{id}, \pi, \mathsf{pk}_{\mathcal{U}_i}), \mathcal{U}_i\left(\mathsf{sk}_{\mathcal{U}_i}, \mathsf{pk}_{\mathcal{B}}\right)\right)$ is an interactive protocol where $\mathcal{U}_j$ gives a coin to $\mathcal{U}_i$. At the end, either $\mathcal{U}_i$ outputs a coin $C = (\mathsf{id}_C, \pi_C)$ or $\perp$, and either $\mathcal{U}_i$ saves that C is a spent coin and outputs OK, or $\mathcal{U}_i$ outputs $\perp$.

- $\mathsf{Deposit}\left(\mathcal{U}\left(\mathsf{id}, \pi, \mathsf{sk}_{\mathcal{U}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{pk}_{\mathcal{B}}\right), \mathcal{B}\left(\mathsf{sk}_{\mathcal{B}}, \mathsf{pk}_{\mathcal{B}}, \mathsf{pk}_{\mathcal{U}}, \mathcal{L}\right)\right)$ is an interactive protocol where $\mathcal{U}$ deposits a coin (id, π) at the bank $\mathcal{B}$. If (id, π) is not consistent/fresh, then $\mathcal{B}$ outputs $\perp_1$. Else, if id already belongs to the list of spent coins $\mathcal{L}$, then there is an entry (id, π') and $\mathcal{B}$ outputs $(\perp_2, \mathsf{id}, \pi, \pi')$. Else, $\mathcal{B}$ adds (id, π) to its list $\mathcal{L}$, credits $\mathcal{U}$'s account, and returns $\mathcal{L}$. $\mathcal{U}'$'s output is OK or $\perp$.

  Remark. *Blockchain-based transferable e-cash systems have the blockchain* $\mathbb{C}$ *in place of the bank* B. *In these systems,* $\mathsf{Deposit}$ *is conducted as a transfer of money from the address* ϵ *to the address of* $\mathcal{U}$ *if the coin* (id, π) *is consistent and the money in the address* ϵ *has never spent.*

  Remark. *In the blockchain-based transferable e-cash systems,* $\mathcal{L}$ *is a part of the blockchain* $\mathbb{C}$. *If* id *already belongs to the blockchain* $\mathbb{C}$*, or its list of spent coin* $\mathcal{L}$*, then there is an entry* (id, π') *and every user* $\mathcal{U}'$ *who received* (id, π) *can output* $(\perp_2, \mathsf{id}, \pi, \pi')$. *Else,* (id, π) *is published on the blockchain* $\mathbb{C}$ *and added to its list of spent coins* $\mathcal{L}$. $\mathcal{U}$*'s output is* OK *or* $\perp$.

- $\mathsf{Identify}\left(\mathsf{id}, \pi, \pi'\right)$ is a deterministic algorithm executed by $\mathcal{B}$ that outputs a public key $\mathsf{pk}_{\mathcal{U}}$ and a proof Π_G.

  Remark. *In the blockchain-based transferable e-cash systems,* $\mathsf{Identify}\left(\mathsf{id}, \pi, \pi'\right)$ *is a deterministic algorithm executed by every user* $\mathcal{U}$ *that outputs a double-spender's secret key* sk_i *since DAA-based anonymous signature schemes have no individual public keys* pk_i. *The systems with such anonymous schemes can not output* $\mathsf{pk}_{\mathcal{U}}$ *of the double spender. Therefore, in such systems,* $\mathsf{Identify}\left(\mathsf{id}, \pi, \pi'\right)$ *outputs* $(\perp, \Pi_G)$. *We consider that* Π_G *includes the double-spenders secret keys* sk_i *and the following* $\mathsf{VerifyGuilt}((\mathsf{id}, \pi), \Pi_G)$ *determines whether the coin* (id, π) *is spent by a guilty user of not.*

- $\mathsf{VerifyGuilt}\left(\mathsf{pk}_{\mathcal{U}} \text{ or } (\mathsf{id}, \pi), \Pi_G\right)$ is a deterministic algorithm that can be executed by any players. It outputs 1 if Π_G is correct and 0 otherwise.

Remark. *In the blockchain-based transferable e-cash systems, we assume* Π_G *is published on the blockchain.*

5.4.1 Oracles

Our security games use oracles. We adopt the oracle notions from [BFQ21, CG08].

Global Variables We store all information about users in the user list $\mathcal{UL}$ consisting of $\mathcal{U}_i = (\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{uds}_i)$ where uds_i indicates how many times user $\mathcal{U}_i$ has performed the double-spending attacks. The set of supplied coins by the oracles is denoted by $\mathcal{SC}$, and the set of all coins owned by the oracles is denoted by $\mathcal{OC}$. The set of deposited coins is denoted by $\mathcal{DC}$.

If an error occurs during the execution of the oracles, the oracles output $\perp$. Otherwise, the call of the oracles is assumed to have succeeded.

Registration and Corruption Users The adversary can instruct the creation of honest users or passively observe registration. It can moreover *spy* on users, meaning that the adversary can learn the user's secret key. Note that the *spy* can not be performed on the challenge users.

- `BRegist`() plays the bank side of key generation algorithm BKeyGen and interacts with the adversary. If successful, it adds $(\mathsf{pk}, \perp, \mathsf{uds} = 0)$ to $\mathcal{UL}$.

  Remark. *Since blockchain-based transferable e-cash systems have the blockchain* $\mathbb{C}$ *in place of the bank and* BKeyGen *outputs nothing, oracle* `BRegist`() *outputs nothing.*

- `URegist`() plays the user side of the key generation algorithm UKeyGen when the adversary controls the bank. Upon successful execution, it adds $(\mathsf{pk}, \mathsf{sk}, 0)$ to $\mathcal{UL}$.

  Remark. *In the blockchain-based transferable e-cash systems,* `URegist`() *plays when the issuer* $\mathcal{I}$ *of DAA-based anonymous signature schemes are controlled by the adversary. Upon successful execution, it adds* $(\perp, \mathsf{sk}, 0)$ *to* $\mathcal{UL}$.

- $\texttt{Regist}()$ plays both parties in the $\mathsf{BKeyGen}$ and $\mathsf{UKeyGen}$ algorithm and adds $(\mathsf{pk}, \mathsf{sk}, 0)$ to $\mathcal{UL}$.

  Remark. *Since blockchain-based transferable e-cash systems have the blockchain* $\mathbb{C}$ *in place of the bank and* $\mathsf{BKeyGen}$ *outputs nothing, the oracle* $\texttt{Regist}()$ *outputs nothing.*

- $\texttt{Spy}(i)$, for $i \leq |\mathcal{UL}|$, returns the user i's secret key sk_i.

Withdrawal Oracles The adversary can withdraw a coin from the bank or passively observe a withdrawal.

- $\texttt{BWith}(i)$ plays the bank side of $\mathsf{Withdraw}(\mathcal{B}, \mathcal{U}_i)$ algorithm. This oracle updates $\mathcal{SC}$ by adding $\mathcal{V}_{\mathcal{B}}^{\mathrm{W}}$ with bit 1 to flag it as corrupted coin.

  Remark. *Since blockchain-based transferable e-cash systems have the blockchain* $\mathbb{C}$ *in place of the bank, oracle* $\texttt{BWith}(i)$ *outputs nothing.*

- $\texttt{UWith}(i)$ plays the user side $\mathsf{Withdraw}(\mathcal{B}, \mathcal{U}_i)$ when the bank is controlled by the adversary. This oracle updates $\mathcal{OC}$ by adding the value $(\mathcal{U}_i, \mathsf{id}, \pi)$.

  Remark. *Since blockchain-based transferable e-cash systems have the blockchain* $\mathbb{C}$ *in place of the bank,* $\texttt{UWith}(i)$ *plays with* $\mathsf{Withdraw}(\perp, \mathcal{U}_i)$ *when the adversary controls the issuer. Then, it updates* $\mathcal{OC}$ *by adding the value* $(\mathcal{U}_i, \mathsf{id}, \pi)$.

- $\texttt{With}(i)$ simulates $\mathsf{Withdraw}$ protocol execution of both $\mathcal{B}$ and user $\mathcal{U}_i$, and it updates $\mathcal{OC}$ as for $\mathsf{Withdraw}(\mathcal{B}, \mathcal{U}_i)$ and $\mathcal{SC}$ by adding $\mathcal{V}_{\mathcal{B}}^{\mathrm{W}}$ with flag 0.

  Remark. *Since blockchain-based transferable e-cash systems have the blockchain* $\mathbb{C}$ *in place of the bank, oracle* $\texttt{With}(i)$ *outputs nothing.*

Spend and deposit oracles.

- $\mathtt{Spd}(j)$ plays the role of user $\mathcal{U}_j$ by spending a coin in $\mathcal{OC}$ owned by user $\mathcal{U}_j$. It uses and updates the entry $(\mathcal{U}_j, \mathsf{id}, \pi)$ of $\mathcal{OC}$ as the Spend protocol.

- $\mathtt{Rcv}(i)$ makes honest user i receive a coin from the adversary and updates the set of $\mathcal{OC}$ by adding a new entry $(\mathcal{U}_i, \mathsf{id}, \pi)$.

- $\mathtt{S\&R}(j, i)$ plays the role of both $\mathcal{U}_j$ and $\mathcal{U}_i$ and it executes the spending of a coin owned by $\mathcal{U}_j$ to user $\mathcal{U}_i$. It updates $\mathcal{OC}$ by adding the value $(\mathcal{U}_i, \mathsf{id}, \pi)$ and by flagging the coin as spent by $\mathcal{U}_j$.

- $\mathtt{BDepo}()$ lets $\mathcal{A}$ deposit a coin. It runs $\mathcal{U}_i$ in $\mathsf{Deposit}$ using the bank's secret key $\mathsf{sk}_{\mathcal{B}}$. If successful, it adds the received coin to $\mathcal{DC}$ or runs $\mathsf{Identify}$ and returns $(\mathsf{pk}, \Pi_G) \leftarrow \mathsf{Identify}(\mathsf{id}, \pi, \pi')$.

  Remark. *Since blockchain-based transferable e-cash systems have the blockchain* $\mathbb{C}$ *in place of the bank, oracle* $\mathtt{BDepo}()$ *outputs nothing.*

- $\mathtt{Depo}(j)$, the honest deposit oracle, runs $\mathsf{Deposit}$ between the j-th coin owner in $\mathcal{OC}$ and an honest bank. If successful, it adds the received coin to $\mathcal{DC}$ or runs $\mathsf{Identify}$ and returns $(\mathsf{pk}, \Pi_G) \leftarrow \mathsf{Identify}(\mathsf{id}, \pi, \pi')$.

  Remark. *Since blockchain-based transferable e-cash systems have the blockchain* $\mathbb{C}$ *in place of the bank,* $\mathtt{Depo}(i)$ *plays the role of the user* $\mathcal{U}_i$ *during a* $\mathsf{Deposit}$ *algorithm.*

5.4.2 Security Notions

In this section, we discuss the security properties. To achieve anonymity in Takahashi and Otsuka [TO21], we need to satisfy two major properties.

The first one is Security properties (*Economic properties* and *Security properties*). For achieving anonymity, we borrow the definition of Transferable E-Cash from Bauer et al. [BFQ21].

The second one is *Double-spending attacks Detection methods* and *Proportional Fee Scheme*. In addition to satisfying anonymity, we outline Double-spending attacks Detection methods, which are the core security design of [TO21], and organize the conditions that must be satisfied.

Economic properties ensure that users do not incur economic losses when they participate in the system. It consists of *soundness*, *unforgeability*, and *exculpability*.

Soundness

Suppose an honest user issues or accepts a ticket during a transfer; he should be guaranteed that others will accept the ticket. In that case, either honest users when transferring or the blockchain escrow account when claiming.

The game is formalized in Figure 5.1. The adversary issues a ticket or sends the received ticket τ to the user i_0. If the result of sending ticket the ticket to the user i_1 or claiming to the blockchain is false, the adversary wins.

Experiment: $\mathrm{Expt}^{\mathtt{sound}}_{\mathcal{A},\Pi}(\lambda)$

1 : $\mathsf{param} \leftarrow \mathsf{ParamGen}(1^\lambda)$; $\mathsf{pk}_{\mathcal{B}} \leftarrow \mathcal{A}(\mathsf{param})$
2 : $(b, i_1, i_2) \leftarrow \mathcal{A}^{\mathtt{URegist},\mathtt{Spy}}$
3 : **if** $b = 0$ then run $\mathtt{UWith}(i_1)$ with $\mathcal{A}$
4 : **else** run $\mathtt{Rcv}(i_1)$ with $\mathcal{A}$
5 : **return** 0 if this outputs $\perp$
6 : run $\mathtt{S\&R}(i_1, i_2)$;
7 : **if** one party outputs $\perp$
8 : **return** 1
9 : **else return** 0

Figure 5.1: Game for *soundness*

Definition 13. *(Soundness) An anonymous transferable scheme is* sound *if for all PPT adversaries $\mathcal{A}$, we have*

$$\mathsf{Adv}^{\mathtt{sound}}_{\mathcal{A},\Pi}(\lambda) = \Pr\left[\mathsf{Expt}^{\mathtt{sound}}_{\mathcal{A},\Pi}(\lambda) = 1\right] < \mathsf{negl}(\lambda). \tag{5.9}$$

Unforgeability

Unforgeability ensures that no user can spend more tickets than the number of tickets they received/issued. Unforgeability also guarantees that the adversary address is accused whenever a ticket is double-spending. The game is formalized in Figure 5.2.

Experiment: $\mathsf{Expt}^{\mathsf{Unforg}}_{\mathcal{A},\Pi}(\lambda)$

$\mathsf{param} \leftarrow \mathsf{ParamGen}(1^\lambda)$
$(\mathsf{sk}_\mathcal{B}, \mathsf{pk}_\mathcal{B}) \leftarrow \mathsf{BKeyGen}(\mathsf{param})$
$(\mathsf{id}, \pi, \mathsf{sk}_\mathcal{A}, \mathsf{pk}_\mathcal{A}) \leftarrow \mathcal{A}^{\mathtt{BRegist},\mathtt{BWith},\mathtt{BDepo}}(\mathsf{param}, \mathsf{pk}_\mathcal{B})$
return 1 if there exists $(\mathsf{id}, \pi') \in \mathcal{DC}$ with $\pi \neq \pi'$
and all of the following conditions hold:
  1) $\perp \leftarrow \mathsf{Identify}(\mathsf{id}, \pi, \pi')$
$\wedge$ 2) $(\mathsf{pk}, \Pi_G) \leftarrow \mathsf{Identify}(\mathsf{id}, \pi, \pi')$
    $\wedge 0 \leftarrow \mathsf{VerifyGuilt}(\mathsf{pk}, \Pi_G)$
$\wedge$ 3) $\mathsf{pk} \leftarrow \mathsf{Identify}(\mathsf{id}, \pi, \pi') \wedge \mathsf{pk} \notin \mathcal{UL}$
return 1 if $\mathsf{id} \notin \mathcal{SC}$
  $\mathsf{Deposit}\Big(\mathcal{U}\left(\mathsf{id}, \pi, \mathsf{sk}_\mathcal{A}, \mathsf{pk}_\mathcal{A}, \mathsf{pk}_\mathcal{B}\right), \mathcal{B}\left(\mathsf{sk}_\mathcal{B}, \mathsf{pk}_\mathcal{B}, \mathsf{pk}_\mathcal{A}, \mathcal{L}\right)\Big) = \mathsf{OK}$
else return 0

Figure 5.2: Game for *unforgeability*

Definition 14. *(Unforgeability) An anonymous transferable scheme is* unforgeable *if for all PPT adversaries $\mathcal{A}$, we have*

$$\mathsf{Adv}^{\mathsf{Unforg}}_{\mathcal{A},\Pi}(\lambda) = \Pr\left[\mathsf{Expt}^{\mathsf{Unforg}}_{\mathcal{A},\Pi}(\lambda) = 1\right] < \mathsf{negl}(\lambda). \tag{5.10}$$

Exculpability

Exculpability ensures that an honest user can not wrongly be accused of double-spending. Exculpability also guarantees that the adversary's address is accused whenever a ticket is double-spending. Specifically, it guarantees that the adversary can not produce double-spending coins that can output the secret key of a user who has not committed double-spending attacks. The game is formalized in Figure 5.3.

Experiment: $\mathrm{Expt}^{\mathtt{excul}}_{\mathcal{A},\Pi}(\lambda)$

1 : $\mathsf{param} \leftarrow \mathsf{ParamGen}(1^\lambda)$; $\mathsf{sk}_{\mathcal{B}} \leftarrow \mathcal{A}(\mathsf{param})$

2 : $(\mathsf{id}^*, \pi^*, \Pi^*_G) \leftarrow \mathcal{A}^{\mathtt{URegist},\mathtt{Spy},\mathtt{UWith},\mathtt{Rcv},\mathtt{S\&R},\mathtt{Depo}}(\mathsf{param})$

3 : **return** 1 if all of the following conditions hold:

1) $\mathsf{VerifyGuilt}(\mathsf{id}^*, \pi^*, \Pi^*_G) = 1$

Let i^* be the owner of (id^*, π^*)

$\wedge$ 2) there was no call $\mathtt{Spy}(i^*)$

$\wedge$ 3) $\mathsf{uds}_{i^*} = 0$

4 : **else return** 0

Figure 5.3: Game for *exculpability*

Definition 15. *An anonymous transferable scheme is* exculpable *if for all PPT adversaries* $\mathcal{A}$*, we have*

$$\mathsf{Adv}^{\mathtt{excul}}_{\mathcal{A},\Pi}(\lambda) = \Pr\left[\mathsf{Expt}^{\mathtt{excul}}_{\mathcal{A},\Pi}(\lambda) = 1\right] < \mathsf{negl}(\lambda). \tag{5.11}$$

5.4.3 Anonymity Notions

Anonymity notion for transferable electronic cash systems is first defined in Okamoto and Ohta [OO92]. Canard and Gouget [CG08] defined the four levels of anonymity for transferable electronic cash systems:

- *Weak anonymity* satisfies the property that any PPT adversary can not link the withdrawal and the deposit views. Still, the adversary may link two independent payments by the same user.

- *Strong anonymity* satisfies the weak anonymity, and the adversary can not link two payments by the same user. Still, the adversary may recognize the coin that he has previously observed.

- *Full anonymity* satisfies strong anonymity, and the adversary can not recognize any coin that is transferred between honest users. Still, the adversary may recognize the coin he has previously owned.

- *Perfect anonymity* satisfies full anonymity, and the adversary can not decide whether a coin is the one that he has previously owned or not.

According to the above anonymity notion, the scheme provided by Okamoto and Ohta [OO92] satisfies weak anonymity. Whereas, transferable electronic cash schemes proposed by Chaum et al. [CP93] and Canard et al. [CGT08] satisfy strong anonymity. Perfect anonymity is proved to be impossible [CG08]. Intuitively, this proof is conducted as follows: suppose that a PPT adversary received coins c_0 and c_1 after spending a coin c such that one of the coins is related to c. Given one of the coins c_b for $b \in \{0, 1\}$, the adversary can always distinguish whether c_b is related to c or not just by depositing c_0 together with c. $b = 0$ if the adversary is accused of double-spending, $b = 1$ otherwise. Thus, the best achievable anonymity notions are full anonymity and restricted variants of perfect anonymity. More recently, Bauer et al. [BFQ21] introduced new notions of user anonymity, coin anonymity and coin-transparency. As detailed later, all of these three notions are restricted variants of perfect anonymity by Canard and Gauget [CG08].

Coin Anonymity (c-an)

Definition 16. *(Coin anonymity) For all PPT adversaries $\mathcal{A}$, there exists $^{\exists}\lambda_0 \in \mathbb{N}$ such that for all security parameter $\lambda \geq \lambda_0$, the protocol of VeloCash, Π_{VC} satisfies **Coin anonymity** if the following holds:*

$$\begin{aligned} &\mathsf{Adv}^{\texttt{c-an}}_{\mathcal{A},\Pi}(\lambda) = \\ &\left|\Pr\left[\mathsf{Expt}^{\texttt{c-an}}_{\mathcal{A},\Pi,1}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\texttt{c-an}}_{\mathcal{A},\Pi,0}(\lambda) = 1\right]\right| < \mathsf{negl}(\lambda). \end{aligned} \tag{5.12}$$

User Anonymity (u-an)

We describe *User anonymity* game in Figure 5.5. The adversary issues or receives a ticket and sends it to one of two user groups. Then, the adversary receives the ticket again, which

Experiment: $\mathsf{Expt}^{\texttt{c-an}}_{\mathcal{A},\Pi,b}(\lambda)$

$\mathsf{param} \leftarrow \mathsf{ParamGen}(1^\lambda)$
$\mathsf{pk}_{\mathcal{B}} \leftarrow \mathcal{A}(\mathsf{param})$
$i_0^{(0)} \leftarrow \mathcal{A}^{\texttt{URegist},\texttt{Spy}}$; run $\texttt{UWith}(i_0^{(0)})$ with $\mathcal{A}$
$i_0^{(1)} \leftarrow \mathcal{A}^{\texttt{URegist},\texttt{Spy}}$; run $\texttt{UWith}(i_0^{(1)})$ with $\mathcal{A}$
$\left((i_1^{(0)}, \cdots, i_{k_0}^{(0)}), (i_1^{(1)}, \cdots, i_{k_1}^{(1)})\right)$
  $\leftarrow \mathcal{A}^{\texttt{URegist},\texttt{Spy}}$
if $k_0 \neq k_1$ then **return** 0
for $j = 1, \cdots, k_0$:
  run $\texttt{S\&R}(i_{j-1}^{(0)}, i_j^{(0)})$
  run $\texttt{S\&R}(i_{j-1}^{(1)}, i_j^{(1)})$
run $\texttt{Spd}(i_j^{(b)})$ with $\mathcal{A}$
run $\texttt{Spd}(i_j^{(1-b)})$ with $\mathcal{A}$
$b^* \leftarrow \mathcal{A}$; return b^*

Figure 5.4: Game for *coin anonymity*

has been passed through between the users, and determines which of two user groups it has passed through.

Definition 17. *(User anonymity) We define User anonymity if the following properties satisfy: For all PPT adversaries $\mathcal{A}$, there exists $^\exists\lambda_0 \in \mathbb{N}$ such that for all security parameter $\lambda \geq \lambda_0$,*

$$\begin{aligned} &\mathsf{Adv}^{\texttt{u-an}}_{\mathcal{A},\Pi}(\lambda) = \\ &\Pr\left[\mathsf{Expt}^{\texttt{u-an}}_{\mathcal{A},\Pi,1}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\texttt{u-an}}_{\mathcal{A},\Pi,0}(\lambda) = 1\right] < \mathsf{negl}(\lambda). \end{aligned} \tag{5.13}$$

Coin Transparency (c-tr)

The experiment is specified in Figure 5.6. We assume $\mathsf{Expt}^{\texttt{c-tr}}_{\mathcal{A},\Pi,b}(\lambda)$ aborts when challenge coins (c_0, c_1) are double-spent. For more details, see the original definition in [BFQ21]. This is a strong anonymity notion, meaning that the user who has transferred a coin will not be able to recognize it if she receives it again. The adversary issues or receives two tickets and sends them to the two user groups, respectively. Then, the adversary receives one of the

Experiment: $\mathrm{Expt}^{\mathtt{u\text{-}an}}_{\mathcal{A},\Pi,b}(\lambda)$

$\mathsf{param} \leftarrow \mathsf{ParamGen}(1^\lambda)$
$\mathsf{pk}_{\mathcal{B}} \leftarrow \mathcal{A}(\mathsf{param})$
$(i_0^{(0)}, i_0^{(1)}) \leftarrow \mathcal{A}^{\mathtt{URegist},\mathtt{Spy}}$
run $\mathtt{Rcv}(i_0^{(b)})$ with $\mathcal{A}$
$\left((i_1^{(0)}, \cdots, i_{k_0}^{(0)}), (i_1^{(1)}, \cdots, i_{k_1}^{(1)})\right)$
$\quad \leftarrow \mathcal{A}^{\mathtt{URegist},\mathtt{Spy}}$
if $k_0 \neq k_1$ then **return** 0
for $j = 1, \cdots, k_0$:
$\quad$ run $\mathtt{S\&R}(i_{j-1}^{(b)}, i_j^{(b)})$
run $\mathtt{Spd}(i_j^{(b)})$ with $\mathcal{A}$
$b^* \leftarrow \mathcal{A}^{\mathtt{BDepo}}$; **return** b^*

Figure 5.5: Game for *user anonymity*

tickets and determines which group the ticket has passed through.

Definition 18. *(Coin transparency) We define Coin transparency if the following properties satisfy: For all PPT adversaries $\mathcal{A}$,*

$$\begin{aligned}&\mathsf{Adv}^{\mathtt{c\text{-}tr}}_{\mathcal{A},\Pi}(\lambda) = \\ &\left|\Pr\left[\mathsf{Expt}^{\mathtt{c\text{-}tr}}_{\mathcal{A},\Pi,1}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathtt{c\text{-}tr}}_{\mathcal{A},\Pi,0}(\lambda) = 1\right]\right| < \mathsf{negl}(\lambda).\end{aligned} \tag{5.14}$$

Definition 19. *(Anonymity) For $\mathsf{x} \in \{\mathtt{c\text{-}an}, \mathtt{u\text{-}an}, \mathtt{c\text{-}tr}\}$, an anonymous transferable scheme satisfies x if it satisfies the following equations: For all PPT adversaries $\mathcal{A}$,*

$$\begin{aligned}&\mathsf{Adv}^{\mathsf{x}}_{\mathcal{A},\Pi}(\lambda) = \\ &\left|\Pr\left[\mathsf{Expt}^{\mathsf{x}}_{\mathcal{A},\Pi,1}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{x}}_{\mathcal{A},\Pi,0}(\lambda) = 1\right]\right| < \mathsf{negl}(\lambda).\end{aligned} \tag{5.15}$$

Note that in this chapter, we assume that the winning amount β is equal for all escrow accounts ϵ, because if the winning amount β is very high, the winning ticket may not satisfy the anonymity notions. Consider the case where a ticket returns to the same sender again, and the ticket is won. The user can identify the ticket from the amount spent and the

Experiment: $\mathrm{Expt}^{\texttt{c-tr}}_{\mathcal{A},\Pi,b}(\lambda)$

$\mathsf{param} \leftarrow \mathsf{ParamGen}(1^\lambda)$
$((\mathsf{sk}_\mathcal{W}, \mathsf{sk}_\mathcal{D}, \mathsf{sk}_{\mathcal{CK}}), \mathsf{pk}_\mathcal{B}) \leftarrow \mathsf{BKeyGen}(\mathsf{param})$
$i^{(0)} \leftarrow \mathcal{A}^{\texttt{URegist},\texttt{BDepo},\texttt{Spy}}(\mathsf{param}, \mathsf{pk}_\mathcal{B}, \mathsf{sk}_\mathcal{W}, \mathsf{sk}_\mathcal{D})$
$i^{(1)} \leftarrow \mathcal{A}^{\texttt{URegist},\texttt{BDepo},\texttt{Spy}}(\mathsf{param}, \mathsf{pk}_\mathcal{B}, \mathsf{sk}_\mathcal{W}, \mathsf{sk}_\mathcal{D})$
Run $\texttt{Rcv}(i^{(0)})$ with $\mathcal{A}$
Run $\texttt{Rcv}(i^{(1)})$ with $\mathcal{A}$
$\left((i_1^{(0)}, \cdots, i_{k_0}^{(0)}), (i_1^{(1)}, \cdots, i_{k_1}^{(1)})\right)$
$\quad \leftarrow \mathcal{A}^{\texttt{URegist},\texttt{Spy}}$
if $k_0 \neq k_1$ then **return** 0
for $j = 1, \cdots, k_0$:
$\quad$ run $\texttt{S\&R}(i_{j-1}^{(b)}, i_j^{(b)})$
run $\texttt{Spd}(i_j^{(b)})$ with $\mathcal{A}$
$b^* \leftarrow \mathcal{A}$; **return** b^*

Figure 5.6: Game for *Coin transparency*

number of times the ticket has been transferred.

This is a trivial assumption and is a setting to simplify the conditions for achieving anonymity. In Proportional Fee Scheme, there is a concern that the high winning amount causes the speed of payment to be slowed down since the difference between the value of the ticket and the winning amount is large. For example, if the winning amount is $\$10,000$ and the ticket value is 1 cent, it is not surprising that people would wait for the winning result before using it to pay. For Proportional Fee Scheme to work practically, it is ideal if the winning amount is not very high, i.e., $\beta - \gamma$ is negligibly small, where γ is the blockchain transaction fee.

5.5 Attested Execution Secure Processors (AESP)

Tamper-proof is a property of a secure processor that prevents the leakage of sensitive information such as cryptographic keys and other confidential information against non-invasive attacks such as side-channel attacks and invasive attacks such as reverse engineering. There

is no known theoretical implementation method. In many practical cases, the hardware is referred to as "tamper-proof" that has passed exhaustive penetration tests such as FIPS 140 and AVA_VAN.5 in CC certification (ISO/IEC 15408).

Several methods have been proposed in academic and commercial literature to achieve tamper-proof secure processors, As a *de facto* standard, it is often referred to as Trusted Execution Environment (TEE). TEE allows any program to be executed in secure processors. Furthermore, TEE also has the "attested execution" function of verifying that the program's output in the device is from a legitimate tamper-proof device by signing the output with the signing key unique to the tamper-proof device.

On initialize:

// Interact with issuer $\mathcal{I}$

1: $\langle \perp, \mathsf{sk} \rangle := \Sigma.\mathsf{Join}_{\mathcal{I},\mathcal{G}_{\mathrm{att}}}\langle (\mathsf{gpk}, \mathsf{isk}), \mathsf{gpk} \rangle,\ T = \emptyset$
2: initialize RT

On receive: `getpk`() from some $\mathcal{P}$

1: send gpk to $\mathcal{P}$

* * * * * Enclave operations:

On receive: `install`(idx, prog) from some $\mathcal{P} \in \mathsf{reg}$:

1: if $\mathcal{P}$ is honest, assert $idx = sid$
2: $eid \leftarrow \{0,1\}^{\lambda}$
3: $T[eid, \mathcal{P}] := (idx, \mathsf{prog}, \mathbf{0})$
4: send eid to $\mathcal{P}$

On receive: `resume`(eid, inp) from some $\mathcal{P} \in \mathsf{reg}$:

1: let $(idx, \mathsf{prog}, \mathsf{mem}) := T[eid, \mathcal{P}]$
2: abort if $(eid, \mathcal{P}) \notin T$
3: $(\mathsf{outp}, \mathsf{mem}) := \mathsf{prog}(\mathsf{inp}, \mathsf{mem})$
4: $T[eid, \mathcal{P}] := (idx, \mathsf{prog}, \mathsf{mem})$
5: $\sigma := \Sigma.\mathsf{Sign}_{\mathsf{sk}}(idx, eid, \mathsf{prog}, \mathsf{outp})$
6: send (outp, σ) to $\mathcal{P}$

* * * * * Internal operations:

On call: `commitment`(rid) from some $(eid, \cdot) \in T$:

1: $\mathsf{r} \leftarrow \mathsf{RT}[rid]$
2: $(x, \mathtt{com}) \leftarrow \Sigma.\mathsf{Com}_{\mathsf{sk}}(\mathsf{r})$
3: return $(x, \mathtt{com})$

On call: `sign`$(m; rid)$ from some $(eid, \cdot) \in T$:

1: $(idx, \mathsf{prog}, \mathsf{mem}) \leftarrow T[eid]$
2: $\mathsf{r} \leftarrow \mathsf{RT}[rid]$
3: $\sigma := \Sigma.\mathsf{Sign}_{\mathsf{sk}}(idx, eid, \mathsf{prog}, m; \mathsf{r})$
4: return $(idx, eid, \mathsf{prog}, m, \sigma)$

Figure 5.7: The algorithms of $\mathcal{G}_{\mathrm{att}}[\Sigma, \mathsf{reg}, \mathsf{gpk}, \mathsf{AE}]$

On input: `initialize()` :
1: $\mathsf{unspentCoin} := \emptyset$
2: $\mathsf{txChain} := \emptyset$
3: $\mathsf{Key} := \emptyset$
4: $\mathsf{A} := \emptyset$

On input: `set escrow()` :
1: $rid_\epsilon \leftarrow \mathbb{Z}$
2: $(x, \mathtt{com}) := \mathcal{G}_{\mathrm{att}}.\mathtt{commitment}(rid_\epsilon)$
3: $\epsilon := \mathsf{Hash}(x, \mathtt{com})$
4: store $\mathsf{unspendCoin}[\epsilon] := rid_\epsilon$
5: return ϵ

On input: `init keyex`(sid, g) :
1: $a \leftarrow \mathbb{Z}_p$
2: $\mathsf{A}[sid] := a$
3: store A
4: return g^a

On input: `get addr`(sid, g^a, σ) :
1: accept σ if
2: $\quad \mathsf{Verify}(\mathsf{gpk}, \mathsf{Priv\text{-}RL}, \mathsf{Sig\text{-}RL}, g^a, \widehat{\sigma}) = 1$
3: $\quad$ // assume up to date $\mathsf{Priv\text{-}RL}, \mathsf{Sig\text{-}RL}$ is stored in $\mathsf{prog_w}$
4: $b \in \mathbb{Z}_p, \mathsf{K} := (g^a)^b$
5: store $\mathsf{Key}[sid] := \mathsf{K}$
6: $rid \leftarrow \mathbb{Z}$
7: $(x, \mathtt{com}) := \mathcal{G}_{\mathrm{att}}.\mathtt{commitment}(rid)$
8: $\mathsf{addr} := \mathsf{Hash}(x, \mathtt{com})$
9: $\widehat{\mathsf{addr}} := \mathsf{AE.Enc_K}(\mathsf{addr})$
10: store $\mathsf{unspentCoin}[\widehat{\mathsf{addr}}] := rid$
11: return $\widehat{\mathsf{addr}}, (g^a, g^b)$

On input: `make payment`$(\widehat{\mathsf{addr}}_X, (sid, eid, \mathsf{prog_w}, \widehat{\mathsf{addr}}_Y, (g^a, g^b), \sigma_Y))$:
// Payment from addr_X to addr_Y
1: accept if
2: $\quad \mathsf{Verify}(\mathsf{gpk}, \mathsf{Priv\text{-}RL}, \mathsf{Sig\text{-}RL}, (idx, eid, \mathsf{prog_w}, \widehat{\mathsf{addr}}_Y, (g^a, g^b)), \sigma_Y) = 1$
3: restore a
4: store $\mathsf{K} := (g^a)^b$
5: $\mathsf{addr}_Y := \mathsf{AE.Dec_K}(\widehat{\mathsf{addr}}_Y)$
6: if $\widehat{\mathsf{addr}}_X \in \mathsf{txChain}$
7: $\quad rid_X := \mathsf{unspentCoin}[\widehat{\mathsf{addr}}_X]$
8: $\quad (x, \mathtt{com}) := \mathcal{G}_{\mathrm{att}}.\mathtt{commitment}(rid_X)$
9: $\quad \mathsf{addr}_X := \mathsf{Hash}(x, \mathtt{com})$
10: $\quad$ restore $\tau := \mathsf{txChain}[\widehat{\mathsf{addr}}_X]$
11: $\quad\quad$ s.t. $\tau = \epsilon \prec \tau_1 \prec \cdots \prec \tau_n$
12: $\quad\quad$ where $\tau_n = (\mathsf{addr}_Z \rightarrow \mathsf{addr}_X, \mathsf{Hash}(\tau_{n-1}), \sigma_Z)$ is a valid transaction to X from some Z
13: $\quad \tau_{n+1} := (\mathsf{addr}_X \rightarrow \mathsf{addr}_Y, \mathsf{Hash}(\tau_n))$
14: $\quad (sid, eid, \mathsf{prog_w}, \tau_{n+1}, \sigma_X) := \mathcal{G}_{\mathrm{att}}.\mathtt{sign}(\tau_{n+1}; rid_X)$
15: $\quad$ delete $\mathsf{unspentCoin}[\widehat{\mathsf{addr}}_X]$
16: $\quad \widehat{\tau} := \mathsf{AE.Enc_K}(sid, eid, \mathsf{prog_w}, (\epsilon \prec \tau_1 \prec \cdots \prec \tau_{n+1}), \sigma_X)$
17: if $\widehat{\mathsf{addr}}_X \notin \mathsf{txChain} \wedge \widehat{\mathsf{addr}}_X \in \mathsf{unspentCoin}$
18: $\quad$ // this must be the case $\widehat{\mathsf{addr}}_X = \epsilon$
19: $\quad rid_\epsilon \leftarrow \mathsf{unspentCoin}[\widehat{\mathsf{addr}}_X]$
20: $\quad (x, \mathtt{com}) := \mathcal{G}_{\mathrm{att}}.\mathtt{commitment}(rid_\epsilon)$
21: $\quad \epsilon := \mathsf{Hash}(x, \mathtt{com})$
22: $\quad \tau_1 := (\epsilon \rightarrow \mathsf{addr}_Y, \mathsf{Hash}(\epsilon))$
23: $\quad (sid, eid, \mathsf{prog_w}, (\epsilon \prec \tau_1), \sigma_X) := \mathcal{G}_{\mathrm{att}}.\mathtt{sign}((\epsilon \prec \tau_1); rid_\epsilon)$
24: $\quad$ delete $\mathsf{unspentCoin}[\widehat{\mathsf{addr}}_X]$
25: $\quad\quad \widehat{\tau} := \mathsf{AE.Enc_K}(sid, eid, \mathsf{prog_w}, (\tau \prec \tau_1), \sigma_X)$
26: $\quad$ abort if $\epsilon \notin \mathsf{unspentCoin}$
27: return $\widehat{\tau}$

On input: `receive payment`$(sid, eid, \mathsf{prog_w}, \widehat{\tau}, \widehat{\sigma}, \widehat{\mathsf{addr}})$:
1: restore $\mathsf{K} := \mathsf{Key}[sid]$
2: $(sid, eid, \mathsf{prog_w}, \tau, \sigma) := \mathsf{AE.Dec_K}(\widehat{\tau})$
//assume $\tau = \{\epsilon \prec \cdots \tau_n\}$ for some $n \geq 1$
3: accept if the outer and inner attested signatures are valid
// assume up to date $\mathsf{Priv\text{-}RL}, \mathsf{Sig\text{-}RL}$ and $\mathbb{C}$ are stored in $\mathsf{prog_w}$
4: $\quad \mathsf{Verify}(\mathsf{gpk}, \mathsf{Priv\text{-}RL}, \mathsf{Sig\text{-}RL},$
5: $\quad\quad\quad (sid, eid, \mathsf{prog_w}, \widehat{\tau}), \widehat{\sigma}) = 1$
6: $\quad\quad$ // for the attested signature on the encrypted ticket
7: $\quad \mathsf{Verify}(\mathsf{gpk}, \mathsf{Priv\text{-}RL}, \mathsf{Sig\text{-}RL},$
8: $\quad\quad (sid, eid, \mathsf{prog_w}, \tau, \sigma) = 1$
9: $\quad\quad$ // for the attested signature on the plaintext ticket
10: $\quad \epsilon \in \mathbb{C}^{\lceil k}$ for some constant k, and not spent
11: store $\mathsf{txChain}[\widehat{\mathsf{addr}}] = \tau$
12: return `status`$(\in \{0, 1\})$

On input: `check winning`$(\mathsf{pk}_Z, \widehat{\mathsf{addr}})$:
1: if $\widehat{\mathsf{addr}} \in \mathsf{txChain}$
2: $\quad rid := \mathsf{unspentCoin}[\widehat{\mathsf{addr}}]$
3: $\quad (x, \mathtt{com}) := \mathcal{G}_{\mathrm{att}}.\mathtt{commitment}(rid)$
4: $\quad \mathsf{addr} := \mathsf{Hash}(x, \mathtt{com})$
5: $\quad$ restore $\tau_n := \mathsf{txChain}[\widehat{\mathsf{addr}}]$
6: if $\tau_n \in \mathsf{win}$
7: $\quad$ // let τ_n has the form :
8: $\quad$ // $\tau_n = (A \rightarrow B, \mathsf{Hash}(\tau_{\mathrm{pre}}), \sigma)$
9: $\quad \tau_{n+1} := (B \rightarrow \mathsf{pk}_Z, \mathsf{Hash}(\tau_{\mathrm{pre}}))$
10: return τ_{n+1}
11: if ϵ is already spent on $\mathbb{C}$, then Fork is detected
12: if $\mathsf{txChain}$ contains $\widehat{\mathsf{addr}}_Z$ where $\tau_n = (\widehat{\mathsf{addr}}_Z \rightarrow \widehat{\mathsf{addr}}_r, \mathsf{Hash}(\tau_{\mathrm{pre}}), \sigma_Z)$ then Collision is detected
13: In both cases, there exists τ'
14: $\quad \mathsf{sk} := \mathsf{KeyExtractor}(\tau, \tau')$
15: $\quad\quad$ return sk
16: otherwise return $\perp$

Figure 5.8: The algorithms of $\mathsf{prog_w}$

Pass et al. have proposed formal abstractions for attested execution secure processors as *Attested Execution Secure Processors (AESP)* [PST17].

The structure of AESP is shown in Figure 5.7, 5.8. Pass et al. have proposed an ideal function $\mathcal{G}_{\text{att}}$ that abstractly captures the essence of the wide range of attested execution processors. The most naive abstraction of $\mathcal{G}_{\text{att}}$ has a public key and a secret key pair $(\mathsf{gpk}, \mathsf{msk})$ for signing embedded by a manufacturer. By sharing the same secret keys among all $\mathcal{G}_{\text{att}}$, no one can distinguish the issuer of the attested signatures.

The signature on message m with the signing key is denoted as $\Sigma.\mathsf{Sign}_{\mathsf{sk}}(idx, eid, \mathsf{prog}, m; \mathsf{r})$.

$\mathcal{G}_{\text{att}}$ has four interfaces as follows:

1. Initialization:

   Generates the key pair $(\mathsf{mpk}, \mathsf{msk})$ to be used for attested execution and initializes the internal memory T.

2. Obtaining the public key:

   Outputs the public key mpk to be used for signature verification of attested execution.

3. Registration of program:

   Register a program prog in $\mathcal{G}_{\text{att}}$ and allocate unique memory space mem for the program. Enclave instance is a pair of $(idx, \mathsf{prog}, \mathsf{mem})$ for idx, which is the session ID when prog is registered. eid is the identifier of the enclave instance and is recorded in T for each platform $\mathcal{P}$.

4. Execution of program:

   Input inp to the prog specified by eid and $\mathcal{P}$, and output outp. Then, sign outp with msk and output the signature σ for the attestation.

When $\mathcal{G}_{\text{att}}$ is executed, the output outp is always signed with the embedded signing key msk. Based on the output and the signature, a verifier can verify that the output is sent from a secure processor $\mathcal{G}_{\text{att}}$.

From the above, if $\mathcal{G}_{\text{att}}$ exists, we can achieve the followings:

1. Any arbitrary programs can be installed in $\mathcal{G}_{\text{att}}$.

2. The installed programs are obfuscated; thus, the adversary can not obtain any information more than input and output.

3. Secure channel can be achieved with the installed programs by Diffie–Hellman key exchange protocol, etc.

4. Outputs from the installed programs are signed by $\mathcal{G}_{\text{att}}$; thus, a verifier can verify that the output is sent from $\mathcal{G}_{\text{att}}$.

The signatures on outputs from the internal program by $\mathcal{G}_{\text{att}}$ should be anonymous. Since even though the programs exchange encrypted data through a secure channel, anonymity is lost if the signer's identity is known. Anonymous cryptographic technology that can not determine from which device the signature came is called "Anonymous Attestation".

Realizing anonymous attestation is achieved by using a digital signature scheme called *Direct Anonymous Attestation (DAA)*, which is similar to group signatures. DAA differs from group signatures in that it has strong anonymity, in that even the group manager can not identify which key was used to create the signature.

5.5.1 Extension of AESP

We propose the following extension to the original AESP to revoke malicious platforms. We assume that AESP utilizes any direct anonymous attestation (DAA) schemes with Fiat-Shamir proof of knowledge for NP relation. The idea is the following: When a platform $\mathcal{P}$ receives a transaction, we force $\mathcal{P}$ to commit to a set of one-time randomness $(x, \texttt{com})$ as the destination address of the transaction. Then, at the time when $\mathcal{P}$ transfers the transaction to other platforms, $\mathcal{P}$ must issue an anonymous attested signature committing to the randomness $(x, \texttt{com})$. As a result, the double-spending platform must issue two different signatures using the same randomness $(x, \texttt{com})$ so that the witness will reveal and registered in the revocation list.

In the following extension, we introduce Randomness Tape RT inside $\mathcal{G}_{\mathrm{att}}$ and give installed programs to specify randomness index rid_i.

Randomness Tape

A randomness tape RT is a list of random numbers with each entry having large enough entropy.

$$\mathsf{RT} = \{rid_0, rid_1, \cdots, rid_n\} \tag{5.16}$$

When $\mathsf{prog}_{\mathsf{w}}$ calls AESP's `commitment`, $\mathcal{G}_{\mathrm{att}}$ returns $(x, \mathtt{com})$. Next, when $\mathsf{prog}_{\mathsf{w}}$ calls `sign` with the same rid on `commitment`, $\mathcal{G}_{\mathrm{att}}$ returns the signature σ from the random tape value corresponding to the eid.

Internal signature

Normally, $\mathcal{G}_{\mathrm{att}}$'s EPID signature is applied to the output outp of $\mathsf{prog}_{\mathsf{w}}$ when $\mathsf{resume}(eid, \mathsf{inp})$ is called. When the ticket is sent, the ticket is encrypted with the shared key of Diffie–Hellman key exchange, and the winning ticket is registered in the blockchain, including the EPID signature on each transmission. If the signature is performed on the ciphertext, anonymity can not be satisfied because the winner will post the ciphertext and the signature on the blockchain.

We extend $\mathcal{G}_{\mathrm{att}}$ to provide internal operations that allow $\mathsf{prog}_{\mathsf{w}}$ to request EPID signing on an arbitrary message to $\mathcal{G}_{\mathrm{att}}$. The $\mathsf{prog}_{\mathsf{w}}$ requests $\mathcal{G}_{\mathrm{att}}$ to $\mathtt{sign}(m; rid)$ to obtain an EPID signature on m.

First, $\mathsf{prog}_{\mathsf{w}}$ creates a plain-text $(\mathsf{addr}_X \rightarrow \mathsf{addr}_Y)$ indicating the transfer from X to Y, then obtains the signature as follows:

$$\sigma := \Sigma.\mathsf{Sign}_{\mathsf{sk}}\left(idx, eid, \mathsf{prog}_{\mathsf{w}}, \left(idx, eid, \mathsf{prog}_{\mathsf{w}}, (\mathsf{addr}_X \rightarrow \mathsf{addr}_Y, \mathsf{Hash}(\tau_{\mathrm{pre}}))\right); \mathsf{r}\right). \tag{5.17}$$

Then, $\mathsf{prog}_{\mathsf{w}}$ encrypts the above plain-text and the signature, and $\mathcal{G}_{\mathrm{att}}$ applies an EPID

signature on it and sends the following items to the payee's AESP.

$$\begin{cases} \widehat{\tau} = \mathsf{AE.Enc_K}\Big(idx, eid, \mathsf{prog_w}, (\mathsf{addr}_X \to \mathsf{addr}_Y, \mathsf{Hash}(\tau_{\mathrm{pre}})), \sigma\Big) \\ \widehat{\sigma} = \sum.\mathsf{Sign_{sk}}(idx, eid, \mathsf{prog_w}, \widehat{\tau}). \end{cases} \tag{5.18}$$

The plain text and the EPID signature on it that will eventually appear on the blockchain will be encrypted between AESPs so that nobody can track the history of transmission from outside the AESP.

Common id values

In the original definition of AESP, at the end of the process of `resume`, a signature is performed that makes it verifiable that the output is from AESP. The signature input value consists of $(idx, eid, \mathsf{prog}, \mathsf{outp})$.

In VeloCash, we make the input values idx and eid be specified in a hash of the prog.

When the ticket is sent, idx and eid can be seen in rich environments. In the payment protocol described later, when a ticket is won, all transaction information, including past transactions, is registered in the blockchain, including the signatures. Since the winning ticket is registered in the blockchain, including idx and eid, it does not satisfy the coin transparency of the anonymity notion. By fixing idx and eid as the hash values of prog, it makes it impossible to identify from which AESP the ticket was sent.

Fixing eid means single instantiation of prog. For prog to send and receive tickets, it only needs to be able to store previously received tickets and information associated with the tickets. Since the memory corresponding to prog can store the states, single instantiation does not affect sending and receiving.

5.6 Key Extractor and Revocation

In VeloCash, we would like to have a mechanism to revoke an adversary if he performs a double-spending attack. For revocation, we use the EPID revocation function.

In this paper, we do not use the EPID's signature based revocation. From the signature based revocation, the value to be registered in the revocation list is $\big(B, K(= B^f)\big)$ where f is the secret key. In VeloCash, the signer takes a different value for B for each signature. Therefore, the signature based revocation list can not revoke the adversary.

Instead, we adopt the secret key based revocation. Once the adversary performs a double-spending attack, we make the secret key f extractable and put the f into the secret key revocation list.

We define *Key Extractor*, which is an extractor for extracting secret keys in non-interactive zero-knowledge proof using Fiat-Shamir heuristic [FS87, PS96]. To construct the definition, we borrow the notion from Fischlin [Fis05].

Definition 20. *(Key Extractor) Suppose a Fiat-Shamir proof of knowledge for NP relation R_λ is a pair of probabilistic polynomial-time algorithms (P, V) with special soundness [Fis05]. Then, there exists a probabilistic polynomial-time algorithm* KeyExtractor *which holds $(x, \omega') \in R_\lambda$, for any security parameter λ satisfies the following:*

$$\begin{cases} {}^\forall(x,\omega) \in R_\lambda \text{ and} \\ {}^\forall(x, \mathtt{com}, \mathtt{ch}, \mathtt{resp}), (x, \mathtt{com}, \mathtt{ch}', \mathtt{resp}') \\ \quad \text{s.t. } \mathsf{V}(x, \mathtt{com}, \mathtt{ch}, \mathtt{resp}) \\ \qquad = \mathsf{V}(x, \mathtt{com}, \mathtt{ch}', \mathtt{resp}') = 1 \\ \quad \textit{with } \mathtt{ch} \neq \mathtt{ch}' \\ \omega' \leftarrow \mathsf{KeyExtractor}((x, \mathtt{com}, \mathtt{ch}, \mathtt{resp}), \\ \qquad\qquad (x, \mathtt{com}, \mathtt{ch}', \mathtt{resp}')) \end{cases} \tag{5.19}$$

where com *is a commitment,* ch *is a challenge, and* resp *is a response.*

Lemma 1. *EPID signature has the Key Extractor defined in Definition 20.*

Proof. EPID signature has a form $\sigma = (B, K, T, c, s_x, s_f,\ s_a, s_b)$. Set the corresponding parameters in the Definition 20 as

$$\begin{aligned}&(x, \mathtt{com}, \mathtt{ch}, \mathtt{resp})\\ &= (\{B, K, T\}, \{R_1, R_2\}, c, \{s_x, s_f, s_a, s_b\})\end{aligned} \tag{5.20}$$

where

$$\begin{aligned}R_1 &= B^{r_f}\\ R_2 &:= e\left(T, g_2\right)^{-r_x} \cdot e\left(h_1, g_2\right)^{r_f}\\ &\quad \cdot e\left(h_2, g_2\right)^{r_b} \cdot e\left(h_2, \omega\right)^{r_a}\end{aligned} \tag{5.21}$$

and r_x, r_f, r_a, r_b are random parameters. Then, given two valid signatures

$$\begin{aligned}\sigma &= (B, K, T, c, s_x, s_f, s_a, s_b),\\ \sigma' &= (B, K, T, c', s'_x, s'_f, s'_a, s'_b)\end{aligned} \tag{5.22}$$

on the same $x = \{B, K, T\}$ and the commitment $\mathtt{com} = \{R_1, R_2\}$ where

$$\begin{aligned}R_1 &= B^{r_f} = B^{s_f} K^{-c} = B^{s'_f} K^{-c'}\\ R_2 &= e\left(T, g_2\right)^{-r_x} \cdot e\left(h_1, g_2\right)^{r_f} \cdot e\left(h_2, g_2\right)^{r_b} \cdot e\left(h_2, \omega\right)^{r_a}\\ &= e\left(T, g_2\right)^{-s_x} \cdot e\left(h_1, g_2\right)^{s_f} \cdot e\left(h_2, g_2\right)^{s_b} \cdot e\left(h_2, \omega\right)^{s_a}\\ &\quad \cdot \left(e\left(g_1, g_2\right) / e(T, \omega)\right)^{c}\\ &= e\left(T, g_2\right)^{-s'_x} \cdot e\left(h_1, g_2\right)^{s'_f} \cdot e\left(h_2, g_2\right)^{s'_b} \cdot e\left(h_2, \omega\right)^{s'_a}\\ &\quad \cdot \left(e\left(g_1, g_2\right) / e(T, \omega)\right)^{c'}\end{aligned} \tag{5.23}$$

with $c \neq c'$, which realizes the NIZKP of the witness (x, f, a, b) by the Fiat-Shamir proof of

knowledge for NP relation R_λ as follows:

$$\begin{cases} s_x = r_x + c \cdot x, & s'_x = r_x + c' \cdot x \\ s_f = r_f + c \cdot f, & s'_f = r_f + c' \cdot f \\ s_a = r_a + c \cdot a, & s'_a = r_a + c' \cdot a \\ s_b = r_b + c \cdot b, & s'_b = r_b + c' \cdot b \end{cases} \tag{5.24}$$

By solving the above equations, the witness (x, f, a, b) is extracted as follows:

$$\begin{aligned} &\mathsf{KeyExtractor}(\sigma, \sigma') \to (x, f, a, b) \\ &= \left(\frac{s_x - s'_x}{c - c'}, \frac{s_f - s'_f}{c - c'}, \frac{s_a - s'_a}{c - c'}, \frac{s_b - s'_b}{c - c'} \right). \end{aligned} \tag{5.25}$$

□

Further, we construct our scheme based on the EPID signature as a representative of DAA signature schemes. However, the only DAA schemes that have the KeyExtractor are applicable to our schemes.

5.7 Construction

This section introduces the protocol Π_{VC} of VeloCash. The general framework of the protocol is the same as Takahashi and Otsuka [TO21]; however, it includes a mechanism to anonymise the sending and receiving of tickets.

The protocol Π_{VC} consists of three phases: 1) Escrow Setup, 2) Payment with Lottery Ticket, and 3) Ticket Winning and Publication.

Setting The EPID's secret key based revocation list Priv-RL and signature based revocation list Sig-RL referenced among all user's AEPS shall be distributed in a blockchain and globally referenceable.

In the construction section, we have omitted the verification method of blockchain data performed by AESP. In each phase, AESP must efficiently verify that the escrow is actually registered in the blockchain and that there are no winners already by referring to the blockchain data provided by the wallet owner.

To achieve the verification, we can use the notion of *Proof of Publication* for PoW blockchains from [DNY17, CZK+19], which allows AESP to verify that a blockchain transaction is valid from the blockchain fragment received from the wallet owner.

In summary, it is an extension of standard transaction confirmation of the Proof-of-Work blockchain. More specifically, an AESP receives n blocks $n\text{-}T = \{B_i, \cdots, B_{i+n}\}$ and evaluates whether the time to produce the blocks is less than the specified security parameter, as follows:

$$|t_i - t_{i+n}| \leq n \cdot \delta \tag{5.26}$$

where t_i and t_{i+n} are time stamps extracted from blocks B_i and B_{i+n} respectively, and δ is a security parameter.

$\mathsf{ParamGen}(1^\lambda)$ In our construction, we assume EPID as a DAA signature scheme with Fiat-Shamir proof of knowledge for NP relation. EPID consists of the five algorithms $\{\mathsf{Setup}, \mathsf{Join}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Revoke}\}$ as described in Appendix A. First, Issuer $\mathcal{I}$ invokes an EPID Setup protocol and generates a pair of group public/secret keys $(\mathsf{gpk}, \mathsf{gsk})$ as follows:

$$(\mathsf{gpk}, \mathsf{gsk}) \leftarrow \mathsf{Setup}(1^\lambda) \tag{5.27}$$

where $\mathsf{gpk} = (p, G_1, G_2, G_3, g_1, g_2, g_3, h_1, h_2, w = g_2^\gamma)$ and $\mathsf{gsk} = \gamma$. G_1, G_2, G_3 are groups of order p. $g_1, h_1, h_2 \in G_1$, $g_2 \in G_2$, and $g_3 \in G_3$. It outputs $\mathsf{param} = \{\mathsf{gpk}\}$.

$\mathsf{BKeyGen}(\mathsf{param})$ As described earlier, blockchain-based transferable e-cash systems have the blockchain $\mathbb{C}$ in place of the bank B, and $\mathbb{C}$ has no secret information. Thus, $\mathsf{BKeyGen}(\mathsf{param})$ outputs nothing such that $(\mathsf{sk}_\mathcal{B}, \mathsf{pk}_\mathcal{B}) = (\perp, \perp)$.

$\mathsf{UKeyGen}(\mathsf{param})$ Let $\mathcal{P}_i$ be the platform to perform $\mathsf{UKeyGen}$ and $\mathsf{param} = \{\mathsf{gpk}\}$. Note that every $\mathcal{P}_i$ is equipped with Attested Execution Secure Processor denoted by $\mathcal{G}_{\mathrm{att}i}[\mathsf{EPID}, \mathsf{reg} = \{\mathcal{P}_i\}, \mathsf{gpk}, \mathsf{AE}]$ where AE is any symmetric-key authenticated encryption algorithm. First, $\mathcal{P}_i$ initializes $\mathcal{G}_{\mathrm{att}i}$ by invoking $\mathcal{G}_{\mathrm{att}i}.\texttt{initialize}()$ that invokes EPID $\mathsf{Join}(\mathsf{gpk})$ protocol with $\mathcal{I}$.

$$\mathsf{sk}_i = ((A, x, y), f)$$

where (A, x, y) is a BBS+ signature on f. More concretely, let p be the group order of G_1.

$$A = (g_1 T h_2^{y''})^{\frac{1}{x+\gamma}}$$
$$x = \mathbb{Z}_p$$
$$y = y' + y'' \mod p$$

where $y' \leftarrow \mathbb{Z}_p$ is randomly chosen by $\mathcal{P}_i$ and $T = h_1^f \cdot h_2^{y'}$ is sent to $\mathcal{I}$. $y'' \leftarrow \mathbb{Z}_p$ is randomly chosen by $\mathcal{I}$ and the above (A, x, y) is the BBS+ signature on f.

Next, $\mathcal{P}_i$ installs a program prog_w to $\mathcal{G}_{\mathrm{att}i}$. $\mathcal{P}_i$ randomly choose a session id $sid \in \mathbb{Z}$ and sends a message $\texttt{install}(sid, \mathsf{prog}_\mathsf{w})$ to $\mathcal{G}_{\mathrm{att}i}$ and obtains eid from $\mathcal{G}_{\mathrm{att}i}$. Then, $\mathcal{P}_i$ initializes prog_w by sending a message $\texttt{resume}(eid, \text{"}\texttt{initialize}\text{"})$ to $\mathcal{G}_{\mathrm{att}i}$.

It outputs $(\mathsf{sk}_i, \perp)$ as $(\mathsf{sk}_i, \mathsf{pk}_i)$.

$\mathsf{Withdraw}\left(\mathcal{B}\left(\mathsf{sk}_\mathcal{B}, \mathsf{pk}_\mathcal{B}, \mathsf{pk}_i\right), \mathcal{U}\left(\mathsf{sk}_i, \mathsf{pk}_i, \mathsf{pk}_\mathcal{B}\right)\right)$ We only take $\mathsf{param} = \{\mathsf{gpk}\}$, sk_i as inputs and neglect $(\mathsf{sk}_\mathcal{B}, \mathsf{pk}_\mathcal{B}, \mathsf{pk}_i)$ and treat them as $(\perp, \perp, \perp)$. First, $\mathcal{P}_i$ obtains an address ϵ from $\mathcal{G}_{\mathrm{att}i}$. First prog_w randomly choose rid to tell $\mathcal{G}_{\mathrm{att}i}$ to use the randomness on the RandomTape specified by rid.

$$(x, \mathtt{com}) \leftarrow \mathcal{G}_{\mathrm{att}}.\texttt{commitment}(rid)$$
$$\mathsf{addr} \leftarrow \mathsf{Hash}(x, \mathtt{com})$$

The address of escrow account $\epsilon = \mathsf{addr}$. The message flow of the above process is complex. See Figure 5.9 for actual operations between $\mathcal{G}_{\mathrm{att}}$ and $\mathsf{prog_w}$.

Next, $\mathcal{P}_i$ creates a transaction $(X \to \epsilon, \beta; (p, q, \mu))$ and publish it to the blockchain $\mathbb{C}$ to send winning amount to the escrow account ϵ, where X is a normal crypto address controlled by the user i, p, q, μ are the parameters for the probabilistic transferable payments. That is, p is the lottery ticket winning probability, q is the ticket transaction fee rate, μ is the fixed number used to specify the block height to calculate VDF.

It outputs $(\epsilon, \perp)$ as a ticket (id, π).

$\mathsf{Spend}\left(\mathcal{U}_j(\mathsf{id}, \pi, \mathsf{pk}_{\mathcal{U}_i}), \mathcal{U}_i\left(\mathsf{sk}_{\mathcal{U}_i}, \mathsf{pk}_{\mathcal{B}}\right)\right)$ We only take $\mathsf{id}, \pi, \mathsf{sk}_{\mathcal{U}_i}$ as inputs, and neglect $\mathsf{pk}_{\mathcal{U}_i}, \mathsf{pk}_{\mathcal{B}}$ and treat them as $(\perp, \perp)$. First, to establish a secure channel, $\mathcal{U}_j$ randomly choose sid and a, and sends g^a and the attested signature σ_j to $\mathcal{U}_i$ by sending a message $\texttt{resume}(sid, eid, \text{"}\texttt{init keyex}\text{"})$ to $\mathcal{G}_{\mathrm{att}j}$. $\mathcal{U}_i$ generates his temporary receiving address addr_i from the randomness identifier rid and randomly choose b, and encrypts it with the shared secret key $\mathsf{K}\left(= (g^a)^b\right)$ and obtains $\widehat{\mathsf{addr}}_i$, and sends $\widehat{\mathsf{addr}}_i$ and g^b to $\mathcal{P}_j$ by sending a message $\texttt{resume}(eid, (\text{"}\texttt{get addr}\text{"}, sid, g^a, \sigma_j))$ to $\mathcal{G}_{\mathrm{att}j}$.

$$(x, \mathtt{com}) = \mathcal{G}_{\mathrm{att}}.\texttt{commitment}(rid)$$
$$\mathsf{addr} = \mathsf{Hash}(x, \mathtt{com})$$
$$\widehat{\mathsf{addr}} = \mathsf{AE.Enc}_{\mathsf{K}}(\mathsf{addr})$$

Next, $\mathsf{prog_w}$ of $\mathcal{P}_j$ creates a transaction and calls $\mathcal{G}_{\mathrm{att}j}.\texttt{sign}(\tau_{n+1}; rid)$ and obtains the signature σ_j on $(sid, eid, \mathsf{prog_w}, \tau_{n+1})$. The transaction consists as follows:

$$\tau_{n+1} = \left(\mathsf{addr}_j \to \mathsf{addr}_i, \mathsf{Hash}(\tau_n), \sigma_{\mathcal{U}_j}\right)$$

such that

$$\epsilon \prec \tau_1 \prec \ldots \prec \tau_n \prec \tau_{n+1}.$$

Note that the rid specified here must be the same rid used to output the receiving address when receiving the ticket. $\mathcal{P}_j$ sends the encrypted address $\widehat{\mathsf{addr}}_j$ returned from

$$\texttt{resume}(eid, (\text{"}\texttt{get addr}\text{"}, sid, g^a, \sigma_j)) \tag{5.28}$$

call to his $\mathcal{G}_{\text{att}_j}$ when he received the ticket. Since $\widehat{\mathsf{addr}}$ and rid are recorded within prog_w, the rid used when receiving the ticket can be reused when sending.

After that, prog_w of $\mathcal{P}_j$ encrypts $(sid, eid, \mathsf{prog}_\mathsf{w}, \tau_{n+1}, \sigma)$ with the shared secret key and obtains $\widehat{\tau}_{n+1}$. Finally, $\mathcal{P}_j$ sends $\widehat{\tau}_{n+1}$ with the attested signature $\widehat{\tau}$.

It outputs $\langle \text{OK}, \widehat{\tau}^* \rangle$

The message flow of the above process is complex. See Figure 5.10 for actual operations between $\mathcal{G}_{\text{att}}$ and prog_w.

$\mathsf{Deposit}\,(\mathcal{U}\,(\mathsf{id}, \pi, \mathsf{sk}_\mathcal{U}, \mathsf{pk}_\mathcal{U}, \mathsf{pk}_\mathcal{B})\,, \mathcal{B}\,(\mathsf{sk}_\mathcal{B}, \mathsf{pk}_\mathcal{B}, \mathsf{pk}_\mathcal{U}, \mathcal{L}))$ We only take $\mathsf{id}, \pi, \mathsf{sk}_\mathcal{U}$ as inputs, and neglect $\mathsf{pk}_\mathcal{U}, \mathsf{sk}_\mathcal{B}, \mathsf{pk}_\mathcal{B}$ and treat them as $(\perp, \perp)$. $\mathcal{P}$ send his crypto address pk to receive the winning money and the ticket receiving address $\widehat{\mathsf{addr}}$ by sending a message

$$\mathsf{resume}(sid, eid, (\text{"}\texttt{check winning}\text{"}, \mathsf{pk}, \widehat{\mathsf{addr}}))$$

to $\mathcal{G}_{\text{att}}$. If the ticket satisfies the winning condition, prog_w creates a blockchain transaction $\tau := (\mathsf{addr} \to \mathsf{pk}, \mathsf{Hash}(\tau_{\text{pre}}))$ and sends it with the attested signature.

It outputs $\langle \text{OK}, \widehat{\tau}^* \rangle$

The message flow of the above process is complex. See Figure 5.11 for actual operations between $\mathcal{G}_{\text{att}}$ and prog_w.

$\mathsf{Identify}\,(\mathsf{id}, \pi, \pi')$ As described earlier, blockchain-based transferable e-cash systems have the blockchain $\mathbb{C}$ in place of the bank B, this algorithm is executed by any user $\mathcal{U}$. The user receiving the ticket will verify within $\mathcal{G}_{\text{att}}$ that the ticket is a double-spending one.

Detection of double-spending attacks is possible if the winning ticket is registered in the blockchain or if one user receives multiple double-spending tickets. See the Definition 11 and 12. (id, π) corresponds to $\mathsf{id} = \mathsf{Hash}(x, \mathsf{com})$ and $\pi = \sigma$, respectively. Suppose there exists double-spending tickets (id, π) and (id, π') such as

$$\begin{aligned} &\mathsf{addr}_i, (\mathsf{addr}_j \rightarrow \mathsf{addr}_i, \sigma_j) \\ &\mathsf{addr}_{i'}, (\mathsf{addr}_j \rightarrow \mathsf{addr}_{i'}, \sigma'_j) \end{aligned} \tag{5.29}$$

where

$$\begin{aligned} \sigma_j &:= (B, K, T, c, s_x, s_f, s_a, s_b), \\ \sigma'_j &:= (B', K', T', c', s'_x, s'_f, s'_a, s'_b) \end{aligned} \tag{5.30}$$

respectively. Since EPID signature has the Key Extractor from Lemma 1, the adversary's secret key is extractable from σ_j, σ'_j. The extracted secret key f will be registered in the secret-key based revocation list $\mathsf{Priv\text{-}RL}$.

It outputs $(\perp, \Pi_G)$ as $(\perp, \mathsf{Priv\text{-}RL} \cup \{f\})$.

$\mathsf{VerifyGuilt}\,((\mathsf{id}, \pi), \Pi_G)$ In the blockchain-based transferable e-cash systems, this algorithm is used to verify if a ticket is created from the secret key f registered in the secret-key revocation list $\mathsf{Priv\text{-}RL}$. Let

$$\begin{aligned} \mathsf{id} = (x, \mathsf{com}) &= \mathsf{Hash}\,((B, K, T), (R_1, R_2)) \\ \pi &= (B, K, T, c, s_x, s_f, s_a, s_b) \end{aligned} \tag{5.31}$$

and $\Pi_G = \mathsf{Priv\text{-}RL} = \{f_1, f_2, \cdots, f_n\}$. It outputs 1 iff

$$\begin{cases} \hat{R}_1 = B^{s_f} \cdot K^{-c} \\ \hat{R}_2 = e\left(T, g_2\right)^{-s_x} \cdot e\left(h_1, g_2\right)^{s_f} \cdot e\left(h_2, g_2\right)^{s_b} \\ \qquad\qquad \cdot e\left(h_2, \omega\right)^{s_a} \cdot (e(g_1, g_2)/e(T, \omega))^c \\ c = \mathsf{Hash}(\mathsf{gpk}, B, K, T, \hat{R}_1, \hat{R}_2, m) \\ \qquad \text{where } m = (\mathsf{addr}_j \rightarrow \mathsf{addr}_i) \\ {}^{\exists} f' \in \mathsf{Priv\text{-}RL}, B^{f'} = K, \end{cases} \tag{5.32}$$

otherwise outputs 0.

5.7.1 Ticket Transfer Overview

The actual operations of the algorithms described above are processed within $\mathcal{G}_{\text{att}}$ and $\mathsf{prog}_{\mathsf{w}}$, and the flow can be complex. The following describes the operations between the wallet owner (described as Wallet) and $\mathcal{G}_{\text{att}}$ and $\mathsf{prog}_{\mathsf{w}}$ for setting up the escrow, making the payment, and processing the winning ticket.

Step 1: The ticket issuer issues a smart contract escrow account ϵ and confirms that ϵ has been registered in the blockchain. **Step 2:** The payee sends the ticket τ to a payee. The payee verifies that the ticket came from a legitimate wallet and that the escrow account ϵ is registered correctly in the blockchain. If there is no problem, the payee receives the ticket and returns the service or product to the payer. Then, the payee signs the ticket with his wallet and sends it to another user. **Step 3:** If the ticket received meets the requirements for lottery winning, The ticket owner can use the ticket to send the winning amount to their address.

Escrow Setup

Figure 5.9 shows the flow diagram. This process is part of $\mathsf{Withdraw}$. It executes a deposit transaction τ_l on the output ϵ and registers τ_l to the blockchain network B. The issuer's

wallet W_X requests an escrow account ϵ from $\mathcal{G}_{\text{att}}$. After the wallet obtains ϵ from $\mathcal{G}_{\text{att}}$, W_X creates a transaction τ_l that transfer the winning amount β to ϵ and sends it to the blockchain network.

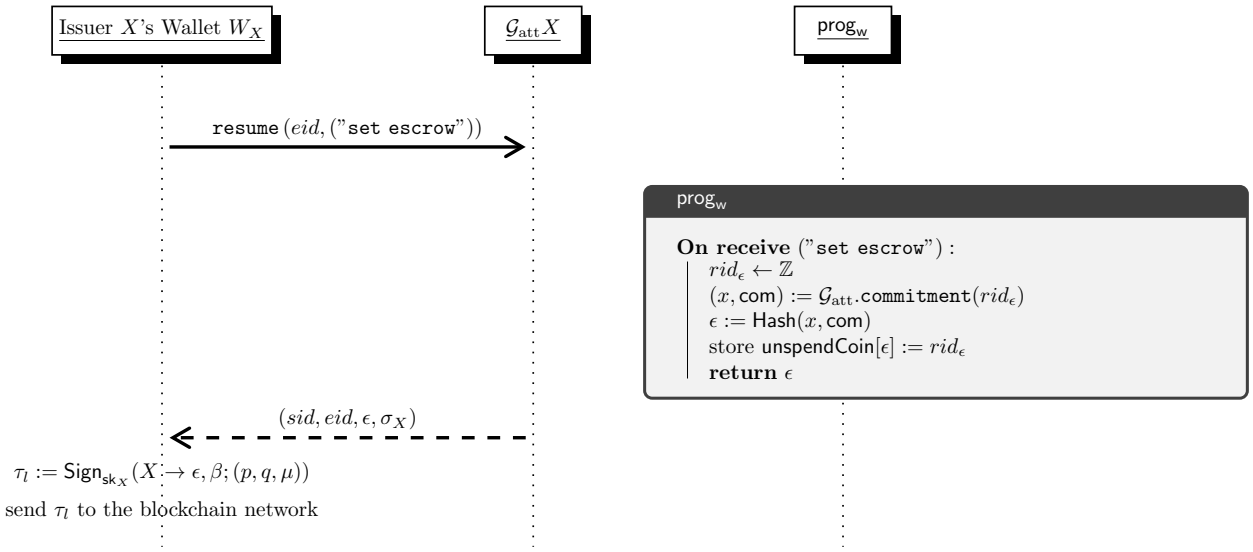


Figure 5.9: Escrow Setup

Payment with Lottery Ticket

Figure 5.10 shows the flow diagram which shows sending a ticket from X to Y. Suppose that the X's wallet has a ticket τ_n and generates τ_{n+1} or generates τ_1 from the escrow account ϵ.

First, the sender X's wallet W_X resumes "`init keyex`" to perform Diffie-Hellman key exchange with the payer Y, and requests invoice to Y's wallet W_Y. $\mathcal{G}_{\text{att}}$ in W_Y generates the receiving address and encrypts it with the DH shared secret key, then sends it to W_X.

Second, W_X processes ticket transfers to the address sent by W_Y. W_X passes to $\mathcal{G}_{\text{att}}$ the address received from W_Y and the encrypted address $\widehat{\mathsf{addr}}_X$ used to receive the ticket or create an escrow account in the past.

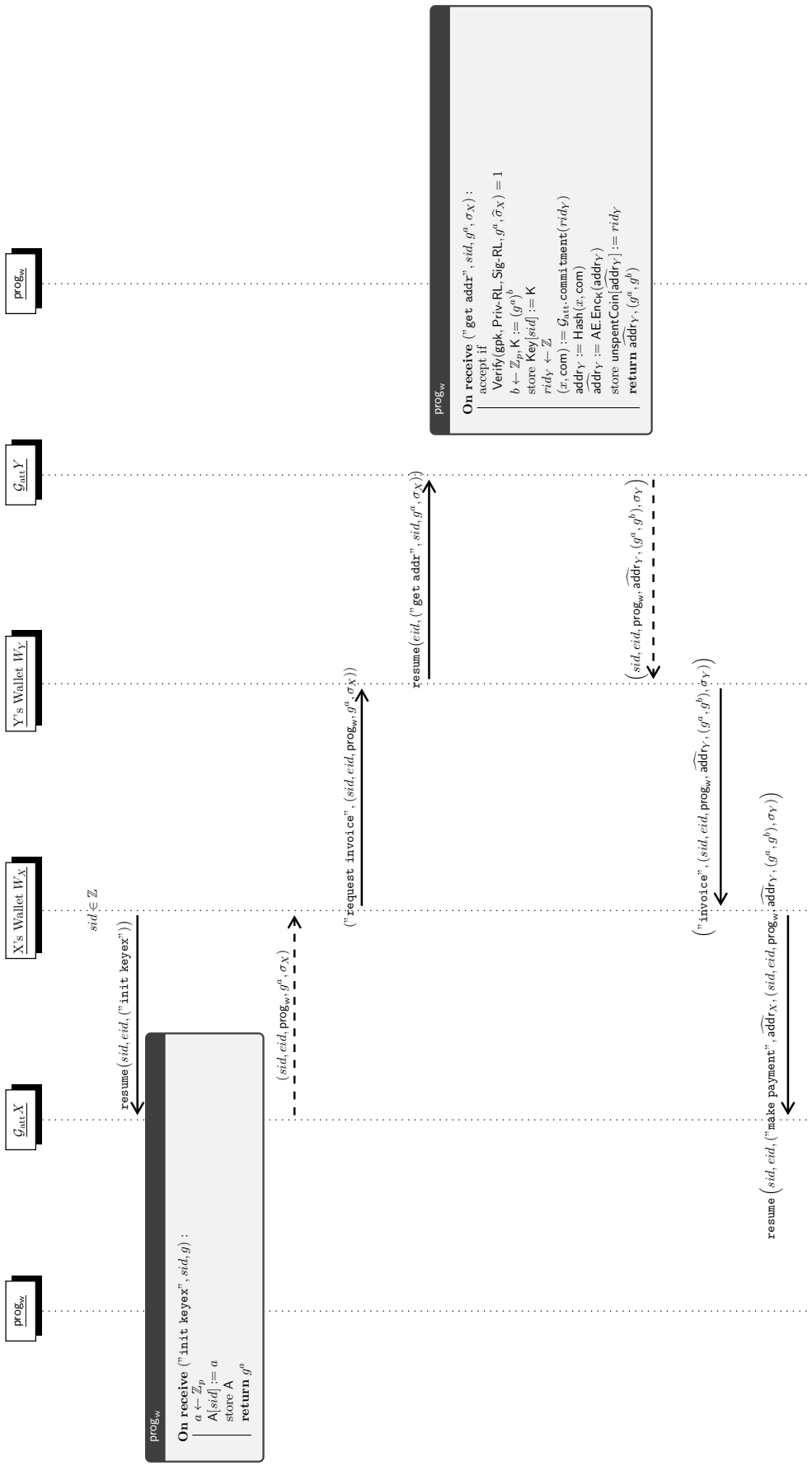
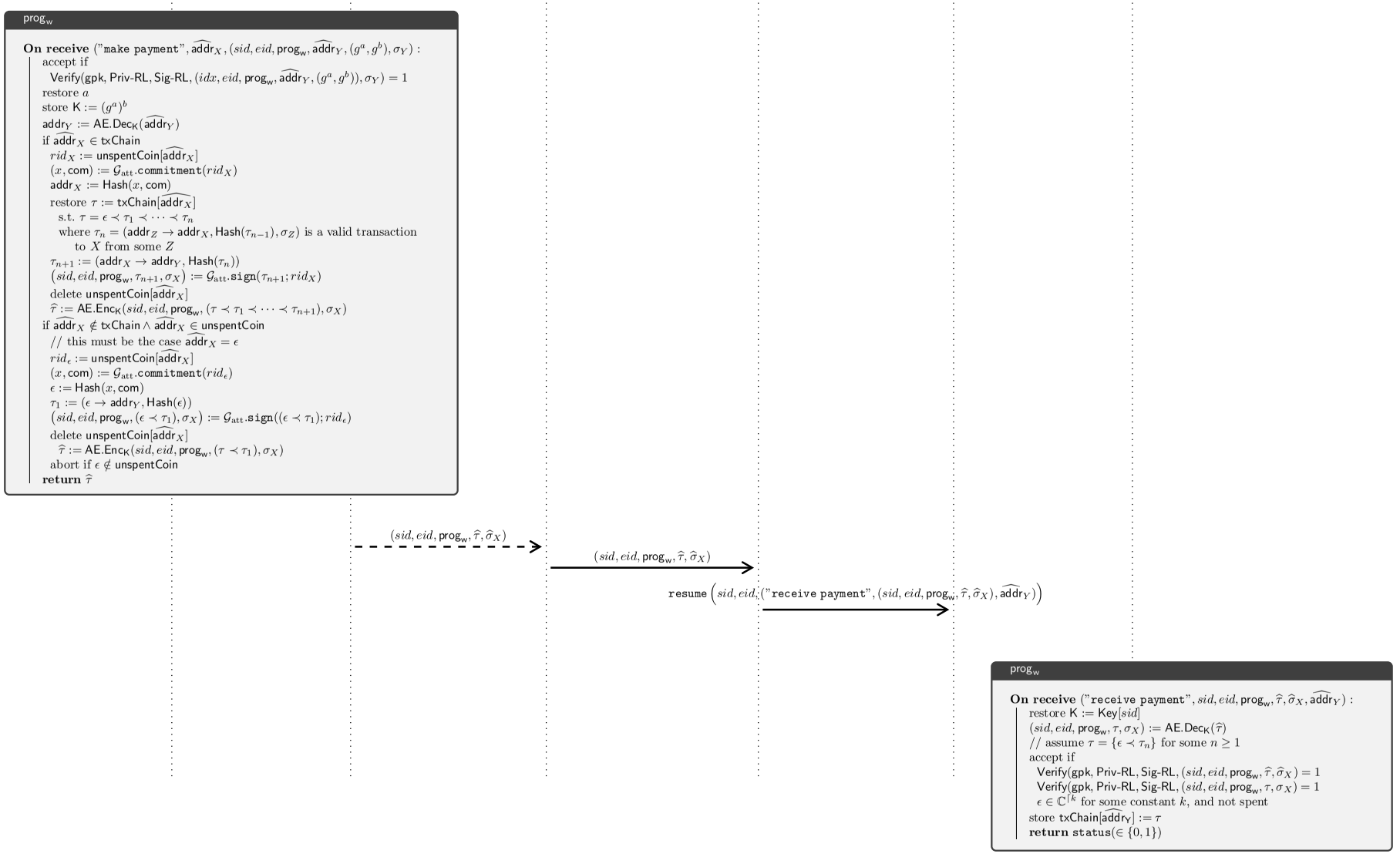


Figure 5.10: Anonymous Payment with Lottery Tickets (Payment from X to Y)


prog_w
On receive ("make payment", $\widehat{\mathsf{addr}}_X$, $(sid, eid, \mathsf{prog_w}, \widehat{\mathsf{addr}}_Y, (g^a, g^b), \sigma_Y)$:
accept if
Verify(gpk, Priv-RL, Sig-RL, $(idx, eid, \mathsf{prog_w}, \widehat{\mathsf{addr}}_Y, (g^a, g^b)), \sigma_Y) = 1$
restore a
store $\mathsf{K} := (g^a)^b$
$\mathsf{addr}_Y := \mathsf{AE.Dec_K}(\widehat{\mathsf{addr}}_Y)$
if $\widehat{\mathsf{addr}}_X \in \mathsf{txChain}$
$rid_X := \mathsf{unspentCoin}[\widehat{\mathsf{addr}}_X]$
$(x, \mathsf{com}) := \mathcal{G}_{\mathrm{att}}.\mathtt{commitment}(rid_X)$
$\mathsf{addr}_X := \mathsf{Hash}(x, \mathsf{com})$
restore $\tau := \mathsf{txChain}[\widehat{\mathsf{addr}}_X]$
s.t. $\tau = \epsilon \prec \tau_1 \prec \cdots \prec \tau_n$
where $\tau_n = (\mathsf{addr}_Z \to \mathsf{addr}_X, \mathsf{Hash}(\tau_{n-1}), \sigma_Z)$ is a valid transaction to X from some Z
$\tau_{n+1} := (\mathsf{addr}_X \to \mathsf{addr}_Y, \mathsf{Hash}(\tau_n))$
$(sid, eid, \mathsf{prog_w}, \tau_{n+1}, \sigma_X) := \mathcal{G}_{\mathrm{att}}.\mathtt{sign}(\tau_{n+1}; rid_X)$
delete $\mathsf{unspentCoin}[\widehat{\mathsf{addr}}_X]$
$\widehat{\tau} := \mathsf{AE.Enc_K}(sid, eid, \mathsf{prog_w}, (\tau \prec \tau_1 \prec \cdots \prec \tau_{n+1}), \sigma_X)$
if $\widehat{\mathsf{addr}}_X \notin \mathsf{txChain} \wedge \widehat{\mathsf{addr}}_X \in \mathsf{unspentCoin}$
// this must be the case $\widehat{\mathsf{addr}}_X = \epsilon$
$rid_\epsilon := \mathsf{unspentCoin}[\widehat{\mathsf{addr}}_X]$
$(x, \mathsf{com}) := \mathcal{G}_{\mathrm{att}}.\mathtt{commitment}(rid_\epsilon)$
$\epsilon := \mathsf{Hash}(x, \mathsf{com})$
$\tau_1 := (\epsilon \to \mathsf{addr}_Y, \mathsf{Hash}(\epsilon))$
$(sid, eid, \mathsf{prog_w}, (\epsilon \prec \tau_1), \sigma_X) := \mathcal{G}_{\mathrm{att}}.\mathtt{sign}((\epsilon \prec \tau_1); rid_\epsilon)$
delete $\mathsf{unspentCoin}[\widehat{\mathsf{addr}}_X]$
$\widehat{\tau} := \mathsf{AE.Enc_K}(sid, eid, \mathsf{prog_w}, (\tau \prec \tau_1), \sigma_X)$
abort if $\epsilon \notin \mathsf{unspentCoin}$
return $\widehat{\tau}$
$(sid, eid, \mathsf{prog_w}, \widehat{\tau}, \widehat{\sigma}_X)$
$(sid, eid, \mathsf{prog_w}, \widehat{\tau}, \widehat{\sigma}_X)$
resume $\Big(sid, eid,$ ("receive payment", $(sid, eid, \mathsf{prog_w}, \widehat{\tau}, \widehat{\sigma}_X), \widehat{\mathsf{addr}}_Y)\Big)$
prog_w
On receive ("receive payment", $sid, eid, \mathsf{prog_w}, \widehat{\tau}, \widehat{\sigma}_X, \widehat{\mathsf{addr}}_Y$) :
restore $\mathsf{K} := \mathsf{Key}[sid]$
$(sid, eid, \mathsf{prog_w}, \tau, \sigma_X) := \mathsf{AE.Dec_K}(\widehat{\tau})$
// assume $\tau = \{\epsilon \prec \tau_n\}$ for some $n \geq 1$
accept if
Verify(gpk, Priv-RL, Sig-RL, $(sid, eid, \mathsf{prog_w}, \widehat{\tau}, \widehat{\sigma}_X) = 1$
Verify(gpk, Priv-RL, Sig-RL, $(sid, eid, \mathsf{prog_w}, \tau, \sigma_X) = 1$
$\epsilon \in \mathbb{C}^{|k}$ for some constant k, and not spent
store $\mathsf{txChain}[\widehat{\mathsf{addr}}_Y] := \tau$
return status$(\in \{0, 1\})$

Ticket winning and Revocation

When the ticket satisfies the winning condition, the ticket owner sends the winning ticket to the blockchain network. See Figure 5.11.

The ticket owner sends his address pk_Y and the encrypted address used to receive the ticket to $\mathcal{G}_{\text{att}}$. Inside $\mathcal{G}_{\text{att}}$, $\mathcal{G}_{\text{att}}$ creates and returns the transaction to transfer to the address sent by the ticket owner. Then, the ticket owner receives the transaction and sends it to the blockchain network.

If the winning ticket is a double-spending one, the adversary's secret key is extracted by Fork and collision detection.

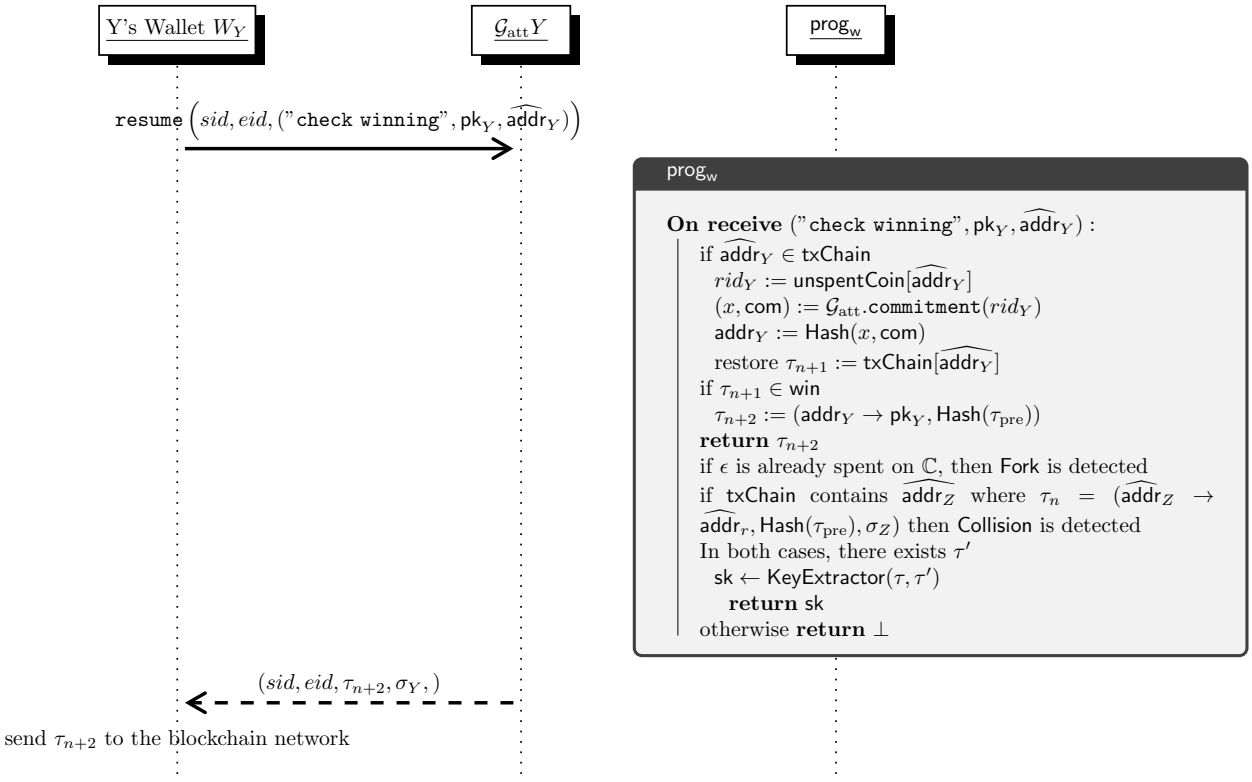


Figure 5.11: Ticket redemption

5.8 Security Analysis

In this section, we analyze whether VeloCash satisfies economic and anonymity properties.

Finally, we analyze whether the proportional fee scheme and double-spending attacks detection methods, a requirement of the transferable micropayment scheme, can be achieved even if the anonymity is preserved.

We construct the theorems under the following assumptions. First, we assume that the tamper-proof hardware wallet and the collision-resistant hash function exist.

Assumption 1. *κ-tamper proof hardware defined in Definition 10 exists.*

Assumption 2. *For all PPT adversaries $\mathcal{A}$ there is a negligible function* negl*, collision resistant hash function* Hash *exists that satisfies the following inequality:*

$$\Pr\left[\mathsf{Hash}(x) = \mathsf{Hash}(x')\right] < \mathsf{negl}(n) \tag{5.33}$$

where $^{\forall}n \in \mathbb{N}_{>0}, x, x' \in \{0,1\}^{|n|}$ and $x \neq x'$.

Next, we describe the assumptions for ensuring EPID's anonymity and unforgeability.

Assumption 3. *The q-SDH problem and the Decisional Diffie-Hellman (DDH) problem are hard.*

We assume that the blockchain network is agreed upon among honest users to simplify the proof.

Assumption 4. *A blockchain network is agreed upon by honest users with a public parameter k satisfying the three properties the common prefix property, the chain quality property and the common-prefix property proposed by Garay et al. [GKL15].*

5.8.1 Economic properties

In this section, we verify that the protocol of $\mathsf{VeloCash}$, Π_{VC} satisfies the economic properties.

Theorem 7 (Soundness)**.** *The protocol of $\mathsf{VeloCash}$, Π_{VC} is **sound**. More formally, for all PPT adversaries $\mathcal{A}$, there exists $^{\exists}\lambda_0 \in \mathbb{N}$ such that for all security parameter $\lambda \geq \lambda_0$, Π_{VC}*

satisfies the following inequality:

$$\mathcal{A}^{\mathtt{sound}}_{\mathcal{A},\Pi_{VC}}(\lambda) = \Pr\left[\mathsf{Expt}^{\mathtt{sound}}_{\mathcal{A},\Pi_{VC}}(\lambda) = 1\right] < \mathsf{negl}(\lambda). \tag{5.34}$$

Proof. Suppose there exists an AESP A_1 of wallet W_1 and another AESP A_2 of wallet W_2, and both of them are honest. The adversary wins if A_1 receives a ticket from the adversary and spends it on A_2, and A_2 refuses. Due to the tamper-proof assumption, the adversary can not break his tamper-proof wallet and can not perform double-spending attacks. Thus, to win the game, the adversary has to forge the EPID secret key and creates a ticket with a valid EPID signature. However, since the probability of forging an EPID signature is negligible small from the EPID's unforgeability in the Theorem 6, A_1 refuses to receive it. □

If the adversary can break his tamper-proof wallet, the adversary will perform double-spending attacks. Suppose the adversary performs double-spending attacks on A_1 and other users, and the attack is detected. In that case, this must be a case where the adversary's secret key is registered in Priv-RL, and A_2 refuses payment from A_1. Soundness is broken if $\mathsf{sk}_{\mathcal{A}} \notin$ Priv-RL when A_1 receives the ticket from $\mathcal{A}$, but $\mathsf{sk}_{\mathcal{A}} \in$ Priv-RL when A_2 received the ticket from A_1. However, for an adversary to gain sufficient economic benefit from a double-spending attack, a large number of double-spending would be required. From Theorem 3, there exists an upper bound on the expected utility of the attack. For users who receive double-spending tickets, it is possible to compensate for the loss by setting an appropriate issuance.

The temporal collapse of soundness is a universal issue that also exists in E-Cash due to the timing gap between the execution of the attack and disabling the adversary.

Theorem 8 (Unforgeability)**.** *The protocol of VeloCash, Π_{VC} is **unforgeable**. More formally, for all PPT adversaries $\mathcal{A}$, there exists there exists $^{\exists}\lambda_0 \in \mathbb{N}$ such that for all security*

parameter $\lambda \geq \lambda_0$, Π_{VC} satisfies the following inequality:

$$\mathcal{A}^{\mathsf{Unforg}}_{\mathcal{A},\Pi_{\mathsf{VC}}}(\lambda) = \Pr\left[\mathsf{Expt}^{\mathsf{Unforg}}_{\mathcal{A},\Pi_{\mathsf{VC}}}(\lambda) = 1\right] < \mathsf{negl}(\lambda). \tag{5.35}$$

Proof. The adversary wins the unforgeability game $\mathrm{Expt}^{\mathsf{Unforg}}_{\mathcal{A},\Pi}(\lambda)$ if he succeeds in creating a new valid ticket (id, π) which is not included in the supplied coin list $\mathcal{SC}$ or to create multiple proofs π, π' for the same ticket id, id, which is included in $\mathcal{SC}$, to get (id, π, π') but never identified as a double-spender by the $\mathsf{Identify}(\mathsf{id}, \pi, \pi')$ algorithm. The former case corresponds to the existential unforgeability property of EPID in VeloCash Π_{VC}. We focus on the latter case, which is further divided into three cases:

Case 1: $\mathsf{Identify}(\mathsf{id}, \pi, \pi') = \perp$

This is the case that the adversary's secret key is not extractable from the double-spending tickets.

Suppose there exists

$$\begin{cases} (x, \mathtt{com}, \mathtt{ch}, \mathtt{resp}) \\ (x, \mathtt{com}, \mathtt{ch}', \mathtt{resp}') \end{cases} \tag{5.36}$$

such that

$$\begin{aligned} \mathsf{V}(x, \mathtt{com}, \mathtt{ch}, \mathtt{resp}) = \mathsf{V}(x, \mathtt{com}, \mathtt{ch}', \mathtt{resp}') \\ = 1. \end{aligned} \tag{5.37}$$

Since KeyExtractor algorithm in VeloCash compute the secret key f by the following equation:

$$f = \frac{s_f - s'_f}{c - c'}. \tag{5.38}$$

The probability that KeyExtractor can not output the secret key f is equal to the probability of being $\mathtt{ch} = \mathtt{ch}'$ in $(x, \mathtt{com}, \mathtt{ch}, \mathtt{resp})$ and $(x, \mathtt{com}, \mathtt{ch}', \mathtt{resp}')$. In order to be verified

correctly, the two challenges $\mathtt{ch}, \mathtt{ch}'$ have to be computed, that is, c and c' in the EPID signatures have to be computed by the following equations:

$$\begin{aligned} c &= \mathsf{Hash}\Big(\mathsf{gpk}, B, K, T, R_1, R_2, (\mathsf{addr}_j \to \mathsf{addr}_i, \mathsf{Hash}(\tau_{\mathrm{pre}}))\Big), \\ c' &= \mathsf{Hash}\Big(\mathsf{gpk}, B, K, T, R_1, R_2, (\mathsf{addr}_j \to \mathsf{addr}_{i'}, \mathsf{Hash}(\tau_{\mathrm{pre}}))\Big). \end{aligned} \tag{5.39}$$

The probability of $c = c'$ is upper bounded by the following inequality:

$$\Pr[c = c'] + \Pr[\mathsf{addr}_i = \mathsf{addr}_{i'}] < \mathsf{negl}(\lambda) \tag{5.40}$$

by the collision resistant property Hash functions. Note that addr_i and $\mathsf{addr}_{i'}$ are hash of randomly generated commitments (in the form of $(x, \mathtt{com})$) of honest recipients.

Case 2: $(\perp, \Pi_G) \leftarrow \mathsf{Identify}(\mathsf{id}, \pi, \pi') \wedge$
$\mathsf{VerifyGuilt}((\mathsf{id}, \pi), \Pi_G) = 0 \wedge \mathsf{VerifyGuilt}((\mathsf{id}, \pi'), \Pi_G) = 0$
This case does not happen unless $B = 1$. Even if the adversary make $\mathsf{id} = (x, \mathtt{com}) = ((B, K, T), (R_1, R_2))$ with $B = 1$, this case is eliminated by the construction of $\mathsf{VeloCash}$ since the honest receiver does not accept the ticket as Verify algorithm fails by Equation (A.8).

Case 3: $\mathsf{Identify}(\mathsf{id}, \pi, \pi') = (\perp, \Pi_G \ni \mathsf{sk}) \wedge \mathsf{sk} \notin \mathcal{UL}$
This must be the case that the adversary succeeded to forge EPID secret key f. Since the EPID's unforgeability property, the probability of succeeding is negligibly small in the security parameter λ.

Next, consider the case where the different $\mathsf{sk}'_{\mathcal{A}} (= f')$ is extracted in Equation (5.37), that is, the same two $\mathtt{com}$ $(= (R_1, R_2))$ are created with different secret key f. □

The following lemma states that, once $\mathsf{VerifyGuilt}$ outputs 1 on input one of the signature (id_i, π_i) with Π_G for some honest user i, then it outputs 1 for the other signatures $(\mathsf{id}^*_i, \pi^*_i)$ with Π_G of the same user i.

Lemma 2. *Let* $\Pi_G = \mathsf{Priv\text{-}RL}$. *Let* (id_i, π_i) *be the signatures of a user* i. *Then, for all honest user* $i \in \mathcal{UL}$ *and for all signature* $(\mathsf{id}_i^*, \pi_i^*)$ *generated by* i, *the following holds:*

$$\begin{aligned} &\mathsf{VerifyGuilt}((\mathsf{id}_i, \pi_i), \Pi_G) = 1 \\ &\Rightarrow \mathsf{VerifyGuilt}((\mathsf{id}_i^*, \pi_i^*), \Pi_G) = 1 \end{aligned} \tag{5.41}$$

Proof. Given the condition of the lemma, the signature (id_i, π_i) is signed by an honest user i, hence correct. From the given condition $\mathsf{VerifyGuilt}((\mathsf{id}_i, \pi_i), \Pi_G) = 1$, we see that $c = \mathsf{Hash}(\mathsf{gpk}, B, K, T, \hat{R}_1, \hat{R}_2, m)$ in Equation (5.32) always holds. Therefore, we focus on whether the last condition in Equation (5.32) holds. Let f be a part of the secret key of the user i which is used in signing the two signatures (id_i, π_i) and $(\mathsf{id}_i^*, \pi_i^*)$. We consider the following two cases:

Case 1: $f \in \Pi_G (= \mathsf{Priv\text{-}RL})$

$\mathsf{VerifyGuilt}((\mathsf{id}_i^*, \pi_i^*), \Pi_G) = 1$ always holds.

Case 2: $f \notin \Pi_G (= \mathsf{Priv\text{-}RL})$

This must be the case that B, K and B^*, K^* in σ and σ^* respectively holds the relation $B^f = K$ and $B^{*f} = K^*$ for the honest user's secret key $f \notin \Pi_G$, But, in order to satisfy the condition in Equation (5.41), there must exist some $f' \in \Pi_G$ different form $f \neq f'$ satisfying $B^{f'} = K$.

Recall that $B \neq 1$ is randomly chosen generator in G_3 of prime order p. Therefore,

$$B^f = K \text{ and } B^{f'} = K \Rightarrow f = f'. \tag{5.42}$$

This contradicts the assumption. Hence, the lemma.

□

Theorem 9. *(Exculpability) The protocol of* VeloCash, Π_{VC} *is* ***exculpable****. More formally,*

for all PPT adversaries $\mathcal{A}$, there exists $^{\exists}\lambda_0 \in \mathbb{N}$ such that for all security parameter $\lambda \geq \lambda_0$, Π_{VC} satisfies the following inequality:

$$\mathsf{Adv}^{\mathtt{excul}}_{\mathcal{A},\Pi}(\lambda) = \Pr\left[\mathsf{Expt}^{\mathtt{excul}}_{\mathcal{A},\Pi}(\lambda) = 1\right] < \mathsf{negl}(\lambda). \tag{5.43}$$

Proof. (sketch) Assume that there exists a PPT adversary $\mathcal{A}_{\mathsf{ex}}$ that can win the exculpability game and can output $(\mathsf{id}, \pi, \Pi_G)$ such that $\mathsf{VerifyGuilt}((\mathsf{id}, \pi), \Pi_G) = 1$. We show that $\mathcal{A}$ can win the EPID's anonymity game by using $\mathcal{A}_{\mathsf{ex}}$; however, this contradicts the anonymity property of EPID stated in Theorem 5. Therefore, there is no $\mathcal{A}_{\mathsf{ex}}$ that wins the exculpability game. See the detailed proof in Appendix C. □

5.8.2 Anonymity properties

In this section, we verify that the protocol of VeloCash, Π_{VC} satisfies the anonymity properties.

Assumption 5 (INT-CTXT and IND-CPA)**.** *The symmetric-key authenticated encryption scheme* AE *consists of three algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *where key generation algorithm* Gen*, encryption algorithm* Enc*, and decryption algorithm* Dec*.* AE *is INT-CTXT and IND-CPA secure, for all PPT adversaries $\mathcal{A}$ there exists a negligible function* negl *such that:*

$$\begin{aligned}
&\mathsf{Adv}^{\mathsf{INT\text{-}CTXT}}_{\mathcal{A},\mathsf{AE}}(\lambda) \\
&\quad = \Pr\left[\mathsf{Expt}^{\mathsf{INT\text{-}CTXT}}_{\mathcal{A},\mathsf{AE}}(\lambda) = 1\right] < \mathsf{negl}(\lambda), \\
&\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A},\mathsf{AE}}(\lambda) \\
&\quad = \Pr\left[\mathsf{Expt}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A},\mathsf{AE}}(\lambda) = 1\right] < \mathsf{negl}(\lambda)
\end{aligned} \tag{5.44}$$

where $\mathsf{Expt}^{\mathsf{INT\text{-}CTXT}}_{\mathcal{A},\mathsf{AE}}(\lambda)$ *and* $\mathsf{Expt}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A},\mathsf{AE}}(\lambda)$ *corresponds to INT-CTXT$^{A}_{\mathcal{SE}}$ and IND-CPA$^{A}_{\mathcal{SE}}$ in Bellare et al. [BN08], respectively.*

From Assumption 5 and Bellare et al. [BN08], we see that the symmetric-key authenti-

cated encryption scheme AE satisfies IND-CCA secure.

Theorem 10 (IND-CCA)**.** *From Theorem 3.1 of [BN08], symmetric-key authenticated encryption scheme* AE *is IND-CCA secure , if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* negl *such that:*

$$\begin{aligned} &\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathcal{A},\mathsf{AE}}(\lambda) \leq \\ &\quad 2 \cdot \mathsf{Adv}^{\mathsf{INT\text{-}CTXT}}_{\mathcal{A},\mathsf{AE}}(\lambda) + \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A},\mathsf{AE}}(\lambda) < \mathsf{negl}(\lambda) \end{aligned} \tag{5.45}$$

where the probability is taken over all randomness used in the experiment.

Theorem 11 (Coin anonymity (c-an))**.** *For any* $\epsilon_0 > 0$*, the protocol of* ***VeloCash****,* Π_{VC} *satisfies* ***coin anonymity*** *if* $k_0 < \frac{\epsilon_0}{2p} - 1$ *where* k_0 *is the number of challenge users per group and* p *is the winning probability. More formally, for all PPT adversaries* $\mathcal{A}$*, there exists* $^{\exists}\lambda_0 \in \mathbb{N}$ *such that for all security parameter* $\lambda \geq \lambda_0$*,* Π_{VC} *satisfies the following inequality:*

$$\begin{aligned} &\mathsf{Adv}^{\mathtt{c\text{-}an}}_{\mathcal{A},\Pi}(\lambda) = \\ &\left|\Pr\left[\mathsf{Expt}^{\mathtt{c\text{-}an}}_{\mathcal{A},\Pi,1}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathtt{c\text{-}an}}_{\mathcal{A},\Pi,0}(\lambda) = 1\right]\right| < \mathsf{negl}(\lambda). \end{aligned} \tag{5.46}$$

Proof. Consider the case where $k_0 = 0$, that is, the case of a single challenge user. The adversary issues and spend the tickets such that

$$\begin{aligned} \widehat{\tau}_0^{(0)} &= \left(\mathsf{AE.Enc}_{\mathsf{K}_0^{(0)}}\left(\epsilon^{(0)} \to i_0^{(0)}, \sigma_{\mathcal{A}}^{(0)}\right), \widehat{\sigma}_{\mathcal{A}}^{(0)}\right), \\ \widehat{\tau}_0^{(1)} &= \left(\mathsf{AE.Enc}_{\mathsf{K}_0^{(1)}}\left(\epsilon^{(1)} \to i_0^{(1)}, \sigma_{\mathcal{A}}^{(1)}\right), \widehat{\sigma}_{\mathcal{A}}^{(1)}\right) \end{aligned} \tag{5.47}$$

where $\sigma_{\mathcal{A}}^{(\cdot)}$ is the adversary's EPID signature on the message $\epsilon^{(\cdot)} \to i_0^{(\cdot)}$ and $\widehat{\sigma}_{\mathcal{A}}^{(\cdot)}$ is the adversary's EPID signature on the ciphertext $\mathsf{AE.Enc}_{\mathsf{K}_0^{(\cdot)}}(\cdot,\cdot)$.

Then, the challenge users $i_0^{(0)}$ and $i_0^{(1)}$ return the adversary the following tickets:

$$\begin{aligned}\widehat{\tau}_1^{(0)} &= \left(\mathsf{AE.Enc}_{\mathsf{K}_1^{(0)}}\left(\left(\epsilon^{(0)} \to i_0^{(0)}, \sigma_{\mathcal{A}}\right), \left(i_0^{(0)} \to \mathcal{A}, \sigma_{i_0^{(0)}}\right)\right), \widehat{\sigma}_{i_0^{(0)}}\right), \\ \widehat{\tau}_1^{(1)} &= \left(\mathsf{AE.Enc}_{\mathsf{K}_1^{(1)}}\left(\left(\epsilon^{(1)} \to i_0^{(1)}, \sigma_{\mathcal{A}}\right), \left(i_0^{(1)} \to \mathcal{A}, \sigma_{i_0^{(1)}}\right)\right), \widehat{\sigma}_{i_0^{(1)}}\right).\end{aligned} \tag{5.48}$$

The adversary can infer two partial messages as follows:

$$\begin{cases} m_0 : (\epsilon^{(0)} \to i_0^{(0)}, \cdot), (i_0^{(0)} \to \mathcal{A}, \cdot) \\ m_1 : (\epsilon^{(1)} \to i_0^{(1)}, \cdot), (i_0^{(1)} \to \mathcal{A}, \cdot) \end{cases} \tag{5.49}$$

such that whose signature parts are hidden from the adversary. It is easy to see that the adversary who wins the IND-CCA game with such partially hidden messages can always win the standard full message IND-CCA game. Thus, given the anonymity property of EPID signatures, we can reduce the IND-CCA game of AE to the coin anonymity game.

The above argument holds regardless of the number of challenge users $k_0 \geq 1$. This concludes the proof. □

Open problem of Coin anonymity (c-an) If one of the challenge users wins and the ticket is registered in the blockchain, that is when $k_0 \geq \frac{\sigma_0}{2p} - 1$, the adversary can win the game in the coin anonymity game. Since the adversary issued and know the escrow account, he can see the challenge user group from the plain-text winning ticket output by $\mathsf{prog_w}$ registered in the blockchain. The reason for this is that the blockchain does not preserve anonymity. Thus, using a blockchain with anonymity, such as Confidential Transaction or ZeroCash [BSCG+14] may solve the problem.

Theorem 12 (User anonymity (u-an))**.** *The protocol of VeloCash,* Π_{VC} *satisfies* ***user anonymity****. More formally, for all PPT adversaries* $\mathcal{A}$*, there exists* $^{\exists}\lambda_0 \in \mathbb{N}$ *such that for all security*

parameter $\lambda \geq \lambda_0$, Π_{VC} *satisfies the following inequality:*

$$\begin{aligned} &\mathsf{Adv}^{\texttt{u-an}}_{\mathcal{A},\Pi}(\lambda) = \\ &\left|\Pr\left[\mathsf{Expt}^{\texttt{u-an}}_{\mathcal{A},\Pi,1}(\lambda)=1\right] - \Pr\left[\mathsf{Expt}^{\texttt{u-an}}_{\mathcal{A},\Pi,0}(\lambda)=1\right]\right| < \mathsf{negl}(\lambda). \end{aligned} \tag{5.50}$$

Proof. Consider the case where $k_0 = 0$, that is, the case of a single challenge user. The adversary issues and spend the ticket such that

$$\widehat{\tau}_0^{(b)} = \left(\mathsf{AE.Enc}_{\mathsf{K}_0}\left(\epsilon \to i_0^{(b)}, \sigma_{\mathcal{A}}\right), \widehat{\sigma}_{\mathcal{A}}\right) \tag{5.51}$$

where $b \in \{0,1\}$. Then, the challenge user returns the following tickets to the adversary.

$$\widehat{\tau}_1^{(b)} = \left(\mathsf{AE.Enc}_{\mathsf{K}_1}\left(\left(\epsilon \to i_0^{(b)}, \sigma_{\mathcal{A}}\right), \left(i_0^{(b)} \to \mathcal{A}, \sigma_{i_0^{(b)}}\right)\right), \widehat{\sigma}_{i_0^{(b)}}\right). \tag{5.52}$$

The adversary can infer two partial messages as follows:

$$\begin{cases} m_0 : (\epsilon^{(0)} \to i_0^{(0)}, \cdot), (i_0^{(0)} \to \mathcal{A}, \cdot) \\ m_1 : (\epsilon^{(1)} \to i_0^{(1)}, \cdot), (i_0^{(1)} \to \mathcal{A}, \cdot) \end{cases} \tag{5.53}$$

such that whose signature parts are hidden from the adversary. Similar to the coin anonymity game, it is easy to see that the adversary who wins the IND-CCA game with such partially hidden messages can always win the standard full message IND-CCA game. Thus, given the anonymity property of the EPID signature, we can reduce IND-CCA of AE to the user anonymity game. The above argument holds regardless of the number of challenge users $k_0 \geq 1$. This concludes the proof. □

Theorem 13 (coin transparency (c-tr))**.** *The protocol of VeloCash,* Π_{VC} *satisfies* ***coin transparency****. More formally, for all PPT adversaries* $\mathcal{A}$*, there exists* $^{\exists}\lambda_0 \in \mathbb{N}$ *such that*

for all security parameter $\lambda \geq \lambda_0$, Π_{VC} satisfies the following inequality:

$$\begin{aligned}&\mathsf{Adv}^{\texttt{c-tr}}_{\mathcal{A},\Pi}(\lambda) = \\ &\left|\Pr\left[\mathsf{Expt}^{\texttt{c-tr}}_{\mathcal{A},\Pi,1}(\lambda) = 1\right] - \Pr\left[\mathsf{Expt}^{\texttt{c-tr}}_{\mathcal{A},\Pi,0}(\lambda) = 1\right]\right| < \mathsf{negl}(\lambda).\end{aligned} \tag{5.54}$$

Proof. The adversary receives the tickets as follows:

$$\begin{aligned}&\widehat{\tau}_n^{(0)} = \\ &\Bigg(\mathsf{AE.Enc}_{\mathsf{K}_{n-1}}\Bigg(\left(\epsilon^{(0)} \to \tau_1^{(0)}, \sigma_{i_0^{(0)}}\right), \\ &\qquad\qquad \left(\tau_1^{(0)} \to \tau_2^{(0)}, \sigma_{i_1^{(0)}}\right), \cdots, \\ &\qquad\qquad \left(i_{n-1}^{(0)} \to \mathcal{A}, \sigma_{i_{n-1}^{(0)}}\right)\Bigg), \widehat{\sigma}_{i_{n-1}^{(0)}}\Bigg), \\ &\widehat{\tau}_n^{(1)} = \\ &\Bigg(\mathsf{AE.Enc}_{\mathsf{K}_{n-1}}\Bigg(\left(\epsilon^{(1)} \to \tau_1^{(1)}, \sigma_{i_0^{(1)}}\right), \\ &\qquad\qquad \left(\tau_1^{(1)} \to \tau_2^{(1)}, \sigma_{i_1^{(1)}}\right), \cdots, \\ &\qquad\qquad \left(i_{n-1}^{(1)} \to \mathcal{A}, \sigma_{i_{n-1}^{(1)}}\right)\Bigg), \widehat{\sigma}_{i_{n-1}^{(1)}}\Bigg).\end{aligned} \tag{5.55}$$

Next, the adversary sends to user $i_{n+1}^{(0)}$ and $i_{n+1}^{(0)}$ the following tickets respectively.

$$\begin{aligned}
&\widehat{\tau}_{n+1}^{(0)} = \\
&\Bigg(\mathsf{AE.Enc}_{\mathsf{K}_n}\Bigg(\Big(\epsilon^{(0)} \to \tau_1^{(0)}, \sigma_{i_0^{(0)}}\Big), \\
&\qquad\qquad \Big(\tau_1^{(0)} \to \tau_2^{(0)}, \sigma_{i_1^{(0)}}\Big), \cdots, \\
&\qquad\qquad \Big(i_{n-1}^{(0)} \to \mathcal{A}, \sigma_{i_{n-1}^{(0)}}\Big), \\
&\qquad\qquad \Big(\mathcal{A} \to i_{n+1}^{(0)}, \sigma_{\mathcal{A}}\Big)\Bigg), \widehat{\sigma}_{\mathcal{A}}\Bigg), \\
&\widehat{\tau}_{n}^{(1)} = \\
&\Bigg(\mathsf{AE.Enc}_{\mathsf{K}_n}\Bigg(\Big(\epsilon^{(1)} \to \tau_1^{(1)}, \sigma_{i_0^{(1)}}\Big), \\
&\qquad\qquad \Big(\tau_1^{(1)} \to \tau_2^{(1)}, \sigma_{i_1^{(1)}}\Big), \cdots, \\
&\qquad\qquad \Big(i_{n-1}^{(1)} \to \mathcal{A}, \sigma_{i_{n-1}^{(1)}}\Big), \\
&\qquad\qquad \Big(\mathcal{A} \to i_{n+1}^{(1)}, \sigma_{\mathcal{A}}\Big)\Bigg), \widehat{\sigma}_{\mathcal{A}}\Bigg).
\end{aligned} \tag{5.56}$$

Then, the challenge user returns the following ticket to the adversary.

$$\begin{aligned}
&\widehat{\tau}_{n+k_0+1}^{(b)} = \\
&\Bigg(\mathsf{AE.Enc}_{\mathsf{K}_{n+k_0}}\Bigg(\Big(\epsilon^{(b)} \to \tau_1^{(b)}, \sigma_{i_0^{(b)}}\Big), \\
&\qquad\qquad \Big(\tau_1^{(b)} \to \tau_2^{(b)}, \sigma_{i_1^{(b)}}\Big), \cdots, \\
&\qquad\qquad \Big(i_{n-1}^{(b)} \to \mathcal{A}, \sigma_{i_{n-1}^{(b)}}\Big), \\
&\qquad\qquad \Big(\mathcal{A} \to i_{n+1}^{(b)}, \sigma_{\mathcal{A}}\Big), \cdots, \\
&\qquad\qquad \Big(i_{n+k_0}^{(b)} \to \mathcal{A}, \sigma_{i_{n+k_0}^{(b)}}\Big)\Bigg), \widehat{\sigma}_{i_{n+k_0}^{(b)}}\Bigg)
\end{aligned} \tag{5.57}$$

where $b \in \{0, 1\}$. For the adversary, the n transactions until the adversary receives and

signature parts of the returned ticket are hidden. Similar to the user anonymity and coin anonymity game, it is easy to see that the adversary who wins the IND-CCA game with such partially hidden messages can always win the standard full message IND-CCA game. Thus, given the anonymity property of the EPID signature, we can reduce IND-CCA of AE to the coin transparency game. This concludes the proof. □

Interestingly, in the coin transparency game, the adversary can not win the game even if the ticket is won among the challenge users. This is because the adversary does not issue the tickets and does not know the escrow accounts. Hence, the adversary has no clue to decide the group of challenge users even from the plain-text tickets registered in the blockchain.

Even if the adversary breaks the tamper-proof wallet, or if the winning ticket is registered in the blockchain and a series of transactions become known in plain text, the plain text reveals only temporary hashed values of (x, com) for each transaction and an anonymous EPID signature.

5.8.3 Double-spending attacks Detection Methods

For double-spending attacks, the attack can be detected perfectly. Furthermore, the adversary can not profit unless the cost of destroying the hardware exceeds the maximum expected utility that he can obtain.

To achieve both Fork and Collision detection methods, two or more thickets must be given, and the series of transactions from the escrow account must be referenceable.

Definition 21 (Conditions of Detection methods)**.** *Given two series of transactions τ and $\tilde{\tau}$. Fork and Collision Detection is achieved if and only if it satisfies the following conditions:*

1. *By τ and $\tilde{\tau}$, a series of transactions from the escrow account ϵ are referenceable.*
2. *If the attack is detected, the adversary's secret key is extracted and placed on the secret-key based revocation list. Thus, the adversary will not be able to transact with honest users.*

Theorem 14 (Fork and Collision Detection)**.** *Our anonymous transfer scheme achieves Fork and Collision detection scheme perfectly.*

Proof. Assume there exists forked ticket τ and $\tilde{\tau}$. Consider the fork detection. Let τ be the $\mathsf{eligible}$ ticket and registered in the blockchain, and let $\tilde{\tau}$ be the received ticket and stored in the wallet's $\mathsf{prog}_{\mathsf{w}}$. By comparing τ and $\tilde{\tau}$, the wallet can figure out the adversary's address from the forked point. Consider the Collision detection. When the wallet receives τ and $\tilde{\tau}$, it can extract the adversary's address by the fork detection and $\mathsf{KeyExtractor}$. □

Theorem 15. *Given two series of transactions τ and $\tilde{\tau}$ such that $(\tau, \tilde{\tau}) \in \mathsf{Fork}$, the double-spender's secret key can be extractable.*

Proof. Given the two transactions $\tau = \{\epsilon \prec \tau_1 \prec \cdots \prec \tau_n\}$ and $\tilde{\tau} = \{\tilde{\epsilon} \prec \tilde{\tau}_1 \prec \cdots \prec \tilde{\tau}_{n'}\}$ such as $(\tau, \tilde{\tau}) \in \mathsf{Fork}$ and Theorem 2, we can obtain the double-spending transactions. The EPID signatures of the double-spending transactions consist of the same $(x, \mathtt{com})$ and different $\mathtt{ch}, \mathtt{resp}$ respectively since the honest user's AESP chooses different rid with high probability, as follows:

$$\begin{cases} (x, \mathtt{com}, \mathtt{ch}, \mathtt{resp}) \\ \qquad = (\{B, K, T\}, \{R_1, R_2\}, c, \{s_x, s_f, s_a, s_b\}) \\ (x, \mathtt{com}, \mathtt{ch}', \mathtt{resp}') \\ \qquad = (\{B, K, T\}, \{R_1, R_2\}, c', \{s'_x, s'_f, s'_a, s'_b\}). \end{cases} \tag{5.58}$$

From Lemma 1, the double-spender's secret key can be extractable. □

Once the adversary's address is detected from the forked point, the wallet sends τ and $\tilde{\tau}$ to the EPID revocation manager $\mathcal{R}$. Then, $\mathcal{R}$ registers the adversary's secret key into the EPID's secret key based revocation list $\mathsf{Priv\text{-}RL}$.

5.9 Efficiency analysis

In this section, we present the size of each object in our proposed VeloCash and its comparison with Bauer et al. [BFQ21] in Table 5.1.

The setting for cyclic groups G_1, G_2 and G_3 and other elements are same as in Appendix A. Note that in Bauer et. [BFQ21] G_1, G_2 should be cyclic groups chosen that Symmetric External Diffie-Hellman assumption (SXDH) holds. On the other hand, in our VeloCash, G_1, G_2 should be the groups that q-Strong Diffie-Hellman (q-SDH) assumption holds, and G_3 should be the group that decisional Diffie-Hellman assumption holds.

Since our VeloCash is a decentralized scheme and there is no online and trusted party Bank as in [BFQ21], we still need to trust to some extent, the tamper-proof device manufacturer, which is also EPID's group manager. Therefore, we compare $\mathsf{sk}_{\mathcal{B}}$ and $\mathsf{pk}_{\mathcal{B}}$ with isk and gpk, respectively.

At first glance, the ticket size (coin size) of Bauer et al. [BFQ21] appears much larger than of VeloCash. This is because VeloCash is based on the stronger assumption (but widely used in industry), such as the existence of tamper-proof devices, whereas [BFQ21] is based purely on cryptographic assumptions.

	Bauer et al. [BFQ21]		VeloCash	
Keys	$\mathsf{sk}_{\mathcal{B}}$	$9\lvert\mathbb{Z}_p\rvert + 2\lvert G_1\rvert + 2\lvert G_2\rvert$	isk	$\lvert\mathbb{Z}_p^*\rvert$
	$\mathsf{pk}_{\mathcal{B}}$	$15\lvert G_1\rvert + 8\lvert G_2\rvert$	gpk	$3\lvert G_1\rvert + 2\lvert G_2\rvert + \lvert G_3\rvert$
	$\mathsf{sk}_{\mathcal{U}}$	$\lvert\mathbb{Z}_p\rvert + 2\lvert G_1\rvert + 2\lvert G_2\rvert$	$3\lvert\mathbb{Z}_p\rvert + \lvert G_1\rvert$	
	$\mathsf{pk}_{\mathcal{U}}$	$\lvert G_1\rvert$	0	
Tickets (Coins)	Π_{guilt}	$2\lvert G_1\rvert$	$\lvert\mathbb{Z}_p\rvert$	
	c_{bstrap}	$6\lvert\mathbb{Z}_p\rvert + 147\lvert G_1\rvert + 125\lvert G_2\rvert$	$5\lvert\mathbb{Z}_p\rvert + \lvert G_1\rvert + 2\lvert G_3\rvert$	
	c_{std}	$54\lvert G_1\rvert + 50\lvert G_2\rvert$	$7\lvert\mathbb{Z}_p\rvert + \lvert G_1\rvert + 2\lvert G_3\rvert$	

Table 5.1: A efficiency comparison between our VeloCash and Bauer et al. [BFQ21]. The coin's size is $\lvert c_{\text{bstrap}}\rvert + k\lvert c_{\text{std}}\rvert$ after k transfers.

5.10 Conclusion

In this chapter, we have proposed VeloCash, transferable decentralized probabilistic micropayments which preserve anonymity. For the construction of VeloCash, we utilized a tamper-proof wallet consisting of AESP. To achieve double-spending attack detection and preserve anonymity, we add extensions to AESP that allow for randomness tape and requesting EPID signatures from prog to $\mathcal{G}_{\text{att}}$.

As discussed in Section 5.8, VeloCash satisfies the u-an property only for the bounded number of challenge users. This pitfall stems from the fact that blockchain accounts are not blinded. Constructing probabilistic anonymous micropayments with transferability that satisfies the anonymity notion in full is left as an open problem.

Chapter 6

Conclusions

In this thesis, we have proposed solutions to three of the blockchain's biggest challenges: Fast payments, High Throughput and Anonymity with Transparency.

Secure Offline Payments in Bitcoin We have achieved a secure fast payments on Bitcoin. Double-spending attacks on fast payments are one of the fatal architectural problems in Cryptocurrencies. The prior study proposed an offline fast payment scheme that relies on tamper-proof wallets produced by trustworthy manufacturers. With the wallets, the payee can immediately trust the transactions generated by the wallets without waiting for their registration to the blockchain. The prior study required an online trusted timestamp server or, without one, usability was sacrificed. In contrast, our proposed method does not need any trusted timestamp server, nor does it sacrifices usability.

Decentralized Probabilistic Micropayments with Transferability We have proposed Decentralized Probabilistic Micropayments with Transferability that allows ticket distribution among users. As a result, the aggregation of the transactions increases the throughput of the blockchain, making it possible to realize micropayments. For example, a $1 payment can be made for 10 cent fee.

We have proposed a scheme with a tamper-proof assumption. By proposing a detection

method that perfectly detects the double-spending attacks and a detection method that places an upper bound on the expected value of the attack, we forced the adversary to weigh the cost of breaking the tamper-proof hardware against the maximum expected value gained from the attack.

VeloCash: Anonymous Decentralized Probabilistic Micropayments with Transferability We have achieved anonymity in the transferable decentralized probabilistic micropayments. Micropayments are one of the challenges in cryptocurrencies. Micropayments on the blockchain have the problem that the fee is high for the transfer amount. As a countermeasure, a method called Layer-two has been proposed to consolidate transactions outside the blockchain and improve the blockchain's throughput. As one of the existing Layer-two schemes, Decentralized Probabilistic Micropayments have been proposed. The winning amount is registered in the blockchain, and the lottery tickets are issued to be won with probability p, which allows us to aggregate approximately $1/p$ transactions into one. Unfortunately, existing solutions do not allow for ticket transferability, and the smaller p, the more difficult it is to use them in the real world. Here we have proposed VeloCash, Decentralized Probabilistic Micropayments with Transferability, which preserves Anonymity. By introducing tamper-proof assumptions for sending and receiving the tickets, we make p smaller. As a tamper-proof hardware assumption, VeloCash uses Attested Execution Secure Processors, a formal abstraction of secure processors with attested execution functionality and Direct Anonymous Attestation to achieve anonymity for sending and receiving tickets. VeloCash can detect double-spending attacks perfectly and revoke the adversary's device.

Appendix A

Construction of EPID

There are four types of entities in EPID: an issuer $\mathcal{I}$, a revocation manager $\mathcal{R}$, platformer $\mathcal{P}$, and verifiers $\mathcal{V}$. EPID consists of the five algorithms:

$$\Pi_{\mathsf{EPID}} = \{\mathsf{Setup}, \mathsf{Join}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Revoke}\}\,.$$

A.0.1 Setup

Given 1^k, issuer $\mathcal{I}$ chooses a group pair (G_1, G_2) of prime order p and let $e : G_1 \times G_2 \to G_T$ be a bilinear map function, and a cyclic group G_3 of order p where the decisional Diffie-Hellman problem is hard. Let g_1, g_2, g_3 be the generators of G_1, G_2, G_3 respectively. $\mathcal{I}$ chooses $h_1, h_2 \leftarrow G_1, \gamma \leftarrow \mathbb{Z}_p^*$, and $\omega := g_2^{\gamma}$. This algorithm outputs

$$(\mathsf{gpk}, \mathsf{isk}) = ((p, G_1, G_2, G_3, g_1, g_2, g_3, h_1, h_2, w), \gamma)\,.$$

A.0.2 Join

The Join protocol is performed interactively between issuer $\mathcal{I}$ and platformer $\mathcal{P}$. The flow is described in Figure A.1.

$\mathcal{I}$ has $(\mathsf{gpk}, \mathsf{isk})$ and $\mathcal{P}$ takes gpk. Finally, $\mathcal{P}$ obtains $\mathsf{sk} = ((A, x, y), f)$ where f is a

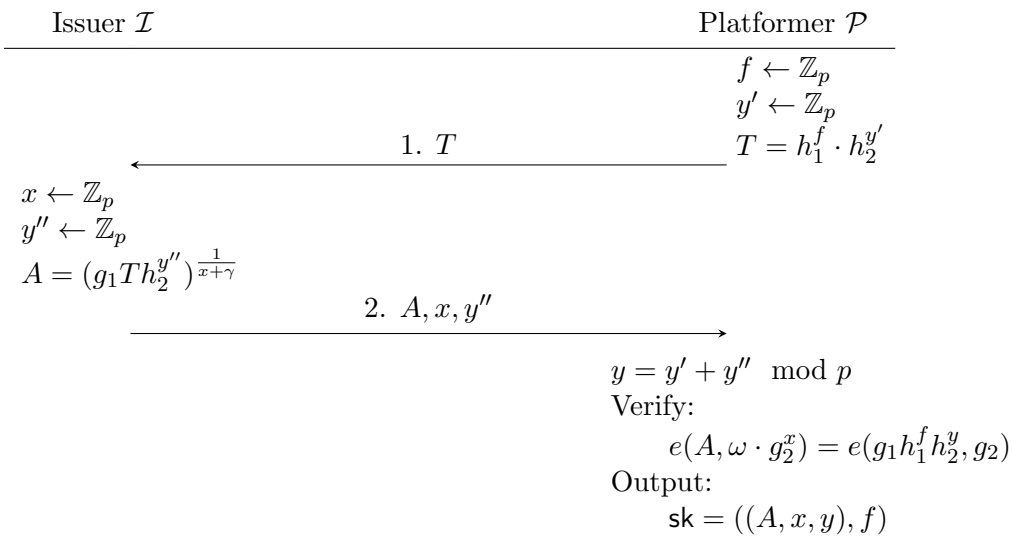


Figure A.1: EPID's Join protocol

unique membership key and (A, x, y) is a BBS+ signature [ASM06] on f.

A.0.3 Sign

Platformer $\mathcal{P}$ inputs $\mathsf{gpk}, \mathsf{sk} := ((A, x, y), f), m \in \{0,1\}^*$, and a signature based revocation list Sig-RL, then outputs the signature σ as follows:

1. Chooses $B \leftarrow G_3$ such that $B \neq 1$ and computes

$$K := B^f \tag{A.1}$$

2. Chooses $a \leftarrow \mathbb{Z}_p$ and computes

$$b := y + ax,\ T := A \cdot h_2^a \tag{A.2}$$

3. It randomly picks

$$r_x \leftarrow \mathbb{Z}_p,\ r_f \leftarrow \mathbb{Z}_p,\ r_a \leftarrow \mathbb{Z}_p,\ r_b \leftarrow \mathbb{Z}_p \tag{A.3}$$

4. Computes

$$\begin{aligned} R_1 &:= B^{r_f} \\ R_2 &:= e\left(T, g_2\right)^{-r_x} \cdot e\left(h_1, g_2\right)^{r_f} \cdot e\left(h_2, g_2\right)^{r_b} \cdot e\left(h_2, \omega\right)^{r_a} \end{aligned} \tag{A.4}$$

5. Computes

$$c = \mathsf{Hash}(\mathsf{gpk}, B, K, T, R_1, R_2, m) \tag{A.5}$$

6. Computes

$$\begin{aligned} s_x &= r_x + c \cdot x, \;\; s_f = r_f + c \cdot f, \\ s_a &= r_a + c \cdot a, \;\; s_b = r_b + c \cdot b \end{aligned} \tag{A.6}$$

7. Sets $\sigma_0 := (B, K, T, c, s_x, s_f, s_a, s_b)$

Then, $\mathcal{P}$ verifies the above output values satisfy the following signature of knowledge protocol as follows:

$$\begin{aligned} \mathrm{SPK}\big\{(x, f, a, b) : B^f = K \wedge \\ e\left(T, g_2\right)^{-x} \cdot e\left(h_1, g_2\right)^{f} \cdot e\left(h_2, g_2\right)^{b} \cdot e\left(h_2, w\right)^{a} \\ = e(T, \omega) / e(g_1, g_2)\big\}(m) \end{aligned} \tag{A.7}$$

A.0.4 Verify

Verifier $\mathcal{V}$ inputs gpk, m, a secret key based revocation list $\mathsf{Priv\text{-}RL}$, a signature-key based revocation list $\mathsf{Sig\text{-}RL}$, and a signature σ, then verifies the signature as follows:

1. Let $\sigma = (\sigma_0, \sigma_1, \ldots, \sigma_{n_2})$,
   where $\sigma_0 = (B, K, T, c, s_x, s_f, s_a, s_b)$

2. Verifies

$$\begin{aligned} B, K &\stackrel{?}{\in} G_3 \text{ and } B \neq 1, \\ T &\stackrel{?}{\in} G_1, \quad s_x, s_f, s_a, s_b \stackrel{?}{\in} \mathbb{Z}_p \end{aligned} \tag{A.8}$$

3. Computes

$$\hat{R}_1 = B^{s_f} \cdot K^{-c}$$
$$\hat{R}_2 = e\left(T, g_2\right)^{-s_x} \cdot e\left(h_1, g_2\right)^{s_f} \cdot e\left(h_2, g_2\right)^{s_b} \cdot e\left(h_2, \omega\right)^{s_a} \cdot \left(e(g_1, g_2)/e(T, \omega)\right)^{c} \tag{A.9}$$

4. Verifies

$$c \stackrel{?}{=} \mathsf{H}\left(\mathsf{gpk}, B, K, T, \hat{R}_1, \hat{R}_2, m\right) \tag{A.10}$$

5. Let $\mathsf{Priv\text{-}RL} = \{f_1, \ldots, f_{n_1}\}$. For $i = 1, \ldots, n_1$, it checks that $K \stackrel{?}{\neq} B^{f_i}$

6. Let $\mathsf{Sig\text{-}RL} = \{(B_1, K_1), \ldots, (B_{n_2}, K_{n_2})\}$. For $i = 1, \ldots, n_2$, it verifies that σ_i is a valid zero-knowledge proof,

$$\mathrm{SPK}\left\{(f) : K = B^f \wedge K_i \neq B_i^f\right\}(m). \tag{A.11}$$

A.0.5 Revoke

Secret key based revocation

Given $\mathsf{gpk}, \mathsf{Priv\text{-}RL}$, and $\mathsf{sk} = (A, x, y, f)$ to be revoked, revocation manager $\mathcal{R}$ updates $\mathsf{Priv\text{-}RL}$ as follows:

1. verify the correctness of sk,

$$e(A, g_2^x \omega) = e(g_1 h_1^f h_2^y, g_2) \tag{A.12}$$

2. appends f in σ_0 to $\mathsf{Sig\text{-}RL}$

Signature based revocation

Given $\mathsf{gpk}, \mathsf{Priv\text{-}RL}, \mathsf{Sig\text{-}RL}, m$, and σ to be revoked, revocation manager $\mathcal{R}$ updates $\mathsf{Sig\text{-}RL}$ as follows:

1. checks

$$\mathsf{Verify}(\mathsf{gpk}, m, \mathsf{Priv\text{-}RL}, \emptyset, \sigma_0) = \mathsf{valid} \tag{A.13}$$

2. appends (B, K) in σ_0 to $\mathsf{Sig\text{-}RL}$

Appendix B

Security Definition of EPID

An EPID scheme is secure if it satisfies the following three requirements: correctness, anonymity, unforgeability [BCC04, BL10].

The correctness requires that every signature generated by a platform is valid except when the platform is revoked by the secret key based revocation or the signature based revocation.

Definition 22. *(Correctness) An EPID scheme is correct, for every probabilistic polynomial-time adversary $\mathcal{A}$, if it satisfies the following equation:*

$$\begin{cases} \sigma \leftarrow \mathsf{Sign}(\mathsf{gpk}, \mathsf{sk}, m, \mathsf{Sig\text{-}RL}), \\ \mathsf{Verify}\,(\mathsf{gpk}, m, \mathsf{Priv\text{-}RL}, \mathsf{Sig\text{-}RL}, \sigma) = \mathsf{valid} \end{cases} \tag{B.1}$$
$$\iff (\mathsf{sk}_i \notin \mathsf{Priv\text{-}RL}) \wedge (\textstyle\sum_i \cap\ \mathsf{Sig\text{-}RL} = \emptyset)$$

where $\sum_i$ is the set of all signatures generated by the platform $\mathcal{P}_i$.

Theorem 16. *(Theorem 4 of [BL09]) The EPID scheme is correct.*

In the anonymity game, the adversary's goal is to determine which one of two secret keys were used in generating the signature. The anonymity game between a challenger and an adversary $\mathcal{A}$ is described in Figure B.1.

Definition 23. *(Anonymous) An EPID scheme is anonymous, if for every probabilistic polynomial-time adversary $\mathcal{A}$, the advantage in winning the anonymity game between a challenger is negligible as follows:*

$$\begin{aligned} &\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Anon}}(\lambda) = \\ &\quad \left|2 \cdot \Pr\left[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Anon}}(\lambda) = 1\right] - 1\right| < \mathsf{negl}(\lambda) \end{aligned} \tag{B.2}$$

Theorem 17. *(Theorem 5 of [BL09]) An EPID scheme is anonymous in the random oracle model under the decisional Diffie-Hellman assumption in G_3.*

Note that the adversary $\mathcal{A}$ can not make corrupt queries on the challenge users i_0 and i_1.

We say that the EPID scheme is unforgeable if no adversary can win the unforgeability game described in Figure B.1. In the unforgeability game, the adversary's goal is to forge a valid signature, given that all secret keys known to the adversary have been revoked.

Definition 24. *(Unforgeability) An EPID scheme is unforgeable, if for every probabilistic polynomial-time adversary $\mathcal{A}$, the advantage in winning the unforgeability game between a challenger is negligible as follows:*

$$\begin{aligned} &\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Unforg}}(\lambda) = \\ &\quad \Pr\left[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{Unforg}}(\lambda) = 1\right] < \mathsf{negl}(\lambda) \end{aligned} \tag{B.3}$$

Theorem 18. *(Theorem 6 of [BL09]) The EPID scheme is unforgeable in the random oracle model under the strong Diffie-Hellman assumption in (G_1, G_2).*

Experiment: $\mathrm{Expt}^{\mathsf{Anon}}_{\mathcal{A}}(\lambda)$

1 : $(\mathsf{gpk}, \mathsf{isk}) \leftarrow \mathcal{A}(1^\lambda)$
2 : $(\mathtt{state}, i_0, i_1, m)$
$\quad \leftarrow \mathcal{A}^{\mathsf{O}_{\mathsf{Issue}}(\cdot),\mathsf{O}_{\mathsf{Join}}(\cdot),\mathsf{O}_{\mathsf{Sign}}(\mathsf{gpk},\mathsf{Sig\text{-}RL},\cdot,\cdot),\mathsf{O}_{\mathsf{Corrupt}}(\cdot)}(\mathsf{gpk})$
3 : $b \leftarrow\!\!\$\ \{0,1\}$
4 : $\sigma \leftarrow \mathsf{Sign}(\mathsf{gpk}, \mathsf{sk}_{i_b}, m, \mathsf{Sig\text{-}RL})$
5 : $b' \leftarrow \mathcal{A}^{\mathsf{O}_{\mathsf{Join}}(\cdot),\mathsf{O}_{\mathsf{Sign}}(\mathsf{gpk},\mathsf{Sig\text{-}RL},\cdot,\cdot),\mathsf{O}_{\mathsf{Corrupt}}(\cdot)}(\mathtt{state}, \sigma)$
6 : **if** $b = b'$, return 1
7 : **else** return 0

Experiment: $\mathrm{Expt}^{\mathsf{Unforg}}_{\mathcal{A}}(\lambda)$

1 : $(\mathsf{gpk}, \mathsf{isk}) \leftarrow \mathsf{Setup}(1^\lambda)$
2 : $(\mathsf{Priv\text{-}RL}^*, \mathsf{Sig\text{-}RL}^*, \sigma^*, m^*)$
$\quad \leftarrow \mathcal{A}^{\mathsf{O}_{\mathsf{Issue}}(\cdot),\mathsf{O}_{\mathsf{Join}}(\cdot),\mathsf{O}_{\mathsf{Sign}}(\mathsf{gpk},\mathsf{Sig\text{-}RL},\cdot,\cdot),\mathsf{O}_{\mathsf{Corrupt}}(\cdot)}(\mathsf{gpk})$
3 : return 1 **if** all of the following conditions hold:
$\quad$ 1) $\mathsf{Verify}(\mathsf{gpk}, \mathsf{Priv\text{-}RL}^*, \mathsf{Sig\text{-}RL}^*, \sigma^*, m^*) = 1$,
$\quad \wedge$ 2) ${}^{\forall} i \in U,\ \mathsf{sk}_i \in \mathsf{Priv\text{-}RL}^*$ or ${}^{\exists}\sigma_i \in \mathsf{Sig\text{-}RL}^*$,
$\quad \wedge$ 3) $\mathcal{A}$ did not obtain σ^* by making a sign query on m^*
4 : **else** return 0

Oracle: $\mathsf{O}_{\mathsf{Join}}(i)$

1 : **if** $i \notin U$
2 : $\quad U \leftarrow U \cup \{i\}$
3 : $\quad$ generate sk_i
4 : $\quad$ return 1
5 : **else** return $\perp$

Oracle: $\mathsf{O}_{\mathsf{Sign}}(\mathsf{gpk}, \mathsf{Sig\text{-}RL}, i, m)$

1 : **if** $i \in U$
2 : $\quad \sigma \leftarrow \mathsf{Sign}(\mathsf{gpk}, \mathsf{sk}_i, m, \mathsf{Sig\text{-}RL})$
3 : $\quad$ return σ
4 : **else** return $\perp$

Oracle: $\mathsf{O}_{\mathsf{Corrupt}}(i)$

1 : **if** $i \in U$
2 : $\quad$ return sk_i
3 : **else** return $\perp$

Figure B.1: EPID game for *Anonymity* and *Unforgeability*

Appendix C

Proof of Theorem 9

To prove exculpability property stated in Theorem 9, we assume there exists a PPT adversary $\mathcal{A}_{\mathsf{ex}}$ that wins the exculpability game defined in Definition 15 with non-negligible probability, namely, that outputs $(\mathsf{id}, \pi, \Pi_G)$ such that $\mathsf{VerifyGuilt}((\mathsf{id}, \pi), \Pi_G) = 1$.

We will show a reduction that we can construct a PPT adversary $\mathcal{A}$ which wins the EPID anonymity game defined in Definition 23 with non-negligible probability provided oracle access to $\mathcal{A}_{\mathsf{ex}}$ is available.

In order to share the same user list between $\mathcal{A}$ and $\mathcal{A}_{\mathsf{ex}}$, we will not allow $\mathcal{A}_{\mathsf{ex}}$ access to Join; instead, $\mathcal{A}$ provide a sufficiently long list of users $(i_1, \ldots, i_n)$ created by $\mathcal{A}$ and provide the list of users to $\mathcal{A}_{\mathsf{ex}}$. Note that this modification to the exculpability game does not affect the success probability of $\mathcal{A}_{\mathsf{ex}}$.

In the reduction, $\mathcal{A}$ generates the parameter param and $\mathsf{sk}_{\mathcal{B}}$, and it joins n users and creates the list of users $(i_1, \ldots, i_n)$. Then, $\mathcal{A}_{\mathsf{ex}}$ is executed with input $(\mathsf{param}, \mathsf{sk}_{\mathcal{B}}, (i_1, \ldots, i_n))$,

and it outputs $(\mathsf{id}, \pi, \Pi_G)$ with non-negligible probability as follows:

$$\mathsf{param} \leftarrow \mathsf{ParamGen}(1^\lambda);\ \mathsf{sk}_{\mathcal{B}} \leftarrow \mathcal{A}(\mathsf{param})$$
$$(i_1, \cdots, i_n) \leftarrow \mathcal{A}^{\mathsf{O}_{\mathsf{Join}}(\cdot)}$$
$$(\mathsf{id}, \pi, \Pi_G) \leftarrow \mathcal{A}_{\mathsf{ex}}^{\mathtt{Spy},\mathtt{UWith},\mathtt{Rcv},\mathtt{S\&R},\mathtt{Depo}}(\mathsf{param}, \mathsf{sk}_{\mathcal{B}}, (i_1, \ldots, i_n))$$
$$\text{such that } \mathsf{VerifyGuilt}((\mathsf{id}, \pi), \Pi_G) = 1$$

Note that the accused double-spender i^* is contained in the user list $(i_1, \ldots, i_n)$. Otherwise, given $\mathsf{VerifyGuilt}((\mathsf{id}, \pi), \Pi_G) = 1$, $\mathcal{A}_{\mathsf{ex}}$ must win the unforgeability game in Figure 5.2, and it contradicts with Theorem 8.

Then, we can easily find out the user i^* who created the signature (id, π) by iteratively ask all honest users to produce signatures for a message in a form of randomly chosen commitment, namely, $m = (x, \mathsf{com})$. That is, for $i^* \in \{i_1, \cdots, i_n\}$, $\mathcal{A}$ queries to $\mathsf{O}_{\mathsf{Sign}}(\mathsf{gpk}, \mathsf{sk}_{i^*}, m, \mathsf{Sig\text{-}RL})$ and gets σ_{i_j}. By Lemma 2, there must exist $i^* \in (i_1, \ldots, i_n)$ such that $\mathsf{VerifyGuilt}\,((m, \sigma_{i^*}), \Pi_G) = 1$. Then, $\mathcal{A}$ randomly chooses $\bar{i^*} \in \{i_1, \ldots, i_n\} \setminus \{i^*\}$. $\mathcal{A}$ outputs $(\Pi_G, i^*, \bar{i^*}, m)$.

At step 5 in the Anonymity Game defined in Figure 5.3, $\mathcal{A}$ outputs $b' = 0$ if

$$\mathsf{VerifyGuilt}\,((m, \sigma), \Pi_G) = 1 \tag{C.1}$$

otherwise $b' = 1$.

By Lemma 2, the adversary $\mathcal{A}$ wins the Anonymity Game with non-negligible probability if there exists $\mathcal{A}_{\mathsf{ex}}$ which wins the Excupability Game defined in Figure 5.3 with non-negligible probability. This contradicts the anonymity property of EPID stated in Theorem 5, and this reduction is tight. Therefore, there is no $\mathcal{A}_{\mathsf{ex}}$ that wins the exculpability game with non-negligible probability. □

Bibliography

[ABC20] Ghada Almashaqbeh, Allison Bishop, and Justin Cappos. MicroCash: Practical Concurrent Processing of Micropayments. In *Financial Cryptography and Data Security - FC 2020*, Lecture Notes in Computer Science, pages 227–244. Springer International Publishing, 2020.

[ASM06] Man Ho Au, Willy Susilo, and Yi Mu. Constant-Size Dynamic k-TAA. In *Security and Cryptography for Networks*, Lecture Notes in Computer Science, pages 111–125, Berlin, Heidelberg, 2006. Springer.

[BBBF18] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable Delay Functions. In *Advances in Cryptology – CRYPTO 2018*, Lecture Notes in Computer Science, pages 757–788. Springer International Publishing, 2018.

[BCC04] Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *Proceedings of the 11th ACM conference on Computer and communications security*, CCS '04, pages 132–145. Association for Computing Machinery, October 2004.

[BCFK15] Foteini Baldimtsi, Melissa Chase, Georg Fuchsbauer, and Markulf Kohlweiss. Anonymous Transferable E-Cash. In *Public-Key Cryptography – PKC 2015*, Lecture Notes in Computer Science, pages 101–124, Berlin, Heidelberg, 2015. Springer.

[BCL09] Ernie Brickell, Liqun Chen, and Jiangtao Li. Simplified security notions of direct anonymous attestation and a concrete scheme from pairings. *International Journal of Information Security*, 8(5):315–330, 2009.

[BFQ21] Balthazar Bauer, Georg Fuchsbauer, and Chen Qian. Transferable E-Cash: A Cleaner Model and the First Practical Instantiation. In *Public-Key Cryptography – PKC 2021*, Lecture Notes in Computer Science, pages 559–590. Springer International Publishing, 2021.

[BL09] Ernie Brickell and Jiangtao Li. Enhanced Privacy ID from Bilinear Pairing. Technical Report 095, 2009. http://eprint.iacr.org/2009/095.

[BL10] Ernie Brickell and Jiangtao Li. Enhanced Privacy ID from Bilinear Pairing for Hardware Authentication and Attestation. In *2010 IEEE Second International Conference on Social Computing*, 2010.

[BL12] Ernie Brickell and Jiangtao Li. Enhanced Privacy ID: A Direct Anonymous Attestation Scheme with Enhanced Revocation Capabilities. *IEEE Transactions on Dependable and Secure Computing*, 2012.

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: formal definitions, simplified requirements, and a construction based on general assumptions. In *Proceedings of the 22nd international conference on Theory and applications of cryptographic techniques*, EUROCRYPT'03, pages 614–629, Berlin, Heidelberg, May 2003. Springer-Verlag.

[BN08] Mihir Bellare and Chanathip Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. *Journal of Cryptology*, 21(4):469–491, October 2008.

[BSCG+14] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized Anonymous

Payments from Bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474, May 2014. ISSN: 2375-1207.

[CG08] Sébastien Canard and Aline Gouget. Anonymity in Transferable E-cash. In *Applied Cryptography and Network Security*, volume 5037, pages 207–223, Berlin, Heidelberg, 2008. Springer. Series Title: Lecture Notes in Computer Science.

[CGL+17] Alessandro Chiesa, Matthew Green, Jingcheng Liu, Peihan Miao, Ian Miers, and Pratyush Mishra. Decentralized Anonymous Micropayments. In *Advances in Cryptology – EUROCRYPT 2017*, Lecture Notes in Computer Science, pages 609–642. Springer, 2017.

[CGT08] Sébastien Canard, Aline Gouget, and Jacques Traoré. Improvement of Efficiency in (Unconditional) Anonymous Transferable E-Cash. In *Financial Cryptography and Data Security*, volume 5143, pages 202–214, Berlin, Heidelberg, 2008. Springer. ISSN: 0302-9743, 1611-3349 Series Title: Lecture Notes in Computer Science.

[Che10] Liqun Chen. A DAA Scheme Using Batch Proof and Verification. In *Trust and Trustworthy Computing*, Lecture Notes in Computer Science, pages 166–180, Berlin, Heidelberg, 2010. Springer.

[Cora] Intel Corporation. A Cost-Effective Foundation for End-to-End IoT Security. `https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/intel-epid-white-paper.pdf`. (accessed February 24th, 2022).

[Corb] Intel Corporation. Intel Enhanced Privacy ID (EPID) Security Technology. `https://www.intel.com/content/www/us/en/developer/articles/technical/intel-enhanced-privacy-id-epid-security-technology.html`. (accessed February 24th, 2022).

[CP93] David Chaum and Torben Pryds Pedersen. Transferred Cash Grows in Size. In *Advances in Cryptology — EUROCRYPT'92*, Lecture Notes in Computer Science, pages 390–407, Berlin, Heidelberg, 1993. Springer.

[CPS10] Liqun Chen, Dan Page, and Nigel P. Smart. On the Design and Implementation of an Efficient DAA Scheme. In *Smart Card Research and Advanced Application*, Lecture Notes in Computer Science, pages 223–237. Springer, 2010.

[CZK+19] Raymond Cheng, Fan Zhang, Jernej Kos, Warren He, Nicholas Hynes, Noah Johnson, Ari Juels, Andrew Miller, and Dawn Song. Ekiden: A Platform for Confidentiality-Preserving, Trustworthy, and Performant Smart Contracts. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 185–200, June 2019.

[Dam99] Ivan Damgård. Commitment Schemes and Zero-Knowledge Protocols. In *Lectures on Data Security, Modern Cryptology in Theory and Practice, Summer School, Aarhus, Denmark, July 1998*, pages 63–86, Berlin, Heidelberg, 1999. Springer-Verlag.

[DNY17] Alexandra Dmitrienko, David Noack, and Moti Yung. Secure Wallet-Assisted Offline Bitcoin Payments with Double-Spender Revocation. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17. Association for Computing Machinery, 2017.

[DPS19] Phil Daian, Rafael Pass, and Elaine Shi. Snow White: Robustly Reconfigurable Consensus and Applications to Provably Secure Proof of Stake. In *Financial Cryptography and Data Security*, Lecture Notes in Computer Science, pages 23–41. Springer International Publishing, 2019.

[Fis05] Marc Fischlin. Communication-Efficient Non-interactive Proofs of Knowledge with Online Extractors. In *Advances in Cryptology – CRYPTO 2005*, Lecture Notes in Computer Science, pages 152–168, Berlin, Heidelberg, 2005. Springer.

[FS87] Amos Fiat and Adi Shamir. How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology — CRYPTO' 86*, Lecture Notes in Computer Science, pages 186–194, Berlin, Heidelberg, 1987. Springer.

[FS07] Eiichiro Fujisaki and Koutarou Suzuki. Traceable Ring Signature. In *Public Key Cryptography – PKC 2007*, Lecture Notes in Computer Science, pages 181–200, Berlin, Heidelberg, 2007. Springer.

[Fuj11] Eiichiro Fujisaki. Sub-linear Size Traceable Ring Signatures without Random Oracles. In *Topics in Cryptology – CT-RSA 2011*, Lecture Notes in Computer Science, pages 393–415, Berlin, Heidelberg, 2011. Springer.

[GHS11] Jie Guo, Lin Hao, and Huimin Sun. A new DAA scheme from one-off public key. In *2011 International Conference on Electronics, Communications and Control (ICECC)*, pages 646–649, 2011.

[GKL15] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The Bitcoin Backbone Protocol: Analysis and Applications. In *Advances in Cryptology - EUROCRYPT 2015*, Lecture Notes in Computer Science, pages 281–310, Berlin, Heidelberg, 2015. Springer.

[GMSR+20] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. SoK: Layer-Two Blockchain Protocols. In *Financial Cryptography and Data Security*, Lecture Notes in Computer Science, pages 201–226. Springer International Publishing, 2020.

[KAC12] Ghassan O. Karame, Elli Androulaki, and Srdjan Capkun. Double-spending fast payments in bitcoin. In *Proceedings of the 2012 ACM conference on Computer and communications security - CCS 2012*, pages 906–917, New York, NY, USA, 2012. Association for Computing Machinery.

[KRDO17] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In *Advances in Cryptology – CRYPTO 2017*, Lecture Notes in Computer Science, pages 357–388. Springer International Publishing, 2017.

[LNE+19] Joshua Lind, Oded Naor, Ittay Eyal, Florian Kelbert, Emin Gün Sirer, and Peter Pietzuch. Teechain: a secure payment network with asynchronous blockchain access. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, SOSP '19, pages 63–79, New York, NY, USA, 2019. Association for Computing Machinery.

[MBB+19] Andrew Miller, Iddo Bentov, Surya Bakshi, Ranjit Kumaresan, and Patrick McCorry. Sprites and State Channels: Payment Networks that Go Faster Than Lightning. In *Financial Cryptography and Data Security*, Lecture Notes in Computer Science, pages 508–526. Springer International Publishing, 2019.

[Nak09] Satoshi Nakamoto. Bitcoin : A peer-to-peer electronic cash system. 2009.

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

[PS96] David Pointcheval and Jacques Stern. Security Proofs for Signature Schemes. In *Advances in Cryptology — EUROCRYPT '96*, Lecture Notes in Computer Science, pages 387–398, Berlin, Heidelberg, 1996. Springer.

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

[PSS17] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the Blockchain Protocol in Asynchronous Networks. In *Advances in Cryptology – EUROCRYPT 2017*, Lecture Notes in Computer Science, pages 643–673, Cham, 2017. Springer International Publishing.

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

[Rab83] Michael O. Rabin. Transaction protection by beacons. *Journal of Computer and System Sciences*, 27(2):256–267, October 1983.

[Riv97] Ronald L. Rivest. Electronic lottery tickets as micropayments. In *Financial Cryptography*, Lecture Notes in Computer Science, pages 307–314, Berlin, Heidelberg, 1997. Springer.

[SRC15] Ben Smyth, Mark D. Ryan, and Liqun Chen. Formal analysis of privacy in Direct Anonymous Attestation schemes. *Science of Computer Programming*, 111:300–317, 2015.

[THO22] Taisei Takahashi, Taishi Higuchi, and Akira Otsuka. VeloCash: Anonymous Decentralized Probabilistic Micropayments with Transferability. *IEEE Access*, 2022.

[TO20] Taisei Takahashi and Akira Otsuka. Short Paper: Secure Offline Payments in Bitcoin. In *Workshop on Trusted Smart Contracts In Association with Fi-*

nancial Cryptography and Data Security, Lecture Notes in Computer Science, pages 12–20, Cham, 2020. Springer International Publishing.

[TO21] Taisei Takahashi and Akira Otsuka. Probabilistic Micropayments with Transferability. In *Computer Security – ESORICS 2021*, Lecture Notes in Computer Science, pages 390–406, Cham, 2021. Springer International Publishing.

[Whe97] David Wheeler. Transactions using bets. In *Security Protocols*, Lecture Notes in Computer Science, pages 89–92, Berlin, Heidelberg, 1997. Springer.

[Woo] Dr Gavin Wood. ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER. `https://gavwood.com/paper.pdf`.

[YCH] YCHARTS. Bitcoin Average Transaction Fee. `https://ycharts.com/indicators/bitcoin_average_transaction_fee`. (accessed April 5th, 2022).