PRACTICAL ESCROW PROTOCOL FOR CRYPTOCURRENCIES

YANG XIAO

PhD

The Hong Kong Polytechnic University

2022

The Hong Kong Polytechnic University
Department of Computing

Practical Escrow Protocol for Cryptocurrencies

Yang Xiao

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

August 2021

CERTIFICATE OF ORIGINALITY

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

__________________________________ (Signed)

____________YANG XIAO_______(Name of student)

Abstract

Escrow protocol for cryptocurrencies is a two-party protocol that enables the fair exchange of goods or services with digital coins. An escrow protocol allows digital coins to be escrowed (i.e., locked) in a way that only the seller can claim the coins (when the deal is completed) or the buyer can claim the coin (when the deal is canceled) with the help of a trusted third party (TTP).

Existing escrow protocols for cryptocurrencies are built based on various approaches with various security and efficiency trade-offs. In this thesis, we introduce a new approach based on verifiably encrypted signature (VES), a specific kind of digital signature whose validity can be verified in encrypted form. Escrow protocols constructed from our approach enjoy many desirable features, including (a) round-efficient; (b) privacy-preserving for participants; and (c) minimal TTP involvement.

ECDSA is the signature scheme adopted by major cryptocurrencies such as Bitcoin and Ethereum. To construct escrow protocols for these cryptocurrencies based on our approach, we develop an efficient verifiably encrypted ECDSA, which may be of independent interest. Besides ECDSA, EdDSA and Schnorr digital signatures are adopted in popular cryptocurrencies. To build escrow protocols to fit these popular cryptocurrencies, we generalize the above signature schemes as an EdDSA-like signature and propose a generic construction of verifiably encrypted signature scheme for EdDSA-like signature.

We conduct a thorough complexity analysis of the escrow protocol obtained from the above VES schemes and demonstrate its feasibility.

Publications

The following papers and manuscripts are based on the results of this thesis.

- **Xiao Yang**, Wang Fat Lau, Qingqing Ye, Man Ho Au, Joseph K. Liu and Jacob Cheng. Practical Escrow Protocol for Bitcoin. IEEE Transactions on Information Forensics and Security 15:3023 - 3034 (2020)
- **Xiao Yang**, Mengling Liu, Man Ho Au and Xiapu Luo. Generic Escrow Protocol for Cryptocurrencies. IEEE Transactions on Information Forensics and Security (under review).

Acknowledgments

First and foremost I would like to express my sincere appreciation to Prof. Man Ho Au, Prof. Luo Xiapu Daniel, and Dr. Liu Yan Wang Dennis, for offering me the unique opportunity to pursue cryptography research and their continued support during my PhD study. In particular, I would like to thank Prof. Man Ho Au, who for many years, steered me through this research with continuous guidance, illuminating discussions, and considerate advice. He has greatly inspired me with his rigorous scientific approach, dedicating spirit for work, and strong sense of responsibility. Without his support and patience, I would not have completed this thesis.

I am deeply thankful to the joint team of researchers at Hong Kong Polytechnic University and Hong Kong University, for their support and friendship, including Jingjing Fan, Borui Gong, Peng Jiang, Wang Fat Lau, Kang Li, Xinyu Li, Mengling Liu, Jiazhuo Lv, Xingye Lu, Yilei Wang, Dongqing Xu, Haiyang Xue, Rupeng Yang, Xiaoyi Yang, and Zuoxia Yu. I am also grateful to the supporting staff Anna and Karina for all the help. I wish to take this opportunity to thank all the co-authors of all my research papers, especially Wang Fat Lau and Mengling Liu.

I wish to thank Prof. Zhe Xia at the Wuhan University of Technology for his encouragement and help in this journey.

I would like to thank Mrs. Cammy Wu and Mr. Ka Wai Lee for their accompany in the past few years.

Finally, but not least, I would like to express my sincere gratitude to my grandma, grandpa, my parents, my sister, my brother, my nephew, and all my family members and friends for their constant love and support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

With the growing popularity of cryptocurrency, digital payment platforms supporting cryptocurrency, e.g., Coinbase Commerce [28], Electroneum [19], and cryptocurrency spot exchanges, e.g., Binance [6], Huobi Global [32], emerged rapidly in the past decade. Digital payment platforms offer a mechanism for the exchange of digital coins with goods or services. Typically, these platforms serve as a trusted intermediary, receiving digital coins from the buyer; withholding these coins; and releasing them to the seller when the deal is completed. Cryptocurrency spot exchanges, on the other hand, enable customers to trade various types of digital currencies. Conceptually, a cryptocurrency spot exchange can be viewed as a digital payment platform where a kind of digital coins is used to buy digital coins of another kind.

The platform in the aforementioned scenario has to be fully trusted. However, in some scenarios, it is difficult, if not impossible, to find an entity trusted by both the buyer and the seller. Furthermore, as large amounts of assets are being held, these platforms become the primary targets of cryptocurrency hacks and breaches. For instance, one of the largest Bitcoin exchange platforms, Bitfinex, suffered from the so-called Bitfinex event in 2016 and lost 119,756 units of bitcoin, causing bitcoin's trading price to drop by 20% [52]. In 2020 alone, these hacks cause millions of dollars worth of loss [55].

To reduce the trust placed on these platforms, Goldfeder et al. [24] formalized the notion of escrow protocol for cryptocurrencies. In an escrow protocol, digital coins from the buyer are first "locked" (escrowed) in a way that they can only be transferred to the seller or returned to the buyer eventually. In other words, the escrowed coins cannot be stolen even if the platform is compromised. Following the terminologies in [24, 54], an escrow protocol for cryptocurrencies should possess four desirable properties, namely, security, privacy, minimal TTP involvement, and efficiency. Intuitively, an escrow protocol should be secure, meaning that the escrowed funds can only be transferred to the buyer or seller. An escrow protocol offers privacy if a transaction using the escrow protocol is indistinguishable from a standard transaction (either by the platform or an external observer). In addition, privacy is also concerned with whether or not the participating parties can be recognised. In terms of TTP involvement, an escrow protocol should only involve the buyer and seller, and the TTP is called upon for arbitration when there is a dispute between them. Such a model would be more scalable in practice since the TTP is idle most of the time assuming most of the deals are completed without dispute. Finally, the escrow protocol should be efficient.

It is worth noting that privacy receives relatively less attention from academia. While escrow protocols offering privacy protection exist, they remain theoretical in nature due to the high computation costs in the use of generic zero-knowledge proofs [24] or involve many rounds of interactions due to the use of cut-and-choose mechanism [4].

1.2 Summary of the Thesis

In this thesis, we investigate the design of escrow protocols of cryptocurrencies achieving the above four desirable properties. We focus on *blockchain*-based cryptocurrencies [53], e.g., Bitcoin and Ethereum, as it is the most popular kind of cryptocurrencies to date. A blockchain is a public and decentralised ledger. The following are typical for blockchain-based cryptocurrencies. All transactions are recorded on the blockchain. Each transaction includes a digital signature of the sender to represent its authorization. The ability to sign a transaction represents the ability to transfer the digital coin.

The main contribution of this thesis is a new approach to build escrow protocol. Observing that, essentially, the transfer of coins is roughly equivalent to the release of a digital signature (on the transaction that transfers the digital coins), we consider the use of a cryptographic primitive, namely, verifiably encrypted signature (VES), to develop escrow protocol for blockchain-based cryptocurrencies. A VES is an encryption of a standard digital signature, with the additional property that the encrypted signature is still verifiable. In other words, given an encrypted signature and a message, everyone can still check if the ciphertext is an encryption of a valid signature on the given message.

Leveraging VES, an escrow protocol can be developed as follows. Buyer Alice signs transaction T which authorises the transfer of her digital coins to seller Bob. Let σ be the signature on T. Instead of sending T to Bob or submitting T to the blockchain, σ is encrypted under the platform's public key. Denote the resulting ciphertext as σ'. σ', along with T, is sent to Bob. With the verifiability of the VES scheme, Bob can check if σ' is the encryption of a valid signature on T. Upon successful validation, Bob delivers the goods to Alice. Alice then submits (T, σ) to the blockchain, resulting in the transfer of coins from Alice to Bob and thus completing the transaction. In case Alice does not respond upon receiving the goods, Bob sends σ' to the platform for arbitration. If the platform decides that Bob has delivered the goods, it can decrypt σ' to obtain σ, and returns it to Bob. Bob can publish (T, σ) to the blockchain to receive the digital coins.

Looking ahead, the approach has the following advantages. Firstly, the platform's involvement is minimal. If there is no dispute, it does not need to get involved. Besides, most of the work is conducted off-chain. The efficiency of the off-chain work depends mainly on the efficiency of the VES. Finally, we believe the above intuition is simple (and thus easy to understand and implement). To the best of our knowledge, we are the first group to investigate building escrow protocol of blockchain-based cryptocurrencies from VES. The rest of the thesis contains contributions that tackle the various technical obstacles from realizing the above design idea. Specifically, the contributions of this thesis are summarized as follows.

- **Escrow Protocol from Verifiably Encrypted Signatures.** We formalise the above design idea. More concretely, we give a concrete security model of escrow protocol for cryptocurrencies, and describe how VES can be used to build protocols fulfilling our definition. We remark that VES alone is not sufficient as there is still a gap between the transfer of the coins and the transfer of digital signatures. We bridge the gap using additional techniques, including time-lock and multi-signature, both of which are supported by major blockchain-based cryptocurrencies.

- **Verifiably Encrypted ECDSA.** While verifiably encrypted signatures have been studied for more than 20 years, we require a way to verifiably encrypt an ECDSA signature, the signature adopted in major cryptocurrencies such as Bitcoin and Ethereum, to realise our framework. We propose the most efficient verifiably encrypted ECDSA to date. We also provide a formal security analysis of our verifiably encrypted ECDSA.

- **Verifiably Encrypted EdDSA-like Signature Scheme.** There are many variants of ECDSA signatures, and different cryptocurrencies support different variants. We develop a framework to categorise these variants (and possibly more), and called this family of digital signatures EdDSA-like signature schemes. We develop a way to encrypt EdDSA-like signatures in a way that is still verifiable. We provide two concrete instantiations which are very efficient. Combining with the first contribution, our work results in escrow protocols for blockchain-based cryptocurrencies using EdDSA-like signatures to endorse transactions.

- **Complexity Analysis.** We conduct a thorough complexity analysis of our escrow protocols and showed that its cost is comparable to the baseline escrow protocol with a fully trusted platform (which involves two standard transactions, i.e., coins first sent to the fully trusted platform, then coins are sent from the platform to the seller or buyer). Our prototype implementation on Bitcoin mainnet confirms our findings.

1.3 Thesis Structure

The rest of the thesis is organized as follows. Chapter 2 describes the related work. Notations and preliminaries are provided in Chapter 3. In Chapter 4, we develop a framework to build escrow protocol from VES, demonstrate its security, and provide a detailed comparison with the existing approaches. To realize such a framework, a specific VES for ECDSA is described and thoroughly analyzed in Chapter 5. Chapter 6 gives a generic construction of VES for all EdDSA-like signatures. Chapter 7 reports the performance of our escrow protocols. Chapter 8 concludes our thesis and discusses future directions.

Chapter 2

Literature Review

In this chapter, we give the related work of this thesis. In Section 2.1, we review the underlying signature scheme adopted by major cryptocurrencies, as well as the time-lock mechanism and multi-signature mechanism. We focus on verifiably encrypted signature in Section 2.2. Related works of escrow protocols, puzzle-solver protocols, and optimistic fair exchange protocols are described in Section 2.3, Section 2.4, and Section 2.5, respectively.

2.1 Cryptocurrency

We review two important concepts in existing blockchain-based cryptocurrencies relevant to this thesis, namely, digital signatures and conditional payments.

Digital Signatures. In blockchain-based cryptocurrencies, the ownership of coins is represented by a unique public/private key pair, and the transfer of digital coins is authorized with digital signatures.

Though the number of cryptocurrencies exceeds 5000, only a few signature algorithms are adopted. Almost all of these algorithms are based on elliptic curve cryptography. Table 2.1 summarises the signature algorithms, along with their underlying elliptic curves, adopted by the top 10 cryptocurrencies ranked by CoinMarketCap [15]. As shown in the table, all leading cryptocurrencies adopt ECDSA, EdDSA, and Schnorr signature scheme.

The only exception is Binance Coin, which exists as a token built on top of another blockchain (the Blockchain of Bitcoin and Ethereum).

Table 2.1: Signature Algorithms behind Cryptocurrency

Cryptocurrency	Signature Algorithm	Elliptic Curve
Bitcoin	ECDSA	secp256k1
Ethereum	ECDSA	secp256k1
Tether	Bitcoin Omni layer / Ethereum ERC-20 token	
Cardano	EdDSA	ed25519
XRP	ECDSA, EdDSA	secp256k1
polkadot	Schnorr signature	Ed25519
Binance Coin	Bitcoin Omni layer / Ethereum ERC-20 token	
Litecoin	ECDSA	secp256k1
Bitcoin Cash	ECDSA, Schnorr signature	secp256k1
Chainlink	Schnorr signature	secp256k1

Conditional Payment. An appealing feature of blockchain-based cryptocurrencies is their programmability. Specifically, one could specify the conditions under which the coins can be spent. In Bitcoin and its forks, one could specify the conditions using its scripting language, the Bitcoin script. There are five standard Bitcoin transaction scripts, namely, 1) pay-to-public-key-hash (`P2PKH`), the default type of transaction, which specifies a hash of some public key and requires a signature under that public key to spend coins, 2) pay-to-public-key (`P2PK`), which specifies a public key and requires a signature under that public key to spend coins, 3) multi-signature (`MultiSig`), which specifies a list of public keys and requires multiple signatures under (the subset of) these public keys to spend coins, 4) pay-to-script-hash, (`P2SH`), which specifies a hash value and requires the pre-image of that hash value to spend coins, and 5) data output(`OP_RETURN`), which is used to store data. Additionally, Bitcoin scripts also support time-lock, which restricts the spending of bitcoins until a specified future time or block height. For more details, please refer to Section 3.8. While Ethereum supports Turing-complete smart contracts which allow arbitrary conditions to be expressed.

Looking ahead, two types of conditions are needed in the construction of our escrow protocols, namely, multi-signature and time-lock. These two conditions are supported by the leading cryptocurrencies listed in Table 2.1. For example, in Bitcoin and its forks, multi-signature and time-lock mechanism can be expressed via `OP_ CHECKMULTISIG` and

`OP_CHECKLOCKTIMEVERIFY`, instructions of the Bitcoin script. On the other hand, these two mechanisms can be readily specified as Ethereum smart contracts.

2.2 Verifiably Encrypted Signature

The notion of verifiably encrypted signature was formalized in 2003 by Boneh et al. [8]. A VES scheme involves a signer, a verifier, and a TTP called adjudicator. In such a scheme, the signer "encrypts" the signature in a way that the validity of the resulting "ciphertext" can be checked by the verifier and a valid signature can be extracted from the "ciphertext" by the adjudicator. We use double quotes here because, as pointed out by Calderon et al. [11], the original definition of VES does not capture the intuition to incorporate encryption. In other words, it is possible to construct a VES scheme that fits in the algorithm definition but makes no use of encryption. As a fix, Calderon et al. introduced an additional property called resolution duplication and proved that VES with resolution duplication implies the use of public-key encryption. Unfortunately, the converse is not true.

Generally speaking, there are three approaches to construct VES schemes, namely, zero-knowledge proofs [2, 5], bilinear maps [8, 36, 11, 26], and Merkle authentication trees [40, 41].

In VES via zero-knowledge proofs, the signer encrypts a signature under the adjudicator's public key and generates a zero-knowledge proof that the ciphertext is a valid signature encryption. Based on interactive zero-knowledge proof, Asokan et al. [2] proposed the first VES scheme for signature schemes with homomorphic-inverse structure (e.g., RSA, DSS, Schnorr, Fiat-Shamir, Guillou-Quisquater, and Ong-Schnorr signatures) in 1998. However, the resulting scheme is highly inefficient and interactive due to the use of a cut-and-choose mechanism. Bao et al. [5] employed the PEDLDLL protocol[1] due to [46] to construct more efficient protocols for the Guillou-Quisquater signature scheme and the DSA-like signature scheme. The former was eventually broken by Boyd and Foo [5] while the latter remains

[1]It stands for Proof of Equivalence of Discrete Logarithm to Discrete LogLogarithm.

secure. We would like to remark that, in principle, Bao et al.'s construction for DSA-like signature can be extended to include ECDSA. However, the security of their construction requires that the discrete logarithm problem remains hard in a double discrete log setting, meaning that the order of the group should be at least one thousand bits and is thus not applicable to typical ECDSA scheme where the group order is a few hundred bits only. [36] sketched the first generic construction of VES based on adaptive unbounded NIZKs. However, adaptive unbounded NIZKs are typically expensive in terms of both computational costs and bandwidth costs. VES schemes for lattice-based signature schemes with efficient NIZKs were proposed in [35, 47].

VES via bilinear maps is specific to signature schemes based on bilinear maps. In this approach, the signer simply encrypts the signature under the adjudicator's public key. The validity of the resulting ciphertext can be checked with pairing operations. In 2003, the first VES scheme from bilinear maps based on BLS signature was proposed by Boneh et al. [8]. Constructions for other signature schemes based on bilinear maps, waters signature [36], Boneh-Boyen signature [11], and SPS-EQ-R [26], are subsequently proposed.

VES via Merkle authentication trees works only for "maskable" signature schemes. Roughly, maskability requires signatures can be masked with some masking values in a way that the "masked" signature is still verifiable and can be unmasked using masking values. In this approach, the signer and the adjudicator set up one-time masking values and establish a Merkle authentication tree during key generation phase. To generate a verifiably encrypted signature, the signer encrypts the pre-defined masking value under the adjudicator's public key, computes a masked signature, and generates an authentication path to show that the "masked" signature is correctly generated. In 2009, Rückert introduced the first VES scheme based on Merkle authentication trees from RSA [40] , which is then formalized and generalized by [41] to fit in all signature schemes with maskability property.

In comparison, generic VES via zero-knowledge proofs can be applied to all standard signature schemes, but the concrete constructions can be very inefficient with respect to computation costs and bandwidth. VES via bilinear maps and Merkle authentication trees are generally more efficient compared with zero-knowledge proof approach. But bilinear maps and Merkle authentication trees restrict the choice of signature schemes. Moreover, VES via Merkle authentication trees requires an interactive key set-up.

2.3 Escrow Protocol for Cryptocurrency

Escrow protocol for cryptocurrencies trades digital coins for goods or services between a buyer (i.e., Alice) and a seller (i.e., Bob) with the help of a TTP in a fair manner. Here, fairness means either both parties or none of them gets what they want. Moreover, the TTP is trusted to make a fair decision in the event of disputes. Roughly, escrow protocol for cryptocurrencies can be divided into three phases, namely, pre-condition, deposit, and withdrawal. In the pre-condition phase, necessary conditions to conduct an escrowed transaction are met. In the deposit phase, coins are locked with some redeem condition. In the withdrawal phase, either Alice or Bob claims the coins according to the pre-defined redeem conditions.

Following the characterization of Goldfeder et al. [24], existing constructions are based on the following approaches. We refer the reader to Section 3.9 for the definition of desirable properties of escrow protocols and detailed analysis of each approach.

- 2-of-3 `MultiSig`[23]. This approach makes use of a standard Bitcoin transaction Script multi-signature (i.e. m-of-n `MultiSig` requires presenting m signatures under m-out-of-n pre-defined public keys to claim coins). In Escrow via 2-of-3 `MultiSig`, bitcoins are sent to the "2-of-3 `MultiSig`" address under the public keys of Alice, Bob, and the TTP. Under normal circumstances, Bob first performs his duty. Alice then sends Bob her multi-signature on transferring coins to Bob. And finally, Bob submits two multi-signatures generated by Alice and Bob and claims the coins. If Alice (Bob) deviates from the protocol, the TTP will help Bob (Alice) to claim the bitcoins by sending a signature on transferring coins to Bob (Alice).

Escrow via 2-of-3 `MultiSig` is very efficient in terms of computational costs (two signing operations). However, it lacks privacy since an external observer might identify an escrow transaction from the 2-of-3 `MultiSig` structure. Moreover, the TTP is partially actively involved in the withdrawal phase (i.e., by partially active involvement, we mean the TTP is involved when Alice or Bob disobeys the protocol). Please refer to [23] for the source code of Bitcoin escrow transaction via `MultiSig`.

- Threshold signature. Escrow via threshold signature makes use of the m-of-n threshold ECDSA signature scheme [22], which allows no less than m out of n players to cooperatively generate a ECDSA signature (i.e., please refer to Section 3.5 for the formal definition of threshold signature). In Escrow via threshold signature, bitcoins are sent to a regular public key address whose secret key is shared among Alice, Bob, and the TTP using a 2-of-3 secret sharing (i.e. with m-of-n secret sharing, any m-out-of-n players can recover the secret value). Under normal circumstances, Bob performs his duty. Alice then helps Bob to reconstruct the secret key. And finally, Bob generates a signature on transferring coins to Bob with the secret key and claims the coins. If Alice (Bob) deviates from the protocol, the TTP will help Bob (Alice) to claim the bitcoins by reconstructing the secret key with Bob (Alice).

  Unlike escrow via `MultiSig`, bitcoins are sent to a regular public address, hence is more private. However, it comes at the costs of round efficiency and TTP involvement. The state-of-the-art threshold signature [22] requires 8 rounds of interaction. Moreover, the TTP is heavily involved in both the deposit phase and the withdrawal phase.

- Encrypt-and-swap Mechanism [24]. Escrow via encrypt-and-swap combines the technique of threshold signature and key exchange. In escrow via encrypt-and-swap, bitcoins are sent to a public key address whose secret key is shared between Alice and

Bob using a 2-of-2 secret sharing. Later, Alice and Bob encrypt their secret shares under the TTP's public key and send the ciphertexts to each other along with a validity proof. Under normal circumstances, Bob performs his duty. Alice then helps Bob to reconstruct the secret key. And finally, Bob generates a signature on transferring coins to Bob with the secret key and claims the coins. If Alice (Bob) deviates from the protocol, the TTP will help Bob (Alice) to claim the bitcoins by reconstructing the secret key with Bob (Alice).

This approach requires less TTP involvement compared to the threshold signature approach (the TTP is not involved in the deposit phase), while maintaining privacy. Unfortunately, it still requires an interactive key set-up. Moreover, it requires two additional zero-knowledge proofs to show the secret share is encrypted correctly, hence is computationally expensive.

Additionally, there are measures to reduce the trust placed on the TTP and to enhance its resilience to denial-of-service (Dos) attacks. The first one is bond, which requires the TTP to deposit a bond. Later in the withdrawal phase, the TTP can retrieve the bonds only if the transaction between Alice and Bob is settled. This method fully defends against DoS attacks, but it comes at the cost of privacy, efficiency, and TTP involvement. Another method is group escrow, where the power of the TTP is distributed among multiple entities. It can partially defend against DoS attacks and collusion attacks at the cost of efficiency. These two measures can be combined with the above-mentioned approaches. Campanelli et al. [12] presented an escrow protocol that embeds bond mechanism into escrow via 2-of-3 `MultiSig` and two group escrow protocols based on `MultiSig` and encrypt-and-swap, respectively. The concrete construction and detailed analysis of the above mentioned approaches can be found in Section 3.9.

2.4 Puzzle-solver Protocol

A puzzle-solver protocol is dedicated to selling digital goods over Bitcoin in an automatic and trustless way [12]. It can be regarded as an escrow protocol where the blockchain fills the role of the trusted party. ZKCP protocol [7] and RSA puzzle-solver protocol [27] are two well-known Bitcoin-based puzzle-solver protocols. ZKCP is built upon NIZK. RSA puzzzle-solver protocol adopts cut-and-choose mechanisms. Both protocols makes use of a standard Bitcoin transaction script `P2SH`, which requires presenting a hash pre-image x of some specified value y to claim coins.

Introduced by Maxwell in 2011, ZKCP allows a party to buy puzzle solutions in bitcoins [27]. Specifically, Alice offers one bitcoin for a solution to a puzzle. Bob, who knows the solution, encrypts solution s using secret key k and computes the hash value of key k such that $y = SHA256(k)$. He then sends Alice the encrypted solution c, the hash value y, together with a NIZK that c is the encryption of solution s under secret key k and that y is the hash value of k. After validating the proof, Alice issues a `P2SH` transaction to Bob which specifies that Bob can only claim the bitcoins if he provides a preimage of y. At the end of the protocol, Bob publishes k and claims the bitcoins. Alice obtains s by decrypting c using k. However, if Bob refuses to publish k, Alice's coins will be locked forever. In this case, a refund mechanism is required to transfer bitcoins back to Alice. Banasik et al. [4] addressed this problem by using time-lock commitments where Alice can get her funds back by solving a time-lock puzzle. To avoid additional computation costs, Banasik et al. further described using `CHECKLOCKTIMEVERIFY` as an alternative solution. `CHECKLOCKTIMEVERIFY` is a Bitcoin opcode that establishes an absolute time from when a transaction can be redeemed. Another option is `CHECKSEQUENCEVERIFY` opcode [51], which provides the same feature as `CHECKLOCKTIMEVERIFY` for relative locktime. To avoid expensive generic zero-knowledge proof, the cut-and-choose technique is employed. However, it requires an additional set-up phase and multiple rounds of interactions. The first reported implementation of ZKCP which allows a buyer to offer bitcoin for a sudoku solution, called "pay-to-sudoku", was by Sean Bowe in 2016 [9]. In 2017, Campanelli et al. [12] showed a practical attack against the ZKCP protocol [27] that allows a buyer to learn partial information about the digital good before paying for it. This attack is due to the fact the buyer is allowed to choose common reference string (CRS) that normally should be selected by a trusted third party. As a fix, Campanelli et al. presented an improvement protocol known as Zero-Knowledge Contingent Service Payment (ZKCSP) such that the security of the protocol can still be guaranteed even if the buyer chooses the CRS.

[27] proposed an RSA puzzle-solver protocol as the core component of TumbleBit, an anonymous payment protocol through an untrusted intermediary. Essentially, an RSA puzzle is an RSA ciphertext and the solution is an RSA decryption. RSA puzzle-solver protocol allows Alice to pay one bitcoin in exchange for a solution to an RSA puzzle. This scheme adopted the cut-and-choose technique and exploited the blinding property of RSA. How the RSA puzzle-solver works is as follows. To solve an RSA puzzle z, Alice first creates m blinded RSA puzzles for z, denoted by $\{z_i\}_{i=1}^{m}$, and n fake RSA puzzles (i.e., RSA encryptions on random values), denoted by $\{f_i\}_{i=1}^{n}$. The blinded puzzles and fake puzzles are randomly permuted to $\{p_i\}_{i=1}^{m+n}$ and then sent to Bob. Denote Z the index set of blinded puzzles such that $Z = \{i|p_i \in \{z_j\}_{j=1}^{m}\}$. Denote F the index set of fake puzzles such that $F = \{i|p_i \in \{f_j\}_{j=1}^{n}\}$. Bob solves all $m + n$ RSA puzzles, encrypts each solution s_i with secret key k_i such that $c_i = \mathsf{Enc}(k_i, s_i)$, computes the hash of k_i such that $y_i = SHA256(k_i)$, and finally, sends $\{c_i, y_i\}_{i=1}^{m+n}$ to the buyer. Next, Alice specifies the index set F and reveals the solutions for fake puzzles $\{p_i\}_{i\in F}$ to Bob. Bob then checks the correctness of these solutions and, in return, proves to the buyer that $\{c_i, y_i\}_{i=1\in F}$ has been correctly computed by revealing $\{k_i\}_{i\in F}$. After that, Alice proves that all blinded puzzles $\{p_i\}_{i\in Z}$ unblind to z and issues a `P2SH` transaction which specifies that Bob can only claim the bitcoins if he provides all pre-images of $\{y_i\}_{i\in Z}$. The point of the proof is to show that Alice can learn nothing else beyond the solution to z. After validating the proof,

Bob publishes $\{k_i\}_{i \in Z}$ and claims the bitcoins. Finally, Alice obtains the solution for z by decrypting c_i using k_i for any $i \in Z$ and unblinding the resulting plaintext.

2.5 Optimistic Fair Exchange for Signatures

Conceptualized by Asokan, Schunter, and Waidner [3] in 1997, Optimistic Fair Exchange (OFE) is a kind of protocol for fair exchanges of digital items between two parties. OFE protocol for signatures typically consists of three rounds. In the first round, Alice sends a partial signature to Bob. Upon receiving Alice's partial signature, Bob sends his full signature in response. Finally, Alice sends her full signature to Bob in the third round. However, if Alice refuses to send her full signature, Bob could ask the TTP to retrieve Alice's full signature from the partial signature.

There are three main approaches to construct OFE, namely, verifiable encryption [1], two-party multi-signature [30], and ring signature [49]. In the verifiably encryption paradigm, the partial signature is an encryption of a conventional signature signed by Alice under the TTP's public key along with a proof which states the partial signature is legally generated (i.e., also known as verifiably encrypted signature). And the full signature is Alice's conventional signature. In the multi-signature paradigm, a two-party multi-signature signing key is split into two private keys, i.e., the primary signer's signing key SK_1 and the cosigner's signing key SK_2. Alice holds both keys and the TTP holds SK_2. The partial signature is a conventional signature signed with SK_1. The full signature is a conventional signature signed with SK. In the ring signature paradigm, members of the ring signature are Alice and the TTP. The partial signature is a conventional signature signed by Alice. The full signature is a ring signature signed by either Alice or the TTP along with the partial signature.

In terms of the security model, Dodis and Reyzin [17] considered the case that the arbitrator could be malicious and proposed a new model to capture this requirement. In

PKC 2007, Dodis et al. [16] considered OFE in the multi-user setting (i.e. multiple signers and verifiers, and one arbitrator), while prior work on OFE is developed in the single-user setting(i.e., one signer, one verifier, and one arbitrator). Furthermore, additional properties such as accountability [31], ambiguity [29, 48], and collusion-resistance [50] are studied as well.

While OFE is conceptually similar to an escrow protocol in the sense that both aim to allow the exchange of digital items by two parties in a fair manner with the help of a third party, directly applying OFE to escrow protocol for Bitcoin is non-trivial. Firstly, the aforementioned OFE schemes only support signatures designed in a specific way (i.e. it does not support the exchange of standard signatures such as ECDSA). More importantly, similar to ZKCP, additional mechanisms are needed to prevent a malicious user from "spending" his coins before completion of the exchange protocol.

Chapter 3

Preliminaries

In this chapter, we give the preliminaries that will be used in the subsequent chapters. We provide the notations used in this thesis in Section 3.1. Definitions of the commitment scheme as well as the construction of Pedersen commitment and Pedersen vector commitment are given in Section 3.2. Section 3.3 describes the definition of public key encryption schemes and the construction of the twisted ElGamal encryption scheme. In Section 3.4, we define signature schemes and provide the construction of EdDSA, Schnorr signature scheme, ECDSA, ECGDSA, GOST, and SM2. The definition of the threshold signature scheme is given in Section 3.5. We review the definitions of proof systems and give a brief description of Σ-protocol and Bulletproofs in Section 3.6. The definition of the verifiably encrypted signature is presented in Section 3.7. Basic Bitcoin script is listed in Section 3.8. Finally, we summarize the requirements of the escrow protocol for cryptocurrencies and analyze the existing approaches yielding escrow protocols in Section 3.9. For readers who are familiar with the foregoing topics may skip this chapter.

3.1 Notation

Let $\mathbb{G}$ be a cyclic group of order q. $\mathbb{G}^n$ denotes the vector space of dimension n over $\mathbb{G}$. Let $\mathbb{Z}_q$ be the ring of integers modulo q. $\mathbb{Z}_q^n$ denotes the vector space of dimension n over $\mathbb{Z}_q$.

We use bold front to denote vectors. Let $\mathbf{a}, \mathbf{b}$ be vectors such that $\mathbf{a} = (a_0, ..., a_{n-1})$ and $\mathbf{b} = (b_0, ..., b_{n-1})$. We denote $\langle \mathbf{a}, \mathbf{b} \rangle$ the inner product of $\mathbf{a}$ and $\mathbf{b}$, i.e., $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=0}^{n-1} a_i b_i$, and $\mathbf{a} \circ \mathbf{b}$ the Hadamard product of $\mathbf{a}$ and $\mathbf{b}$, i.e., $\mathbf{a} \circ \mathbf{b} = (a_0 \cdot b_0, ..., a_{n-1} \cdot b_{n-1})$. For $k \in \mathbb{Z}_p$, $\mathbf{k}^n$ denotes the vector containing the first n powers of k, i.e., $\mathbf{k}^n = (1, k, ..., k^{n-1})$. Assume two vectors $\mathbf{u} \in \mathbb{Z}_q^n$, $\mathbf{h} \in \mathbb{G}^n$ such that $\mathbf{u} = (u_0, ..., u_{n-1})$, $\mathbf{h} = (h_0, ..., h_{n-1})$. We define $\mathbf{h}^{\mathbf{u}} = \prod_{i=0}^{n-1} h_i^{u_i}$ and $\mathbf{h}_{[i:j]} = (h_i, ..., h_j)$ where $0 \leq i \leq j \leq n-1$.

For the following cryptography primitives, we consider a common reference string model and assume all parties have access to the same string (aka system parameters) generated in by some trusted party. This captures the real-world scenario that all parties use the same elliptic curve published by some authority.

3.2 Commitment Scheme

Definition 1. ***Commitment Scheme.*** *Let SP denote the system parameters of a commitment scheme. A commitment scheme* $(\mathsf{Com}, \mathsf{Open})$ *can be defined as below:*

- $\mathsf{Com}(m; \beta)$. *Taking as input a message m, this algorithm outputs a commitment c of m using randomness β.*
- $\mathsf{Open}(c, m, \beta)$. *Taking as input c, m, and β, this algorithm outputs 1 if $c = \mathsf{Com}(m; \beta)$.*

Pedersen Commitment. Let $\mathbb{G}$ be a cyclic group of order q and generators g, h. The system parameters are defined as $SP = (\mathbb{G}, q, g, h)$. The commitment scheme, Pedersen Commitment [38] is described below.

- $\mathsf{Com}(m; \beta)$. Output the commitment $c = g^m h^\beta$.
- $\mathsf{Open}(c, m, \beta)$. Output 1 if $c = g^m h^\beta$.

Pedersen Vector Commitment. Let $\mathbb{G}$ be a cyclic group of order q and generators $g_1, ..., g_n, h$. Let $\mathbf{g} = (g_1, ..., g_n)$. The system parameters are defined as $SP = (\mathbb{G}, q, \mathbf{g}, h)$. The Pedersen vector commitment [10] is described as follows.

- $\mathsf{Com}(\mathbf{m} = (m_1, ..., m_n); \beta)$. Output the commitment $c = \mathbf{g}^{\mathbf{m}} h^{\beta}$.
- $\mathsf{Open}(c, \mathbf{m}, \beta)$. Output 1 if $c = \mathbf{g}^{\mathbf{m}} h^{\beta}$.

3.3 Encryption Scheme

Definition 2. ***Public Key Encryption Scheme.*** *Let SP denote the system parameters of an encryption algorithm. A public key encryption scheme* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is defined as below:*

- $\mathsf{KeyGen}(SP)$. *Taking as input SP, this algorithm outputs the encryption key and the decryption key pair (EK, DK).*
- $\mathsf{Enc}(m, EK; \beta)$. *Taking as input EK and a plaintext m, this algorithm outputs a ciphertext c of m with randomness β.*
- $\mathsf{Dec}(c, DK, EK)$. *Taking as input DK, EK, and a ciphertext c, this algorithm outputs m.*

$\mathbf{n'}$-bit Twisted ElGamal Encryption Scheme. Below, we review twisted ElGamal encryption proposed by Yu Chen et al. [14]. Let $\mathbb{G}$ be a cyclic group of order q with generators g_1, g_2. Let n' be an integer with $n' \leq |q|$. The system parameters are defined as $SP = (\mathbb{G}, g_1, g_2, q, n')$. The n'-bit twisted ElGamal encryption scheme is described as follows.

- $\mathsf{KeyGen}(SP)$. Choose $t \xleftarrow{R} \mathbb{Z}_q$ and compute $T = g_2^t$. It sets $EK = T$ and $DK = t$.
- $\mathsf{Enc}(m, EK; \beta_0, ..., \beta_{l-1})$. Assume $|m| = n$ and $\frac{n}{n'} = l$. Define $(m_0, ..., m_{l-1})$ where $m = \sum_{i=0}^{l-1} m_i \cdot 2^{n' \cdot i}$. For $i = 0, ..., l-1$, compute $(u_i, v_i) = (g_1^{m_i} g_2^{\beta_i}, T^{\beta_i})$. The ciphertext c is set as $c = (u_0, ..., u_{l-1}, v_0, ..., v_{l-1})$.

- $\mathsf{Dec}(c, DK, EK)$. For $i = 0, ..., l-1$, compute $M_i = \frac{u_i}{v_i^{1/t}}$ and recover m_i from M_i[1]. Reconstruct $m = \sum_{i=0}^{l-1} m_i \cdot 2^{n' \cdot i}$.

The intuition of the twisted ElGamal Encryption is to switch the role of key encapsulation and the session key of the standard ElGamal encryption. Specifically, the ciphertexts on some message m with randomness β as a result of the standard ElGamal and twisted ElGalmal are in the form of $(g_1^m T^\beta, g_2^\beta)$ and $(g_1^m g_2^\beta, T^\beta)$, respectively.

3.4 Signature Scheme

Definition 3. ***Signature Scheme.*** *Let SP denote the system parameters of a signature algorithm. A signature scheme* $(\mathsf{KeyGen}, \mathsf{Sig}, \mathsf{Ver})$ *is defined as below:*

- $\mathsf{KeyGen}(SP)$. *On input SP, this algorithm outputs the signing key SK and the verification key PK.*

- $\mathsf{Sig}(m, SK, PK; k)$. *On input SK, PK, and message m, this algorithm outputs a signature σ on m with randomness k.*

- $\mathsf{Ver}(m, \sigma, PK)$. *On input PK, σ, and m, this algorithm outputs* 1 *or* 0.

Below, we review various digital signature schemes from elliptic curve cryptography whose security is based on the discrete logarithm problem. Specifically, the cyclic group $\mathbb{G}$ in the system parameters below is always groups of points on a certain elliptic curve, and the security of these schemes depends on the discrete logarithm problem defined over $\mathbb{G}$. These schemes have been included in various standards and are widely adopted in various cryptocurrencies. Following the above definition, each of these schemes, denoted by $\mathcal{S}$, is a tuple $(\mathcal{S}.\mathsf{KeyGen}, \mathcal{S}.\mathsf{Sig}, \mathcal{S}.\mathsf{Sig})$ of algorithms.

[1]By implementing Shanks' algorithm [44], for $|m'| = n'$, m' can be extracted from $g_1^{m'}$ in time $O(2^{n'/2})$ using a table of size $O(2^{n'/2})$. In practice, n' will be chosen to be 32 to ensure Shanks' algorithm remains efficient.

ECGDSA [33]. Let $\mathbb{G}$ be a cyclic group of order q with generator g_1. Define cryptographic hash function $H : \{0,1\}^* \to \{0,1\}^b$. Let (Q_x, Q_y) be the coordinates of point $Q \in \mathbb{G}$ and function $F : F(Q_x, Q_y) = Q_x \bmod q$. The system parameters are defined as $SP = (\mathbb{G}, g_1, q, H, b)$.

- $\mathcal{S}.\mathsf{KeyGen}(SP)$. Choose $d \overset{R}{\leftarrow} \mathbb{Z}_q$ and compute $Q = g_1^{d^{-1}}$. Set $SK = d, PK = Q$.
- $\mathcal{S}.\mathsf{Sig}(m, SK, PK; k)$. Compute $h = H(m)$, $R = g_1^k$, $r = F(R)$, and $s = (kr - h)d$. The signature is $\sigma = (r, s)$.
- $\mathcal{S}.\mathsf{Sig}(m, \sigma, PK)$. Compute $h = \mathrm{H}(m)$ and output 1 if $r = F(g_1^{h \cdot r^{-1}} Q^{s \cdot r^{-1}})$.

GOST [18]. Let $\mathbb{G}$ be a cyclic group of order q with generator g_1. Define cryptographic hash function $H : \{0,1\}^* \to \{0,1\}^b$. Let (Q_x, Q_y) be the coordinates of point $Q \in \mathbb{G}$ and function $F : F(Q_x, Q_y) = Q_x \bmod q$. The system parameters are defined as $SP = (\mathbb{G}, g_1, q, H, b, F)$.

- $\mathcal{S}.\mathsf{KeyGen}(SP)$. Choose $d \overset{R}{\leftarrow} \mathbb{Z}_q$ and compute $Q = g_1^d$. Set $SK = d, PK = Q$.
- $\mathcal{S}.\mathsf{Sig}(m, SK, PK; k)$. Chooses $k \overset{R}{\leftarrow} \mathbb{Z}_p$. Compute $h = H(m)$, $R = g_1^k$, $r = F(R)$, and $s = rd + kh$. The signature is $\sigma = (r, s)$.
- $\mathcal{S}.\mathsf{Ver}(m, \sigma, PK)$. Compute $h = \mathrm{H}(m)$ and output 1 if $r = F(g_1^{s \cdot h^{-1}} Q^{-r \cdot h^{-1}})$.

ECDSA [39]. Let $\mathbb{G}$ be a cyclic group of order q with generator g_1. Define cryptographic hash function $H : \{0,1\}^* \to \{0,1\}^b$. Let (Q_x, Q_y) be the coordinates of point $Q \in \mathbb{G}$. Define $F : F(Q) = Q_x \mod q$. The system parameters are defined as $SP = (\mathbb{G}, g_1, q, H, b, F)$.

- $\mathcal{S}.\mathsf{KeyGen}(SP)$. Choose $d \overset{R}{\leftarrow} \mathbb{Z}_q$ and compute $Q = g_1^d$. Set $SK = d, PK = Q$.
- $\mathcal{S}.\mathsf{Sig}(m, SK, PK; k)$. Chooses $k \overset{R}{\leftarrow} \mathbb{Z}_p$. Compute $h = H(m)$, $R = g_1^k$, $r = \mathrm{F}(R)$, and $s = k^{-1}(h + rd)$. The signature is $\sigma = (r, s)$.

- $\mathcal{S}.\mathsf{Ver}(m, \sigma, PK)$. Compute $h = \mathrm{H}(m)$ and output 1 if $r = F(g_1^{h \cdot s^{-1}} Q^{r \cdot s^{-1}})$.

SM2 [45]. Let g_1 be a base point over elliptic curve $E(\mathbb{F}_q)$ defined by two elements $a, c \in \mathbb{F}_q$. Let (Q_x, Q_y) be the coordinates of point $Q \in \mathbb{G}$. Define cryptographic hash function $H : \{0,1\}^* \to \{0,1\}^b$. The signer has a distinguishing identifier ID_Q of length $entlen_Q$ bits. Denote $ENTL_Q$ the two bytes converted from integer $entlen_Q$. The system parameters are defined as $SP = (\mathbb{F}_q, g_1, H, b, a, c)$.

- $\mathcal{S}.\mathsf{KeyGen}(SP)$. Choose $d \xleftarrow{R} \mathbb{Z}_q$ and compute $Q = g_1^d$. Set $SK = d, PK = Q$.
- $\mathcal{S}.\mathsf{Sig}(m, SK, PK; k)$. Compute $\overline{m} = H(ENTL_Q||ID_Q||a||c||Q||g_1)||m$ and $h = H(\overline{m})$, $R = g_1^k$, $r = h + F(R)$, and $s = (1+d)^{-1} \cdot (k - rd)$. The signature is $\sigma = (r, s)$.
- $\mathcal{S}.\mathsf{Ver}(m, \sigma, PK)$. Compute $\overline{m} = H(ENTL_Q||ID_Q||a||c||Q||g_1)||m$, $h = \mathrm{H}(\overline{m})$, and output 1 if $r = F(g_1^s Q^{r+s}) + h$.

EdDSA [34]. Let $\mathbb{G}$ be a cyclic group of order q with generator g_1. Define hash function $H : \{0,1\}^* \to \{0,1\}^{2b}$. Denote $H_{i:j}(x)$ the concatenation of the i-th to j-th bits of $H(x)$. The system parameters are defined as $SP = (\mathbb{G}, g_1, q, H, b)$. We review the EdDSA scheme $\mathcal{S}$ below:

- $\mathcal{S}.\mathsf{KeyGen}(SP)$. Compute $d = H_{1:b}(x), k = H_{b+1:2b}(x)$. Let $Q = g_1^d$. Set $SK = d, PK = Q$
- $\mathcal{S}.\mathsf{Sig}(m, SK, PK; k)$. Compute $r = H(k, m)$, $R = g_1^r$, $h = H(R, PK, m)$, and $s = r + hd$. The signature is $\sigma = (R, s)$.
- $\mathcal{S}.\mathsf{Ver}(m, \sigma, PK)$. Compute $h = H(R, PK, m)$ and output 1 if $RQ^h = g_1^s$.

Schnorr Signature Scheme [42]. Let $\mathbb{G}$ be a cyclic group of order q with generator g_1. Define hash function $H : \{0,1\}^* \to \{0,1\}^b$. The system parameters are defined as $SP = (\mathbb{G}, g_1, q, H, b)$.

- $\mathcal{S}.\mathsf{KeyGen}(SP)$. Choose $d \overset{R}{\leftarrow} \mathbb{Z}_q$ and compute $Q = g_1^d$. Set $SK = d, PK = Q$.
- $\mathcal{S}.\mathsf{Sig}(m, SK, PK; k)$. Chooses $k \overset{R}{\leftarrow} \mathbb{Z}_q$. Compute $R = g_1^k$, $r = H(R, m)$, and $s = k - rd$. The signature is $\sigma = (R, s)$.
- $\mathcal{S}.\mathsf{Ver}(m, \sigma, PK)$. Compute $r = H(R, m)$. Output 1 if $Q^{-r}R = g_1^s$.

Original Schnorr signature uses a multiplicative group of integers modulo a large prime as the cyclic group, while lately group of points of an elliptic curve is more popular in practice.

3.5 Threshold Signature Scheme

Definition 4. ***Threshold Signature Scheme.*** *A (t, n)-threshold signature scheme allows t out of n players to jointly generate a signature. Consider a signature scheme $\mathcal{S} = (\mathsf{KeyGen}, \mathsf{Sig}, \mathsf{Ver})$ where $(PK, SK) \leftarrow \mathsf{KeyGen}(SP)$, $\sigma \leftarrow \mathsf{Sig}(m, SK, PK)$, $0/1 \leftarrow \mathsf{Ver}(m, \sigma, PK)$. The (t, n)-threshold signature scheme for $\mathcal{S}$ includes two additional protocols, namely,* $\mathsf{ThreshKeyGen}, \mathsf{ThreshSig}$, *for the set of n players $P_1, ..., P_n$ to jointly generate the key and signatures respectively.* $\mathsf{ThreshKeyGen}, \mathsf{ThreshSig}$ *are defined as follows.*

- $\mathsf{ThreshKeyGen}(SP)$. *On input SP, this protocol outputs PK to all players and s_i to P_i for $i = \{1, ..., n\}$ such that*
  - *$(s_1, ..., s_n)$ forms a (t, n)-threshold secret sharing [43] of a certain value SK*[2].
  - *The distribution of (PK, SK) is the same as that of* KeyGen *on input SP, i.e., $(PK, SK) \in \{\mathsf{KeyGen}(SP)\}$.*
- $\mathsf{ThreshSig}(s'_1, ..., s'_{t'}, m, PK)$. *On input $t' \geq t$ out of n secret shares, i.e., $\{s'_1, ..., s'_{t'}\} \subset \{s_1, ..., s_n\}$ and $t' \geq t$, message m, and public key PK, this algorithm returns σ such that* $\mathsf{Ver}(m, \sigma, PK) = 1$.

[2]In other words, SK can be reconstructed from any subset $\mathcal{T} \subset \{s_1, \ldots, s_n\}$ with $|\mathcal{T}| \geq t$.

Besides the standard unforgeability requirement, the security of threshold signature schemes requires that collusion of less than t parties will not be able to generate a valid signature.

3.6 Proof System

Let R be a polynomial-time decidable relation. We call w a witness for statement u under Common Reference String (CRS) ck if $(ck, u, w) \in R$. The CRS-dependent language L_{ck} in the relation R is defined as

$$L_{ck} = \{u | \exists w : (ck, u, w) \in R\}$$

Consider a probabilistic polynomial-time algorithm $\mathcal{G}$ and a pair of interactive probabilistic polynomial-time algorithms $\langle \mathcal{P}, \mathcal{V} \rangle$ defined as below:

- $\mathcal{G}(1^\lambda)$. Taking as input security parameter λ, this algorithm outputs the CRS ck.
- $\langle \mathcal{P}(w, u, ck), \mathcal{V}(u, ck) \rangle$. Taking as input w, u, and ck, this pair of algorithms output 1 if $\mathcal{V}$ accepts.

Definition 5. ***Proof System.*** *$(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is a proof system for relation R in the CRS model if it satisfies completeness and soundness.*

- **Completeness**. *$(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is complete if for any $(ck, u, w) \in R$,*

$$Pr\left[\langle P(w, u, ck), V(u, ck) \rangle = 1\right] = 1$$

- **Soundness**. *$(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is sound if for any $u \notin L_{ck}$ and any probabilistic polynomial-time cheating prover P^*,*

$$Pr\left[\langle P^*(u, ck), V(u, ck) \rangle = 1\right] \approx 0$$

Definition 6. ***Argument of Knowledge.*** *$(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is an argument of knowledge for relation R in the CRS model if it satisfies completeness and computational witness-extended emulation.*

- ***Witness-Extended Emulation.*** *$(\mathcal{G}, \mathcal{P}, \mathcal{V})$ satisfies computational witness-extended emulation if for all deterministic polynomial-time P^*, there is an efficient extraction algorithm B such that for any probabilistic polynomial-time adversary $\mathcal{A}_1, \mathcal{A}_2$,*

$$Pr\left[\begin{array}{r} ck \leftarrow \mathcal{G}(1^\lambda); (u,s) \leftarrow \mathcal{A}_2(ck); \\ tr \leftarrow < P^*(ck,u,s), V(ck,u) >: \\ \mathcal{A}_1(tr) = 1 \end{array}\right] \approx$$

$$Pr\left[\begin{array}{r} ck \leftarrow \mathcal{G}(1^\lambda); (u,s) \leftarrow \mathcal{A}_2(ck); \\ (tr, w) \leftarrow B^{\mathcal{O}}(ck,u): \\ \mathcal{A}_1(tr) = 1 \\ \wedge\ (tr \text{ is accepting} \rightarrow (ck,u,w) \in R) \end{array}\right]$$

, where the transcript oracle $\mathcal{O} =< P^*(ck,u,w), V(ck,u) >$, and allows for rewinding with fresh randomness.

Definition 7. ***Special Honest-Verifier Zero-Knowledge (SHVZK).*** *A proof system (i.e. argument of knowledge) $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is SHVZK if for every $(ck,u,w) \in R$, there exists a probabilistic polynomial-time simulator $\mathcal{S}$ such that for any probabilistic polynomial-time ad-*

versary $\mathcal{A}_1, \mathcal{A}_2$*,*

$$Pr\begin{bmatrix} ck \leftarrow \mathcal{G}(1^\lambda); (u, w, \rho) \leftarrow \mathcal{A}_2(ck); \\ tr \leftarrow< P(ck, u, w), V(\sigma, u; \rho) >: \\ (ck, u, w) \in R \wedge \mathcal{A}_1(tr) = 1 \end{bmatrix} \approx$$

$$Pr\begin{bmatrix} ck \leftarrow \mathcal{G}(1^\lambda); (u, w, \rho) \leftarrow \mathcal{A}_2(ck); \\ tr \leftarrow S(u, \rho) : \\ (ck, u, w) \in R \wedge \mathcal{A}_1(tr) = 1 \end{bmatrix}$$

, where ρ *is the public coin used by* V*.*

Definition 8. ***Non-interactive Zero-Knowledge Proof (NIZK).*** *NIZK is a zero-knowledge proof that requires no interaction between the prover and the verifier. A non-interactive zero-knowledge proof for* $(ck, u, w) \in R$ *in the CRS model consists of three probabilistic polynomial-time algorithms* $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ *defined as follows.*

- $ck \leftarrow \mathcal{G}(1^\lambda)$*: Taking as input security parameter* λ*, this algorithm outputs a CRS* ck*.*
- $\pi \leftarrow \mathcal{P}(ck, u, w)$*: Taking as input* $(ck, u, w) \in R$*, this algorithm outputs a proof* π*.*
- $b \leftarrow \mathcal{V}(ck, u, \pi)$*: Taking as input* (ck, x, π)*, this algorithm returns a single bit* b*.*

Σ-protocol. A Σ-protocol is a three-move zero-knowledge proof system between a prover P and a verifier V. We adopt the definition from [25] for Σ-protocol with a common reference string (CRS). Assume there is a probabilistic polynomial-time setup algorithm $\mathcal{G}$ that generates CRS ck, a Σ-protocol is defined as follows.

- $ck \leftarrow \mathcal{G}(1^\lambda)$: Generate the CRS on input security parameter λ.
- $a \leftarrow \mathcal{P}(ck, u, w)$: Given $(ck, u, w) \in R$ where w is a witness for a statement u, the prover generates an initial message a.
- $x \leftarrow \{0, 1\}^\lambda$: The verifier chooses x uniformly at random.

- $z \leftarrow \mathcal{P}(x)$: The prover responds to challenge x with z.
- $b \leftarrow \mathcal{V}(ck, u, a, x, z)$: The verifier returns a single bit b.

Σ-protocol satisfies soundness and zero-knowledge in the following sense:

- **Special Soundness**. $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is sound if there is an efficient extraction algorithm $\mathcal{B}$ such that for any probabilistic polynomial-time adversary $\mathcal{A}$ outputting two accepting transcripts with the same initial message,

$$Pr\left[\begin{array}{c} ck \leftarrow \mathcal{G}(1^\lambda); (u, a, x_1, z_1, x_2, z_2) \leftarrow \mathcal{A}(ck); \\ w \leftarrow \mathcal{B}(ck, u, a, x_1, z_1, x_2, z_2) : (ck, u, w) \in R \end{array}\right] \approx 1$$

- **SHVZK**. $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is SHVZK if there exists a probabilistic polynomial-time simulator $\mathcal{S}$ such that for any probabilistic polynomial-time adversary $\mathcal{A}$,

$$Pr\left[\begin{array}{c} ck \leftarrow \mathcal{G}(1^\lambda); (u, w, x) \leftarrow \mathcal{A}(ck); \\ a \leftarrow \mathcal{P}(ck, u, w); z \leftarrow \mathcal{P}(x) : \mathcal{A}(a, z) = 1 \end{array}\right] \approx$$
$$Pr\left[\begin{array}{c} ck \leftarrow \mathcal{G}(1^\lambda); (u, w, x) \leftarrow \mathcal{A}(ck); \\ (a, z) \leftarrow \mathcal{S}(ck, u, x) : \mathcal{A}(a, z) = 1 \end{array}\right]$$

Σ-protocol can be converted into NIZK proof using the Fiat-Shamir heuristic.

Bulletproofs. Bulletproofs [10] is a ZKAoK with logarithmic proof size (with respect to the witness size). The core component of Bulletproofs is Bünz's inner product argument. In particular, on input $(\mathbf{h}, \mathbf{k}, P, \hat{t}; \mathbf{l}, \mathbf{r})$, where $\mathbf{h}, \mathbf{k}$ are dependent generators in $\mathbb{G}^n$ and $\mathbf{l}, \mathbf{r} \in \mathbb{Z}_q^n$, Bünz's inner product argument allows the prover to convince the verifier that

$$P = \mathbf{h}^{\mathbf{l}}\mathbf{k}^{\mathbf{r}}, \hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle$$

with proof size logarithmic in n. In this thesis, we will use Bünz's inner product argument in a black-box manner. For the concrete construction, please refer to [10]. Bulletproofs can be converted into a NIZK proof using the Fiat-Shamir heuristic.

Bulletproofs for Pedersen Commitment. Bulletproofs is well-suited for range proofs for Pedersen commitment and supports efficient proof aggregation. Let $\mathbb{G}$ be a cyclic group of order q and generators g_1, g_2. The Pedersen commitment u to m is of the form $g_1^m g_2^\beta$, where $\beta \overset{R}{\leftarrow} \mathbb{Z}_q$. Essentially, Bulletproofs for Pedersen commitment proves the following:

$$L = \{u, g_1, g_2, m, \beta | u = g_1^m g_2^\beta, m \in [0, 2^n - 1]\}$$

Let $\mathbf{m}_L = (m_1, ..., m_n)$ be the bit vector of m where $m = \sum_{i=1}^n m_i \cdot 2^{i-1}$. The fact that m is in the range of $[0, 2^n - 1]$ can be equivalently expressed as

$$\langle \mathbf{m}_L, \mathbf{2^n} \rangle = m \ \wedge \ \mathbf{m}_L \circ \mathbf{m}_R = \mathbf{0}^n \ \wedge \ \mathbf{m}_l - \mathbf{m}_R = \mathbf{1}^n \ (1)$$

Using the fact that $< \mathbf{0}^n, \mathbf{y}^n >= 0$ and $z^2 \cdot 0 + z \cdot 0 = 0$ for all integers y, z, (1) holds if

$$z^2 \cdot \langle \mathbf{m}_L, \mathbf{2^n} \rangle + z \cdot \langle \mathbf{m}_l - 1^n - \mathbf{m}_R, \mathbf{y}^n \rangle + \langle \mathbf{m}_L, \mathbf{m}_R \circ \mathbf{y}^n \rangle = z^2 \cdot m \ (2)$$

where $y, z \overset{R}{\leftarrow} Z_q$. And (2) can be equivalently transformed into

$$\langle \mathbf{m}_L - z \cdot \mathbf{1}^n, \mathbf{y}^n \circ (\mathbf{m}_R + z \cdot \mathbf{1}^n) + z^2 \cdot \mathbf{2}^n \rangle = z^2 \cdot m + \delta(y, z) \ (3)$$

, where $\delta(y, z) = (z - z^2) \cdot \langle \mathbf{1}^n, \mathbf{y}^n \rangle - z^3 \langle \mathbf{1}^n, \mathbf{2}^n \rangle$.

Therefore, the proof for $m \in [0, 2^n - 1]$ is reduced to the proof of (3). We review the construction of Bulletproofs for Pedersen commitment $u = g_1^m g_2^\beta$ in Figure 3.1.

3.7 Verifiably Encrypted Signature Scheme

A verifiably encrypted signature scheme involves a signer, a verifier, and a TTP called adjudicator. At first, the signer encrypts the signature under an adjudicator's public key. The verifier then checks the validity of the ciphertext. Later in the scheme, if the signer refuses to release the signature, the adjudicator decrypts the ciphertexts and releases the

$\mathcal{P}(u, g_1, g_2; \mathbf{m}_L, \mathbf{m}_R, \beta)$ $\qquad\qquad$ $\mathcal{V}(u, g_1, g_2)$

Let $\mathbf{h}, \mathbf{k}$ two set of independent generators in $\mathbb{G}^n$.

$\alpha, \rho \overset{R}{\leftarrow} \mathbb{Z}_q$

$\rho_{\mathbf{L}}, \rho_{\mathbf{R}} \overset{R}{\leftarrow} \mathbb{Z}_q^n$

$A = g_2^{\alpha}\mathbf{h}^{\mathbf{m_L}}\mathbf{k}^{\mathbf{m_R}}$

$S = g_2^{\rho}\mathbf{h}^{\rho_{\mathbf{R}}}\mathbf{k}^{\rho_{\mathbf{L}}}$

$\xrightarrow{A,S}$

$y, z \overset{R}{\leftarrow} \mathbb{Z}_q^*$

$\xleftarrow{y,z}$

Define $l(X), r(X), t(X)$ as below :

$l(X) = (\mathbf{m_L} - z \cdot 1^n) + \rho_{\mathbf{L}} \cdot X$

$r(X) = y^n \circ (\mathbf{m_L} + z \cdot 1^n + \rho_{\mathbf{R}} X) + z^2 \cdot 2^n$

$t(X) = \langle l(X), R(X)\rangle = t_0 + t_1 X + t_2 X^2$

$i.e., t_0 = u \cdot z^2 + \delta(y, z)$

$\alpha \overset{R}{\leftarrow} \mathbb{Z}_q$

$\tau_1, \tau_2 \overset{R}{\leftarrow} \mathbb{Z}_q$

$T_1 = g_1^{t_1} g_2^{\tau_1}, T_1 = g_1^{t_2} g_2^{\tau_2}$

$\xrightarrow{T_1,T_2}$

$x \overset{R}{\leftarrow} \mathbb{Z}_q^*$

$\xleftarrow{x}$

$\mathbf{l} = l(x), \mathbf{r} = r(x), \hat{t} = \langle l, r\rangle$

$\tau_x = \tau_2 \cdot x^2 + \tau_1 \cdot x + z^2 \cdot \beta$

$\mu = \alpha + \rho \cdot x$

$\xrightarrow{\mathbf{l},\mathbf{r},\hat{\tau},\tau_x,\mu}$

Set $\mathbf{k}' = (k_1, k_2^{y^{-1}}, ..., k_n^{y^{-n+1}})$

Compute a commitment P to $l(x), r(x),$

$i.e., P = A \cdot S^x \cdot \mathbf{h}^{-z}(\mathbf{k}')^{z \cdot y^n + z^2 \cdot 2^n}$

Accept if and only if:

1) $g^{\hat{\tau}} h^{\tau_x} = u^{z^2} g_1^{\delta(y,z)} T_1^x T_2^{x^2}$

// Check that $\hat{\tau} = t_0 + t_1 x + t_2^2$.

2) $P = g_2^{\mu} \cdot \mathbf{h}^{\mathbf{l}} \cdot \mathbf{k}'^{\mathbf{r}}$

// Check that $\mathbf{l}, \mathbf{r}$ are correctly formed.

3) $\hat{\tau} = \langle \mathbf{l}, \mathbf{r}\rangle$

// Check that $\hat{\tau}$ are correctly formed.

Figure 3.1: Bulletproofs for Pedersen Commitment

signatures. Let $\{\mathsf{KeyGen}, \mathsf{Sig}, \mathsf{Ver}\}$ denote a standard signature scheme. We review the notion and security model of the verifiably encrypted signature as below [8].

Definition 9. ***Verifiably Encrypted Signature Scheme.*** *Let SP denote the system parameters of a verifiably encrypted signature scheme. A verifiably encrypted signature scheme consists of the following probabilistic polynomial-time algorithms:*

- $\mathsf{KeyGen}, \mathsf{Sig}, \mathsf{Ver}$. *Same as a standard signature scheme.*
- $\mathsf{AdjKeyGen}(SP)$. *Taking as input the system parameters SP, this algorithm generates the adjudicator's key pair (APK, ASK).*
- $\mathsf{VESSig}(m, SK, APK)$. *Taking as input signer's secret key SK, adjudicator's public key APK, and message m, this algorithm outputs the verifiably encrypted signature σ'.*
- $\mathsf{VESVer}(m, \sigma', PK, APK)$. *Take as input a public key PK, an adjudicator's public key APK, a message m, and a verifiably encrypted signature σ' on m, this algorithm outputs 1 if σ' a valid verifiably encrypted signature on m under PK; otherwise, outputs 0.*
- $\mathsf{Adj}(m, \sigma', ASK, APK)$. *Taking as input an adjudicator's key pair (APK, ASK), a public key PK, a message m, and a verifiably encrypted signature σ' on m, this algorithm outputs the standard signature σ on m.*

We say a verifiably encrypted signature is 2-phase if VESSig can be computed from a standard signature and an adjudicator's public key. More formally, $\mathsf{VESSig}(m, sk, PK)$ can be divided into $\sigma \leftarrow \mathsf{VESSig}^{(1)}(m, SK)$ and $\sigma' \leftarrow \mathsf{VESSig}^{(2)}(\sigma, APK)$ such that $\mathsf{VESSig}^{(1)}$ is the same as Sig, and $\mathsf{VESSig}^{(2)}$ takes only σ and APK as input.

The security of the verifiably encrypted signature consists of four aspects, namely, validity, unforgeability, (strong) opacity, and resolution-independence (Optional). We strengthen the unforgeability notion by merging the security requirements of abuse-freeness. Specifically, the attacker will be given the adjudicator's secret key when it attempts to forge a verifiably encrypted signature.

- Validity requires correctly generated encrypted signatures and adjudicated signatures being accepted by verifiers. That is,

$$\begin{aligned}\mathsf{VESVer}(m, \mathsf{VESSig}(m, SK, PK, APK), PK, APK) &= 1\\ \mathsf{Ver}(m, \mathsf{Adj}(m, \sigma', ASK, APK), PK) &= 1\end{aligned}$$

- Unforgeability requires that it is computationally infeasible for any probabilistic polynomial-time adversary $\mathcal{F}$ to forge a verifiably encrypted signature. We define the following experiment:

$$\begin{aligned}\mathsf{AdjKeyGen}(SP) &\rightarrow (ASK, APK)\\ \mathsf{KeyGen}(SP) &\rightarrow (SK, PK)\\ (m, \sigma') &\leftarrow \mathcal{F}^{O_{\mathsf{VESSig}}}(PK, ASK, APK)\\ \text{success of } \mathcal{F} := &\begin{bmatrix} & \mathsf{VESVer}(m, \sigma', PK, APK) = 1\\ \wedge & (m) \notin Query(\mathcal{F}, O_{\mathsf{VESSig}})\end{bmatrix}\end{aligned}$$

  where the verifiably encrypted signing oracle O_{VESSig} is defined as $O_{\mathsf{VESSig}}(m) \rightarrow (\sigma', m)$, and $Query(\mathcal{F}, O_{\mathsf{VESSig}})$ is the set of valid queries issued to the oracle O_{VESSig}.

- Strong opacity requires that it is computationally infeasible for any probabilistic polynomial-time adversary $\mathcal{B}$ to extract a message-signature pair from a verifiably encrypted signature. We define the following experiment:

$$\begin{aligned}
\mathsf{AdjKeyGen}(SP) &\rightarrow (ASK, APK)\\
\mathsf{KeyGen}(SP) &\rightarrow (SK, PK)\\
(m^*, \sigma^*) &\leftarrow \mathcal{B}^{O_{\mathsf{VESSig}}, O_{\mathsf{Adj}}}(PK, APK)\\
\text{success of } \mathcal{F} :&= \begin{bmatrix} \mathsf{Ver}(m^*, \sigma^*, PK) = 1 \\ \wedge \quad (m^*, \sigma^*) \notin Res(\mathcal{B}, O_{\mathsf{Adj}}) \end{bmatrix}
\end{aligned}$$

where the resolution adjudication oracle O_{Adj} is defined as $O_{\mathsf{Adj}}(m, \sigma') \rightarrow (m, \sigma)$, and $Res(\mathcal{B}, O_{\mathsf{Adj}})$ is the set of responses from oracle O_{Adj}.

- Resolution-independence requires that the signatures returned by the adjudicator is identically distributed with the ordinary signature. That is,

$$\begin{aligned}
\mathsf{AdjKeyGen}(SP) &\rightarrow (ASK, APK)\\
\mathsf{KeyGen}(SP) &\rightarrow (SK, PK)\\
\mathsf{Sig}(m, SK, PK) &= Adj(m, \mathsf{VESSig}(m, SK, PK, APK), ASK, APK)
\end{aligned}$$

3.8 Bitcoin Script

Bitcoin Script (or simply *Script*) is a stack-based scripting programming language for Bitcoin transactions. It consists of data and `opcodes`. The data includes, but is not limited to, hash to public keys, public keys, and signatures. `opcodes` used in our protocol are listed in Table 3.1.

The five standard Bitcoin transaction scripts are pay-to-public-key-hash (`P2PKH`), pay-to-public-key (`P2PK`), multi-signature (`MultiSig`), pay-to-script-hash (`P2SH`), and data output (`OP_RETURN`).

- `P2PKH` allows Bitcoin transactions to the public key hash. It is the most commonly

Table 3.1: Basic opcodes

opcode	Description
OP_IF	If the top stack value is not False, the statements are executed. The top stack value is removed.
OP_ELSE	If the preceding OP_IF or OP_NOTIF or OP_ELSE was not executed, the statements are executed.
OP_ENDELSE	Ends an if/else block
OP_DROP	Removes the top stack item.
OP_EQUAL	Returns 1 if the inputs are exactly equal, 0 otherwise.
OP_HASH160	The input is hashed twice: first with SHA-256 and then with RIPEMD-160.
OP_CHECKSIG	Return 1 if the signature for the public key is valid.
OP_CHECKMULTISIG	Return 1 if the signatures for a set of public keys are valid.
OP_CheckLockTimeVerify	Allow transaction outputs to be encumbered by a timelock.

used, also the default type of transaction. The script pattern of P2PKH is "OP DUP OP HASH160 <PUBLIC KEY A HASH> OP EQUAL OP CHECKSIG". To unlock a P2PKH transaction, a spender must provide the public key that yields the <PUBLIC KEY A HASH> and a valid signature under the public key.

- P2PK allows Bitcoin transactions to a public key. It is most often used in coinbase transactions. The script pattern of P2PK is "<PUBLIC KEY A> OP CHECKSIG". To unlock a P2PK transaction, a spender must provide a valid signature under <PUBLIC KEY A>.

- Multisig allows Bitcoin transactions authorized by multiple parties. It is most commonly used as a joint account. The script pattern of Multisig, also known as M-of-N scheme, is " M <PUBLIC KEY 1> <PUBLIC KEY 2> ... <PUBLIC KEY N> N OP CHECKMULTISIG". To unlock a Multisig transaction, the spender(s) must at least provide M signatures under M-of-N listed public keys.

- P2SH allows Bitcoin transactions to an arbitrary script hash. The most common use of P2SH is the standard Multisig script. Take the P2SH Script in this thesis for example, i.e., "OP_HASH160 script hash OP_EQUAL". To unlock such a P2SH transaction, the spender must provide a Multisig script that yields the script hash along with the signature that makes the Multisig script evaluate to true.

- OP_RETURN is used to store data rather than make payments. The script pattern is "

`OP RETURN <DATA>`”.

3.9 Escrow protocols for Cryptocurrency

3.9.1 Desirable Properties of Escrow Protocols

The desirable properties of escrow protocols fall into four categories, namely, security, privacy, TTP’s involvement, and efficiency. We follow the definition of [24]. The requirements are elaborated as follows.

- **Security**. An escrow protocol should guarantee (a) the exchange will be accomplished if any two parties follow the protocol, and (b) no party gets worse off if the other party deviates from the protocol. Security can be further classified into three aspects, namely, *Security against Alice, Security against Bob*, and *Security against TTP*. Furthermore, it is desirable to be resilient against denial-of-service attacks.
  - *Security against Alice* requires that Alice (the buyer) should not be able to obtain Bob’s (the seller’s) goods without paying even if TTP decided to aid Bob. In other words, Alice should not be able to produce a fake escrow that appears to be valid but cannot be redeemed even if Bob and TTP cooperate. Furthermore, Alice should not be able to take back coins in the escrow before Bob has the chance to request the TTP for resolution.
  - *Security against Bob* requires that Bob should not be able to transfer coins in the escrow without the help of the TTP (or Alice).
  - *Security against TTP* requires that TTP should not be able to transfer any coins in the escrow without the help of Alice or Bob.
  - *Denial-of-Service (DoS).* An escrow protocol is anti-DoS if the TTP must mediate when there is a dispute.

- **Privacy**. An escrow protocol is private if it does not leak information about what happens to different parties. It can be further classified into three aspects, namely, *Internally-hiding*, *Externally-hiding*, and *Dispute-hiding*.
  - *Internally-hiding*. An escrow protocol is *internally-hiding* if the TTP cannot identify the transactions involved in the escrow protocols if no dispute occurs.
  - *Externally-hiding*. An escrow protocol is *externally-hiding* if an external observer cannot distinguish the transactions of escrow protocols from the standard transactions on the blockchain.
  - *Dispute-hiding*. An escrow protocol is *dispute-hiding* if an external observer cannot tell if there is a dispute.
- **TTP Involvement**. It is desirable to keep TTP involvement to be minimal.
  - *Active on deposit.* An escrow protocol is *active on deposit* if the TTP must actively participate in the deposit phase of the escrow protocol.
  - *Active on withdrawal.* An escrow protocol is *active on withdrawal* if the TTP must actively participate in the coin transfer phase no matter there is a dispute or not. An escrow protocol is *partially active on withdrawal* if the TTP must participate in the coin transfer if the buyer or the seller deviates from the protocol. An escrow protocol is *passive on withdrawal* if the TTP must participate in the coin transfer phase when the buyer deviates from the protocol.
- **Efficiency.** Efficiency contains three aspects, namely, round efficiency, communication efficiency, and computation efficiency. In most cases, round efficiency is the most important metric for efficiency. In particular, round efficiency is considered under three situations, namely, normal, abort and refund, and adjudication by the TTP.

3.9.2 Existing Approaches

Escrow via 2-of-3 $\mathbf{Multisig}$. In this approach, multiple keys are required to authorize a transaction. Denote T the transaction from Alice to Bob and T' the refund transaction to Alice. In the deposit phase, the redeem conditions are set as any two multi-signatures generated by Alice, Bob, and the TTP, which is denoted by 2-of-3{`Signature_A`, `Signature_B`, `Signature_TTP`}.

Let $\mathcal{S} = (\mathcal{S}.\mathsf{KeyGen}, \mathcal{S}.\mathsf{Sig}, \mathcal{S}.\mathsf{Ver})$ be a standard signature scheme with system parameters $SP_{\mathcal{S}}$. The concrete construction of the escrow protocol for Bitcoin via 2-of-3 $\mathbf{Multisig}$ is shown in Figure 3.2.

- *Security against Alice*. Bob can claim the coins with the help of the TTP if Alice refuses to pay.
- *Security against Bob*. Alice can claim the coins with the help of the TTP if Bob refuses to perform his duty.
- *Security against TTP*. The TTP can only transfer coins by collaborating with Alice or Bob.
- *DoS*. Escrow via 2-of-3 $\mathbf{Multisig}$ does not offer anti-DoS. The TTP can simply refuse to mediate.
- *Internally-hiding*. If no dispute occurs, the TTP is not involved and will not be able to learn if an escrow transaction has taken place.
- *Partially externally-hiding*. Due to the 2-of-3 $\mathbf{Multisig}$ structure, an external party may successfully identify an escrow transaction.
- *Partially dispute-hiding*. An external observer may recognize a dispute through transaction graph analysis.

Pre-condition:

- Alice runs $(PK_A, SK_A) \leftarrow \mathcal{S}.\mathsf{KeyGen}(SP_{\mathcal{S}})$.
- Bob runs $(PK_B, SK_B) \leftarrow \mathcal{S}.\mathsf{KeyGen}(SP_{\mathcal{S}})$.
- The TTP runs $(APK, ASK) \leftarrow \mathcal{S}.\mathsf{KeyGen}(SP_{\mathcal{S}})$.

The Deposit Phase:

- The redeem condition is set as:
  2-of-3{`Signature_A`, `Signature_B`, `Signature_TTP`}.

The Withdrawal Phase:
If Bob fails to perform his duty within block length t,

- The TTP broadcasts its part of multi-signature σ'_{TTP} on T', i.e., $\sigma'_{\mathsf{TTP}} \leftarrow \mathcal{S}.\mathsf{Sig}(\mathsf{T}', ASK, APK)$.
- Alice broadcasts her part of multi-signature σ'_A on transaction T', i.e., $\sigma'_A \leftarrow \mathcal{S}.\mathsf{Sig}(\mathsf{T}', SK_A, PK_A)$, and claims the coins.

If Bob completes his duty within t and Alice broadcasts her part of multi-signature σ_A on transaction T, i.e., $\sigma_A \leftarrow \mathcal{S}.\mathsf{Sig}(\mathsf{T}, SK_A, PK_A)$,

- Bob broadcasts his part of multi-signature σ_B on transaction T, i.e., $\sigma_B \leftarrow \mathcal{S}.\mathsf{Sig}(\mathsf{T}, SK_B, PK_B)$, and claims the coins.

If Bob completes his duty but Alice refuses to broadcast her part of multi-signature,

- The TTP broadcasts its part of multi-signature σ_{TTP} on T, i.e., $\sigma_{\mathsf{TTP}} \leftarrow \mathcal{S}.\mathsf{Sig}(\mathsf{T}, ASK, APK)$.
- Bob broadcasts his part of multi-signature σ_B on T, i.e., $\sigma_B \leftarrow \mathcal{S}.\mathsf{Sig}(\mathsf{T}, SK_B, PK_B)$, and claims the coins.

Figure 3.2: Escrow via 2-of-3 $\mathbf{Multisig}$

- *TTP Involvement*. Escrow via 2-of-3 **Multisig** is *passive-on-deposit* since the TTP is not involved in the deposit phase, and is *partially active-on-withdrawal* since the TTP will be called upon to handle resolution requests whenever Alice or Bob misbehaves.
- *Round Efficiency.* One round of off-chain communication is required.

Escrow via Threshold Signature. In this approach, the digital coins are sent to an address whose secret key is shared among Alice, Bob, and the TTP using a 2-of-3 secret sharing. Denote T the transaction from the joint account to Bob and T' the transaction from the joint account to Alice. In the deposit phase, the redeem conditions are set as signatures generated under the threshold address PK, which is denoted by {`Signature_PK`}.

Let $\mathcal{S} = (\mathcal{S}.\mathsf{KeyGen}, \mathcal{S}.\mathsf{Sig}, \mathcal{S}.\mathsf{Ver})$ be the standard ECDSA signature scheme with system parameters $SP_{\mathcal{S}}$. Let $(\mathsf{ThreshkeyGen}, \mathsf{ThreshSig})$ be a (2, 3)-threshold ECDSA signature scheme of Gennaro al. [22]. The concrete construction of the escrow protocol via threshold signature is shown in Figure 3.3.

Pre-condition:

- Alice, Bob, and the TTP run $(s_A, s_B, s_{\mathsf{TTP}}; PK) \leftarrow \mathsf{ThreshKeyGen}(SP_{\mathcal{S}})$.

The Deposit Phase:

- The redeem condition is set as: {`Signature_PK`}

The Withdrawal Phase:
If Bob fails to perform his duty,

- Alice and the TTP generate a signature σ'_{PK} on T', i.e., $\sigma'_{\mathsf{PK}} \leftarrow \mathsf{Thresh.Sig}(s_A, s_{\mathsf{TTP}}, \mathsf{T}', PK)$.
- Alice broadcasts σ'_{PK} and claims the coins.

If Bob completes his duty, and Alice and Bob jointly generate a signature σ_{PK} on T, i.e., $\sigma_{\mathsf{PK}} \leftarrow \mathsf{Thresh.Sig}(s_A, s_B, \mathsf{T}, PK)$,

- Bob broadcasts σ_{PK} and claims the coins.

If Bob completes his duty but Alice refuses to pay,

- Bob and the TTP generate a signature σ_{PK} on T, i.e., $\sigma_{\mathsf{PK}} \leftarrow \mathsf{Thresh.Sig}(s_B, s_{\mathsf{TTP}}, \mathsf{T}, PK)$,
- Bob broadcasts σ_{PK} and claims the coins.

Figure 3.3: Escrow via Threshold Signature

- *Security against Alice.* Bob can claim the coins with the help of the TTP if Alice refuses to pay.

- *Security against Bob.* Alice can claim the coins with the help of the TTP if Bob refuses to perform his duty.
- *Security against TTP.* The TTP can only transfer coins by collaborating with Alice or Bob.
- *DoS.* Escrow via threshold signature does not offer anti-DoS. The TTP can simply refuse to mediate.
- *Not Internally-hiding.*The TTP is actively involved in the deposit phase. Therefore, the TTP can recognize an escrow transaction even if no dispute occurs.
- *Externally-hiding.* An external party will not be able to learn if an escrow transaction has taken place since the threshold signature is indistinguishable from a standard signature.
- *Dispute-hiding.* An external party cannot identify a dispute because a standard signature will be published no matter there is a dispute or not.
- *TTP Involvement.* Escrow via threshold signature is *active-on-deposit* since the TTP must participate in the deposit phase to establish the joint account, and is *partially active-on-withdrawal* since the TTP is only called upon to handle resolution requests whenever Alice or Bob misbehaves.
- *Round Efficiency.* In the pre-condition phase, 4 rounds of off-chain communication are required to establish the threshold address. In the withdrawal phase, 4 rounds of off-chain communication are required to jointly generate a valid signature. To sum up, 8 rounds of off-chain communication are required.

Escrow via encrypt-and-swap. In this approach, the digital coins are sent to an address whose secret key is shared between Alice and Bob using a 2-of-2 secret sharing. Alice (i.e. Bob) then sends the secret share to Bob (i.e. Alice) under the TTP's public key. Denote T the transaction from the joint account to Bob and T' the transaction from the joint account to Alice. In the deposit phase, the redeem conditions are set as signatures generated under the threshold address PK, which is denoted by `{Signature_PK}`.

Let $\mathcal{S} = (\mathcal{S}.\mathsf{KeyGen}, \mathcal{S}.\mathsf{Sig}, \mathcal{S}.\mathsf{Ver})$ be the standard ECDSA signature scheme with system parameters $SP_{\mathcal{S}}$. Let $(\mathsf{ThreshKeykeyGen}, \mathsf{ThreshSig})$ be the (2, 2)-threshold ECDSA signature scheme of Gennaro al. [22]. Let $\mathcal{E} = (\mathcal{E}.\mathsf{KeyGen}, \mathcal{E}.\mathsf{Enc}, \mathcal{E}.\mathsf{Dec})$ be an encryption scheme with system parameter $SP_{\mathcal{E}}$. The concrete construction of the escrow protocol via encrypt-and-swap is shown in Figure 3.4.

- *Security against Alice*. If Alice refuses to pay, the TTP will send Alice's secret share of to Bob so that Bob could transfer the coins in the joint account to his own account.
- *Security against Bob*. If Bob refuses to perform his duty, the TTP will send Bob's secret share to Alice so that Alice could transfer the coins in the joint account to her own account.
- *Security against TTP*. The TTP can only transfer the coins by collaborating with Alice or Bob.
- *DoS*. Escrow via threshold signature does not offer anti-DoS. The TTP can simply refuse to mediate.
- *Internally-hiding*. If no dispute occurs, the TTP is not involved and will not be able to learn if an escrow transaction has taken place.
- *Externally-hiding*. An external party will not be able to learn if an escrow transaction has taken place since the signature resulting from the protocol is indistinguishable from a standard signature.

Pre-condition:

- Alice and Bob run $(s_A, s_B; y_A, y_B, PK) \leftarrow \mathsf{ThreshKeyGen}(SP_{\mathcal{S}})$ s.t. $y_A = g^{s_A}$, $y_B = g^{s_A}$, $PK = y_A \cdot y_B$
- The TTP runs $(APK, ASK) \leftarrow \mathcal{E}.\mathsf{KeyGen}(SP_{\mathcal{E}})$.
- Alice computes $c_A \leftarrow \mathcal{E}.\mathsf{Enc}(APK, s_A)$ and generates a zero-knowledge proof π_A such that c_A is an encryption of the discrete log with respect to g of y_A. Alice sends c_A, π_A to Bob.
- Bob computes $c_B \leftarrow \mathcal{E}.\mathsf{Enc}(APK, s_B)$ and generates a zero-knowledge proof π_B such that c_B is an encryption of the discrete log with respect to g of y_B. Bob sends c_B, π_B to Alice.

The Deposit Phase:

- The redeem condition is set as: {`Signature_PK`}

The Withdrawal Phase:
If Bob fails to perform his duty,

- Alice sends c_B to the TTP.
- The TTP runs $s_B \leftarrow \mathcal{E}.\mathsf{Dec}(ASK, APK, c_B)$ and sends s_B to Alice.
- Alice broadcasts a signature σ'_{PK} on T', i.e., $\sigma'_{\mathsf{PK}} \leftarrow \mathcal{S}.\mathsf{Sig}(s_A + s_B, \mathsf{T}', PK)$, and claims the coins.

If Bob completes his duty within t, and Alice sends s_A to Bob,

- Bob broadcasts a signature σ_{PK} on T, i.e., $\sigma_{\mathsf{PK}} \leftarrow \mathcal{S}.\mathsf{Sig}(s_A + s_B, \mathsf{T}, PK)$, and claims the coins.

If Bob completes his duty but Alice refuses to pay,

- Bob sends c_A to the TTP.
- The TTP runs $s_A \leftarrow \mathcal{E}.\mathsf{Dec}(ASK, APK, c_A)$, and sends s_A to Bob.
- Bob broadcasts a signature σ_{PK} on T, i.e., $\sigma_{\mathsf{PK}} \leftarrow \mathcal{S}.\mathsf{Sig}(s_A + s_B, \mathsf{T}, PK)$, and claims the coins.

Figure 3.4: Escrow via encrypt-and-swap

- *Dispute-hiding*. An external party cannot identify a dispute since the signature resulting from the protocol is indistinguishable from a standard signature.
- *TTP Involvement.* Escrow via encrypt-and-swap is *passive-on-deposit* since the TTP doesn't participate in the deposit phase, and is *partially active-on-withdrawal* since the TTP is called upon to handle resolution requests from both Alice and Bob.
- *Round Efficiency.* In the pre-condition phase, 2 rounds of off-chain communication are required to generate keys, and 2 rounds of off-chain communication are required to exchange encrypted key shares. In the withdrawal phase, 1 or 2 rounds of off-chain communication are required to recover the threshold secret key. To sum up, 5 or 6 rounds of off-chain communication are required.

Escrow with Bond. In this approach, the TTP is required to put deposits into the escrow. Denote T the transaction from Alice to Bob, T' the refund transaction to Alice, and T^* the deposit refund transaction to the TTP. Let H denote the SHA-256 hash function. In the deposit phase, the redeem conditions are set as, (a) 2-of-3 multi-signatures generated by Alice, Bob, and the TTP and a preimage of hash value y, which is denoted by 2-of-3{`Signature_A`, `Signature_B`, `Signature_TTP`} $\wedge x|H(x) = y$, and (b) a signature generated by the TTP and a preimage of hash value y, i.e., {`Signature_TTP`} $\wedge x|H(x) = y$.

Let $\mathcal{S} = (\mathcal{S}.\mathsf{KeyGen}, \mathcal{S}.\mathsf{Sig}, \mathcal{S}.\mathsf{Ver})$ be a standard signature scheme with system parameters $SP_{\mathcal{S}}$. The concrete construction of the escrow protocol with bond is shown in Figure 3.5.

- *Security against Alice.* Bob can claim the coins with the help of the TTP if Alice refuses to pay.
- *Security against Bob.* Alice can claim the coins with the help of the TTP if Bob refuses to conduct his duty.
- *Security against TTP.* The TTP can only transfer coins by collaborating with Alice or Bob.
- *Anti-DoS.* Escrow via bond offers anti-DoS. The TTP cannot obtain x and claim the deposit if it refuses to mediate.
- *Not Internally-hiding.* The TTP must participate in the deposit phase. Therefore, the TTP can recognize an escrow protocol even if no dispute occurs.
- *Partially Externally-hiding.* Due to the 2-of-3 $\mathbf{Multisig}$ and hash preimage structure, an external party may successfully identify an escrow transaction.
- *Partially Dispute-hiding.* An external observer may recognize a dispute through traffic graph analysis.

Pre-condition:

- Alice and Bob agree on a value x and compute $y = H(x)$.
- Alice runs $(PK_A, SK_A) \leftarrow \mathcal{S}.\mathsf{KeyGen}(SP_{\mathcal{S}})$.
- Bob runs $(PK_B, SK_B) \leftarrow \mathcal{S}.\mathsf{KeyGen}(SP_{\mathcal{S}})$.
- The TTP runs $(APK, ASK) \leftarrow \mathcal{S}.\mathsf{KeyGen}(SP_{\mathcal{S}})$.

The Deposit Phase:

- The redeem condition is set as: `Signature_T`$^* \wedge x|H(x) = y$;
  2-of-3{`Signature_A`, `Signature_B`, `Signature_TTP`} $\wedge x|H(x) = y$.

The Withdrawal Phase:
If Bob fails to perform his duty within block length t,

- The TTP broadcasts its part of multi-signature σ'_{TTP} on T', i.e., $\sigma'_{\mathsf{TTP}} \leftarrow \mathcal{S}.\mathsf{Sig}(\mathsf{T}', ASK, APK)$.
- Alice broadcasts x, and her part of multi-signature σ'_A on transaction T', i.e., $\sigma'_A \leftarrow \mathcal{S}.\mathsf{Sig}(\mathsf{T}', SK_A, PK_A)$, and claims the coins.
- The TTP submits x and σ^*_{TTP} on transaction T^*, i.e., $\sigma^*_{\mathsf{TTP}} \leftarrow \mathcal{S}.\mathsf{Sig}(\mathsf{T}^*, ASK, APK)$, and claims the deposit coins.

If Bob completes his duty within t and Alice broadcasts her part of multi-signature σ_A on transaction T, i.e., $\sigma_A \leftarrow \mathcal{S}.\mathsf{Sig}(\mathsf{T}, SK_A, PK_A)$,

- Bob submits x and his part of multi-signature σ_B on transaction T, i.e., $\sigma_B \leftarrow \mathcal{S}.\mathsf{Sig}(\mathsf{T}, SK_B, PK_B)$, and claims the coins.
- The TTP submits x and σ^*_{TTP} on transaction T^*, i.e., $\sigma^*_{\mathsf{TTP}} \leftarrow \mathcal{S}.\mathsf{Sig}(\mathsf{T}^*, ASK, APK)$, and claims the deposit coins.

If Bob completes his duty but Alice refuses to broadcast her part of multi-signature,

- The TTP broadcasts its part of multi-signature σ_{TTP} on T, i.e., $\sigma_{\mathsf{TTP}} \leftarrow \mathcal{S}.\mathsf{Sig}(\mathsf{T}, ASK, APK)$,
- Bob broadcasts x and his part of multi-signature σ_B on T, i.e., $\sigma_B \leftarrow \mathcal{S}.\mathsf{Sig}(\mathsf{T}, SK_B, PK_B)$, and claims the coins.
- The TTP submits x and σ^*_{TTP} on transaction T^*, i.e., $\sigma^*_{\mathsf{TTP}} \leftarrow \mathcal{S}.\mathsf{Sig}(\mathsf{T}^*, ASK, APK)$, and claims the deposit coins.

Figure 3.5: Escrow with Bond

- *TTP Involvement*. Escrow with bond is *active-on-deposit* since the TTP must participate in the deposit phase, and is *active-on-withdrawal* since the TTP must participate in the withdrawal phase even if no dispute occurs.
- *Round Efficiency.* One round of off-chain communication is required.

Group Escrow via 2-of-3 $\mathbf{Multisig}$. Denote T the transaction from Alice to Bob and T' the refund transaction to Alice. In the deposit phase, the redeem conditions are set as, a) two multi-signatures generated by Alice and Bob, or b) one multi-signature generated by Alice or Bob together with $n - \text{out} - \text{of} - (2n + 1)$ multi-signatures generated by the TTPs, which is denoted by 2-of-3{`Signature_A`, `Signature_B`, n-out-of-(2n+1){`Signature_TTP`$_0$`, ...,  Signature_TTP`$_{2n}$}}.

Let $\mathcal{S} = (\mathcal{S}.\mathsf{KeyGen}, \mathcal{S}.\mathsf{Sig}, \mathcal{S}.\mathsf{Ver})$ be a standard signature scheme with system parameters $SP_{\mathcal{S}}$. The concrete construction of the group escrow protocol via 2-of-3 $\mathbf{Multisig}$ is shown in Figure 3.6.

Group escrow via 2-of-3 $\mathbf{Multisig}$ improves the anti-DoS of escrow via 2-of-3 $\mathbf{Multisig}$ with the cost of round efficiency. It is partially against the DoS attack since we spare the power of adjudication. Under normal circumstances, only one round of off-chain communication is required. However, if either Alice or Bob misbehaves, at least n rounds of off-chain communication are required (i.e. at least n TTPs send their signature to Alice or Bob).

Pre-condition:

- Alice runs $(PK_A, SK_A) \leftarrow \mathcal{S}.\mathsf{KeyGen}(SP_\mathcal{S})$.
- Bob runs $(PK_B, SK_B) \leftarrow \mathcal{S}.\mathsf{KeyGen}(SP_\mathcal{S})$.
- TTP_1 runs $(APK_0, ASK_0) \leftarrow \mathcal{S}.\mathsf{KeyGen}(SP_\mathcal{S})$.
- ...
- TTP_{2n} runs $(APK_{2n}, ASK_{2n}) \leftarrow \mathcal{S}.\mathsf{KeyGen}(SP_\mathcal{S})$.

The Deposit Phase:

- The redeem condition is set as:
  2-of-3$\{$`Signature_A`, `Signature_B`, $n-$out-of$-(2n+1)\{$`Signature_TTP`$_0, \ldots,$ `Signature_TTP`$_{2n}\}\}$.

The Withdrawal Phase:
If Bob fails to perform his duty within block length t,

- Suppose that TTP_i votes for Alice, it broadcasts σ'_{TTP_i} on T', i.e., $\sigma'_{\mathsf{TTP}_i} \leftarrow \mathcal{S}.\mathsf{Sig}(\mathsf{T}', ASK_i, APK_i)$.
- Alice broadcasts her part of multi-signature σ'_A on transaction T', i.e., $\sigma'_A \leftarrow \mathcal{S}.\mathsf{Sig}(\mathsf{T}', SK_A, PK_A)$, and claims the coins.

If Bob completes his duty within t and Alice broadcasts her part of multi-signature σ_A on transaction T, i.e., $\sigma_A \leftarrow \mathcal{S}.\mathsf{Sig}(\mathsf{T}, SK_A, PK_A)$,

- Bob broadcasts his part of multi-signature σ_B on transaction T, i.e., $\sigma_B \leftarrow \mathcal{S}.\mathsf{Sig}(\mathsf{T}, SK_B, PK_B)$, and claims the coins.

If Bob completes his duty but Alice refuses to broadcast her part of multi-signature,

- Suppose that TTP_i votes for Bob, it broadcasts σ_{TTP_i} on T, i.e., $\sigma_{\mathsf{TTP}_i} \leftarrow \mathcal{S}.\mathsf{Sig}(\mathsf{T}, ASK_i, APK_i)$.
- Bob broadcasts his part of multi-signature σ_B on T, i.e., $\sigma_B \leftarrow \mathcal{S}.\mathsf{Sig}(\mathsf{T}, SK_B, PK_B)$, and claims the coins.

Figure 3.6: Group Escrow via 2-of-3 $\mathsf{Multisig}$

Group Escrow via encrypt-and-swap. In this approach, the digital coins are sent to an address whose secret key is shared between Alice and Bob using a 2-of-2 secret sharing. Alice (i.e. Bob) will then initiate a n-out-of-($2n + 1$) Shamir secret sharing [43] of her key share with $2n + 1$ TTPs, i.e., $\mathrm{TTP}_0, ..., \mathrm{TTP}_{2n}$. More specifically, Alice (Bob) generates a random polynomial of degree $n - 1$ whose integer coefficient is the key share, and sends Bob (Alice) $2n + 1$ different points on the polynomial. Since that n points can uniquely define a polynomial with degree $n - 1$, n-out-of-$2n + 1$ TTPs can collaboratively reconstruct the polynomial and recover the secret share. Denote T the transaction from the joint account to Bob and T' the transaction from the joint account to Alice. In the deposit phase, the redeem conditions are set as signatures generated under threshold address PK, which is denoted by {`Signature_PK`}.

Let $\mathcal{S} = (\mathcal{S}.\mathsf{KeyGen}, \mathcal{S}.\mathsf{Sig}, \mathcal{S}.\mathsf{Ver})$ be the standard ECDSA signature scheme with system parameters $SP_{\mathcal{S}}$. Let $(\mathsf{ThreshKeykeyGen}, \mathsf{ThreshSig})$ be the $(2, 2)$-threshold ECDSA signature scheme for $\mathcal{S}$. Let $\mathcal{E} = (\mathcal{E}.\mathsf{KeyGen}, \mathcal{E}.\mathsf{Enc}, \mathcal{E}.\mathsf{Dec})$ be an encryption scheme. The concrete construction of the group escrow protocol via encrypt-and-swap is shown in Figure 3.7. In the pre-condition phase, two rounds of communication are required to generate keys, and two rounds of communication are required to exchange the encrypted key shares. In the withdrawal phase, depending on whether there is a dispute, one or $2n$ rounds of off-chain communication are required to recover the signing key. To sum up, 5 rounds of off-chain communication if no dispute occurs, and $2n + 4$ rounds otherwise.

Pre-condition:

1. Alice and Bob run $(s_A, s_B; PK) \leftarrow \mathsf{ThreshKeyGen}(SP_{\mathcal{S}})$.
2. TTP_1 runs $(APK_0, ASK_0) \leftarrow \mathcal{S}.\mathsf{KeyGen}(SP_{\mathcal{S}})$...
   TTP_{2n} runs $(APK_{2n}, ASK_{2n}) \leftarrow \mathcal{S}.\mathsf{KeyGen}(SP_{\mathcal{S}})$.
3. Alice shares s_A over a random polynomial P_A with degree $n-1$ whose integer coefficient is s_A,
   $$P_A(x) = s_A + a_1 x ... + a_{n-1}x^{n-1}.$$
4. For $i = 0, ..., 2n-1$, Alice computes $s_{A,i} = P_A(i)$, $c_{A,i} \leftarrow \mathcal{E}.\mathsf{Enc}(APK_i, s_{A,i})$, and generates a proof $\pi_{A,i}$ such that the decryption of $c_{A,i}$ is indeed a Shamir secret sharing of s_A. Alice sends $c_A = (c_{A,1}, ..., c_{A,2n-1})$ and $\pi_A = (\pi_{A,1}, ..., \pi_{A,2n-1})$ to Bob.
5. Bob checks π_A.
6. Bob and Alice repeat steps 5-7 with input s_B.

The Deposit Phase:

- The redeem condition is set as: {`Signature_PK`}

The Withdrawal Phase:
If Bob fails to perform his duty,

- Alice sends c_B to the TTPs.
- Suppose that TTP_i votes for Alice, it runs $s_{B,i} \leftarrow \mathcal{E}.\mathsf{Dec}(ASK_i, APK_i, c_{B,i})$ and sends $s_{B,i}$ to Alice.
- Alice recovers polynomial $P_B(x)$, extracts s_B, and computes signature σ'_{PK}, i.e., $\sigma'_{PK} \leftarrow \mathcal{S}.\mathsf{Sig}(s_A + s_B, \mathsf{T}', PK)$. Alice broadcasts σ'_{PK} and claims the coins.

If Bob completes his duty within t, and Alice sends s_A to Bob,

- Bob broadcasts a signature σ_{PK} on T, i.e., $\sigma_{PK} \leftarrow \mathcal{S}.\mathsf{Sig}(s_A + s_B, \mathsf{T}, PK)$.

If Bob completes his duty but Alice refuses to pay,

- Bob sends c_A to the TTPs.
- Suppose that TTP_i votes for Bob, it runs $s_{A,i} \leftarrow \mathcal{E}.\mathsf{Dec}(ASK_i, APK_i, c_{A,i})$ and sends $s_{A,i}$ to Bob.
- Bob recovers polynomial $P_A(x)$, extracts s_A, and computes signature σ_{PK}, i.e., $\sigma_{PK} \leftarrow \mathcal{S}.\mathsf{Sig}(s_A + s_B, \mathsf{T}, PK)$. Bob broadcasts σ_{PK} and claims the coins.

Figure 3.7: Group Escrow via encrypt-and-swap

Chapter 4

Escrow Protocol via VES

In this chapter, we describe our proposed framework to build escrow protocols from VES. The concrete construction of escrow via VES is described in Section 4.1. We analyze our protocol in terms of security, privacy, and TTP involvement in Section 4.2. A detailed comparison with the existing approaches is given in Section 4.3.

4.1 Construction

As mentioned, using verifiably encrypted signature scheme alone is insufficient for an escrow protocol. Specifically, if Alice claims the coins after Bob performs his duty (by signing a new transaction which also claims the coin), Bob may not be able to claim the coins even if he obtains Alice's signature. We use the multi-signature mechanism and the time-lock mechanism to solve this problem.

Denote T the transaction from Alice to Bob and T' the refund transaction to Alice. In the deposit phase, the redeem conditions are set as either (a) Alice and Bob publish multi-signatures on T, denoted by $\{\texttt{Signature\_A + Signature\_B}\}$, or (b) Alice publishes a signature on T' after t blocks, denoted by $\{\texttt{Signature\_T}' + \texttt{t}\}$.

Let $(\mathsf{KeyGen}, \mathsf{AdjKeyGen}, \mathsf{Sig}, \mathsf{VerVESSig}, \mathsf{VESVer}, \mathsf{Adj})$ a VES scheme with validity, unforgeability, and opacity. The concrete construction of the escrow protocol for Bitcoin is provided in Figure 4.1.

Pre-condition Phase:
Alice runs $(PK_A, SK_A) \leftarrow \mathsf{KeyGen}(SP)$.
Bob runs $(PK_B, SK_B) \leftarrow \mathsf{KeyGen}(SP)$.
The adjudicator runs $(APK, ASK) \leftarrow \mathsf{AdjKeyGen}(SP)$.
Deposit Phase:
Alice performs the following:

- Time-lock the promised coins with redeem conditions: {`Signature_A + Signature_B`} or {`Signature_T′+t`}.
- Run $\sigma'_A \leftarrow \mathsf{VESSig}(\mathsf{T}, SK_A, PK_A, APK)$ and send σ'_A along with block number t to Bob.

Bob performs the following:

- Accept if $1 \leftarrow \mathsf{VESVer}(\mathsf{T}, \sigma'_A, PK_A, APK)$.

Withdrawal Phase:
If Bob fails to perform his duty within block length t,

- Alice submits signature $\sigma_{\mathsf{T}'}$ on transaction T', i.e., $\sigma_{\mathsf{T}'} \leftarrow \mathsf{Sig}(\mathsf{T}', SK_A, PK_A)$, and claims the coins.

If Bob completes his duty within t and Alice broadcasts her part of multi-signature σ_A on transaction T, i.e., $\sigma_A \leftarrow \mathsf{Sig}(\mathsf{T}, SK_A, PK_A)$,

- Bob broadcasts his part of multi-signature σ_B on transaction T, i.e., $\sigma_B \leftarrow \mathsf{Sig}(\mathsf{T}, SK_B, PK_B)$, and claims the coins.

If Bob completes his duty but Alice refuses to broadcast her part of multi-signature,

- Bob broadcasts his part of multi-signature σ_B on transaction T, i.e., $\sigma_B \leftarrow \mathsf{Sig}(\mathsf{T}, SK_B, PK_B)$, and sends σ'_A to the TTP.
- TTP runs $\sigma_A \leftarrow \mathsf{Adj}(m, \sigma', ASK, APK)$ and broadcasts σ_A on behalf of Alice.

Figure 4.1: Escrow via VES

4.2 Security Analysis

In the escrow protocol, the TTP is trusted that it will release Alice's signature only if Bob proves the accomplishment of his duty. We show below that our escrow protocol is secure, private (internally-hiding and dispute-hiding but not externally-hiding), and is passive on withdrawal in terms of TTP involvement. However, it does not achieve anti-DoS.

- *Security against Alice*. By the validity of the verifiably encrypted ECDSA, a valid verifiably encrypted signature σ'_A on transaction T can always be reconstructed into a full signature σ by an honest TTP. Furthermore, the time-lock mechanism prevents Alice from claiming her coins before block number t and thus give sufficient time for Bob to file, and for the TTP to handle, the resolution request.
- *Security against Bob*. By the opacity of the verifiably encrypted ECDSA, it is infeasible for any probabilistic polynomial-time adversary to extract the σ_A on T from σ'_A without the help of the TTP.
- *Security against TTP*. By the unforgeability of the ECDSA and the verifiably encrypted ECDSA, it is infeasible for any probabilistic polynomial-time adversary to construct σ_A without seeing σ'_A.
- *Denial-of-Service (DoS)*. Our scheme does not offer anti-DoS. The TTP can simply refuse to mediate.
- *Internally-hiding*. If no dispute occurs, the TTP is not involved and it will not be able to learn if an escrow transaction has taken place.
- *Partially Externally-hiding*. Due to the unique format of an escrow transaction, an external party may correctly identify an escrow transaction.
- *Dispute-hiding*. An external observer cannot recognize a dispute because the underlying verifiably encrypted ECDSA scheme is resolution-independent.

- *TTP Involvement.* Our protocol is *passive-on-deposit* since the TTP is not involved in the deposit phase. Our protocol is *passive-on-withdrawal* since the TTP is only called upon to handle resolution requests from Bob when there is a dispute.

- *Round Efficiency.* One round of communication is required when Bob deviates from the protocol. Two rounds of communication are required otherwise.

4.3 Comparison with Existing Schemes

Table 4.1 summarizes the properties of escrow protocols for cryptocurrencies. As shown in the table, our approach achieves most of the desirable properties at reasonable costs. Specifically, our escrow protocol is passively involved in both the deposit phase and the withdrawal phase, internally-hiding, dispute-hiding, and relatively efficient. The main drawback of our approach is that it is partially externally-hiding due to the 2-of-3 `Multisig` structure.

Compared with our approach, escrow via 2-of-3 `Multisig` is more efficient in terms of off-chain communication and computational costs, but it is less private and requires more TTP involvement. Specifically, 2-of-3 `Multisig` approach is only partially externally-hiding and partially dispute-hiding due to the 2-of-3 `Multisig` structure. Moreover, it is partially active-on-withdrawal.

Escrow via threshold signature is externally hiding, but it is not internally hiding. Moreover, it requires more off-chain messages and the active participation of the TTP in the deposit phase.

Indeed, our protocol is slightly weaker in terms of privacy compared to the encrypt-and-swap mechanism proposed in [24]. But escrow via encrypt-and-swap requires additional key set-up, several zero-knowledge proofs, and five off-chain messages. In contrast, our protocol is setup-free and only requires a single DL-based zero-knowledge proof and at most two off-chain messages, hence it is more efficient and practical.

As mentioned, we can always enhance the resilience to DoS attacks by incorporating bond or sharing trust among a group of TTPs. But it comes at the costs of efficiency and TTP involvement. Compared with the 2-of-3 `Multisig` approach, the TTP is more heavily involved when it incorporates bond mechanism (i.e., the TTP is actively involved in both the deposit and the withdrawal phase). The privacy is also diminished due to the pay-to-script-hash structure. Group escrow protocols, on the other hand, requires more off-chain communications in the case of disputes.

ZKCP and RSA-puzzle are TTP-free and achieve almost all desirable properties. However, their application scenarios are limited. ZKCP can only exchange some sort of verifiable solution that the solution provider is capable of generating a zero-knowledge proof for a pair (y, c) such that $y = SHA256(k)$, $c = \mathsf{Enc}(s, k)$ where s is a correct solution. RSA-puzzle solver protocol is dedicated to trade RSA puzzles.

Table 4.1: Summary of Escrow Protocols

Escrow Protocol	Security		Privacy			TTP Involvement		No. of off-chain Message Exchanged (N/A&R/T)*	No. of on-chain Transaction
	Secure	Anti-DoS	Internally-Hiding	Externally-Hiding	Dispute-Hiding	Passive-on-Deposit	Passive-on-Withdrawal		
2-of-3 `Multisig`	✓	●	✓	○	○	✓	○	$1/1/1$	2
Threshold Signature	✓	●	●	✓	✓	●	○	$8/8/8$	2
Encrypt-and-Swap	✓	●	✓	✓	✓	✓	○	$5/6/6$	2
Escrow with Bond	✓	✓	●	○	●	●	●	$1/1/1$	4
Group `Multisig`	✓	○	✓	●	●	✓	○	$1/n/n$	2
Group Encrypt-and-Swap	✓	○	✓	✓	✓	✓	○	$5/2n+4/2n+4$	2
ZKCP	✓	✓	✓	✓	✓	✓	✓	$0/0/0$	2
RSA Puzzle Solver	✓	✓	✓	✓	✓	✓	✓	$7/5/7$	2
This paper	✓	●	✓	○	✓	✓	✓	$2/1/2$	2

✓ Achieved / Passive
○ Partially Achieved / Partially Active
● Not Achieved / Active
⋆ Normal /Abort and Refund / TTP completes transfer after adjudication

Chapter 5

Verifiably Encrypted ECDSA

The verifiably encrypted ECDSA signature can be thought of as an encryption of a standard ECDSA signature using the twisted ElGamal encryption scheme, together with a zero-knowledge proof that the plaintext is a valid signature. In this chapter, we present an efficient verifiably encrypted ECDSA signature scheme. The concrete construction is described in Section 5.1 and Section 5.2. The security analysis and efficiency analysis are given in Section 5.3 and Section 5.4.

5.1 The Construction of Verifiably Encrypted ECDSA

Let group $\mathbb{G}$ be a cyclic group of order q and generators g_1, g_2. Define hash function $H : H(x) = SHA_{256}(x) \bmod q$. Let (Q_x, Q_y) be the coordinates of point $Q \in \mathbb{G}$. Define $F(Q) = Q_x \bmod q$. Let $\mathcal{S} = (\mathcal{S}.\mathsf{KeyGen}, \mathcal{S}.\mathsf{Sig}, \mathcal{S}.\mathsf{Ver})$ be the standard ECDSA signature scheme. Let $\mathcal{E} = (\mathcal{E}.\mathsf{KeyGen}, \mathcal{E}.\mathsf{Enc}, \mathcal{E}.\mathsf{Dec})$ be the 1-bit twisted ElGamal encryption scheme. The system parameters are defined as $SP = (\mathbb{G}, g_1, g_2, q, H, F)$. The concrete verifiably encrypted ECDSA scheme is defined as follows:

- KeyGen, Sig, Ver. Same as the $\mathcal{S}.\mathsf{KeyGen}, \mathcal{S}.\mathsf{Sig}, \mathcal{S}.\mathsf{Ver}$.
- $\mathsf{AdjKeyGen}(SP)$. Choose $t \xleftarrow{R} \mathbb{Z}_q$ and compute $T = g_2^t$. It sets $ASK = t, APK =$

T.

- $\mathsf{VESSig}(m, \mathsf{Sig}, APK)$. Run Sig and obtain $\sigma = (r, s)$ with $|s| = l$. From r get point R whose x coordinate is r. Choose $\beta_0, ..., \beta_{l-1} \overset{R}{\leftarrow} \mathbb{Z}_q$ and calculate $c = \mathcal{E}.Enc(s, APK; \beta_0, ..., \beta_{l-1})$. That is, $c = (u_i, v_i)_{i=0}^{l-1}$ where $\{s_i\}_{i=0}^{l-1}$ is the binary representation of s, $u_i = g_1^{s_i} g_2^{\beta_i}$, $v_i = T^{\beta_i}$. Generate a non-interactive zero-knowledge proof π to show that c is the encryption of s under APK. Details of the construction of π will be presented in Section 5.2. The verifiably encrypted signature is set as $\sigma' = (R, c, \pi)$.

- $\mathsf{VESVer}(m, \sigma', PK, APK)$. If π holds, output 1, otherwise, output 0.

- $\mathsf{Adj}(m, \sigma', ASK, APK)$. Compute $s = \mathcal{E}.\mathsf{Dec}(c, ASK, APK)$ and $r = F(R)$. Specifically, parse $c = (u_i, v_i)_{i=0}^{l-1}$. For each i, if $u_i / v_i^{1/t} = g_1$, set $s_i = 1$. Otherwise, set $s_i = 0$. Compute $s = \sum_{i=0}^{l-1} 2^i \cdot s_i$. It returns the ECDSA signature $\sigma = (r, s)$.

5.2 The Construction of π

Conceptually, π is a Σ-protocol that proves the ciphertext is indeed the encryption on an ECDSA signature. Note that instead of encrypting the whole ECDSA signature (r, s), we only encrypt s. Here, including R in σ' will not affect the unforgeability of the verifiably encrypted ECDSA signature since it is just a random group element. Our construction of π can be formalized as:

$$\pi = \mathsf{PK}\left\{\begin{array}{ll} (s, \beta_0, ..., \beta_{l-1}): & \\ & c = \mathcal{E}.\mathsf{Enc}(s, APK; \beta_0, ..., \beta_{l-1}) \\ \wedge & \qquad \mathcal{S}.\mathsf{Ver}(m, (r, s), PK) = 1 \end{array}\right\}.$$

By the concrete construction of the bit-by-bit ElGamal encryption scheme, $c = \mathcal{E}.\mathsf{Enc}(s, APK; \beta_0, ..., \beta_{l-1})$ is equivalent to $u_i = g_1^{s_i} g_2^{\beta_i}$, $v_i = T^{\beta_i}$, $s_i \in \{0, 1\}$, where s_i is the i-th bit of s. $\mathcal{S}.\mathsf{Ver}(m, (r, s), PK) = 1$ means $r = F(g_1^{h \cdot s^{-1}} Q^{r \cdot s^{-1}})$, where $h = H(m)$. Recall that $r = F(R)$, the above equation holds if $R = g_1^{h \cdot s^{-1}} Q^{r \cdot s^{-1}}$. Rearranging the terms, the above equation holds if $R^s = g_1^h Q^r$. Therefore, the concrete relationship to be proven in π is:

$$\pi = \text{PK}\left\{\begin{array}{ll} (s_0, ..., s_{l-1}, \beta_0, ..., \beta_{l-1}) : & \\ \quad u_i = g_1^{s_i} g_2^{\beta_i} \wedge v_i = T^{\beta_i} \wedge s_i \in \{0, 1\} & \\ \wedge \qquad\qquad g_1^h Q^r = R^{\sum_{i=0}^{l-1} s_i \cdot 2^i} & \end{array}\right\}.$$

We can further decompose π into the following proofs:

$$\pi_1 = \text{PK}\left\{\begin{array}{l} (s_0, ..., s_{l-1}, \beta_0, ..., \beta_{l-1}) : \\ \quad u_i = g_1^{s_i} g_2^{\beta_i} \wedge v_i = T^{\beta_i} \wedge s_i \in \{0, 1\} \end{array}\right\}$$

$$\pi_2 = \text{PK}\left\{\begin{array}{l} (s, \gamma) : \\ \qquad u_i^{\sum_{i=0}^{l-1} 2^i} = g_1^s g_2^\gamma \wedge g_1^h Q^r = R^s \end{array}\right\},$$

using witness $\gamma = \sum_{i=0}^{l-1} \beta_i \cdot 2^i$.

π_1 is essentially the Σ-protocol for commitment to $m \in \{0, 1\}$ introduced in [25]. The construction of Σ-protocol π_1 and π_2 is described in Figure 5.1 and Figure 5.2 respectively. We prove that π_1 and π_2 is complete, sound and SHVZK in Appendix A and B. Using Fiat-Shamir heuristic [21] (i.e. replacing the challenge with the hashes of the transcripts), a non-interactive Σ-protocol π is given in Figure 5.3.

$\mathcal{P}(g_1, g_2, T; s_i, \beta_i)$ $\qquad\qquad\qquad$ $\mathcal{V}(g_1, g_2, T, u_i, v_i)$

For $i = \{0, ..., l-1\}$, choose $\rho_{s_i}, \rho_{\beta_i}, \rho_i \overset{R}{\leftarrow} \mathbb{Z}_q$.
Compute $A_{i,1} = g_1^{\rho_{s_i}} g_2^{\rho_{\beta_i}}, A_{i,2} = T^{\rho_{\beta_i}}$,
$A_{i,3} = g_1^{s_i \rho_{s_i}} g_2^{\rho_i}, A_{i,4} = T^{\rho_i}$.

$$\xrightarrow[A_{i,3}, A_{i,4}]{A_{i,1}, A_{i,2}}$$

$x \overset{R}{\leftarrow} \{0,1\}^*$

$$\xleftarrow{x}$$

Compute $z_{i,1} = s_i x + \rho_{s_i}$,
$z_{i,2} = \beta_i x + \rho_{\beta_i}, z_{i,3} = \beta_i (x - z_{i,1}) + \rho_i$

$$\xrightarrow{z_{i,1}, z_{i,2}, z_{i,3}}$$

Accept if and only if:
$A_{i,1} u_i^x = g_1^{z_{i,1}} g_2^{z_{i,2}}$,
$A_{i,2} v_i^x = T^{z_{i,2}}$,
$A_{i,3} u_i^{x - z_{i,1}} = g_2^{z_{i,3}}$,
$A_{i,4} v_i^{x - z_{i,1}} = T^{z_{i,3}}$.

Figure 5.1: The construction of π_1

5.3 Security Analysis

Theorem 1. *The verifiably encrypted ECDSA signature scheme is secure against existential forgery if (a) the ECDSA signature scheme $\mathcal{S}$ is existentially unforgeable, and (b) the soundness and SHVZK of π hold.*

Proof. Let $\mathcal{A}$ be a verifiably encrypted ECDSA forger algorithm. $\mathcal{A}$ makes at most q_S queries to O_{VESSig}, and finally, outputs a verifiably encrypted signature σ'^* on message m^*. We construct a forger algorithm $\mathcal{F}$ for the underlying ECDSA scheme. $\mathcal{F}$ is given the public key PK, and has access to the ECDSA signing oracle. $\mathcal{F}$ simulates the challenger and interacts with $\mathcal{A}$ as follows:

- $\texttt{Setup}$. $\mathcal{F}$ runs AdjKeyGen and generates an adjudicator's key pair (APK, ASK). $\mathcal{A}$ is given (APK, ASK, PK).

$$\begin{array}{lcl}
\mathcal{P}(g_1, g_2, R; s, \gamma) & & \mathcal{V}(g_1, g_2, U, R, Q, h, r) \\
\text{Choose } \rho_s, \rho_\gamma \xleftarrow{R} \mathbb{Z}_q. & & \\
\text{Compute } Y_1 = g_1^{\rho_s} g_2^{\rho_\gamma}, Y_2 = R^{\rho_s}. & & \\
 & \xrightarrow{Y_1, Y_2} & \\
 & & \text{Choose } x \xleftarrow{R} \{0,1\}^* \\
 & \xleftarrow{x} & \\
\text{Compute } z_1 = \rho_s - xs, z_2 = \rho_\gamma - x\gamma. & & \\
 & \xrightarrow{z_1, z_2} & \\
 & & \text{Accept if and only if} \\
 & & Y_1 = U^x g_1^{z_1} g_2^{z_2}, \\
 & & Y_2 = (g_1^h Q^r)^x R^{z_1}.
\end{array}$$

Figure 5.2: The construction of π_2

Generation of π. $\mathcal{P}$ performs the following:

1. For $i = \{0, ..., l-1\}$, choose $\rho_{s_i}, \rho_{\beta_i}, \rho_i \xleftarrow{R} \mathbb{Z}_q$ and compute $A_{i,1} = g_1^{\rho_{s_i}} g_2^{\rho_{\beta_i}}$, $A_{i,2} = T^{\rho_{\beta_i}}$, $A_{i,3} = g_1^{s_i \rho_{s_i}} g_2^{\rho_i}$, $A_{i,4} = T^{\rho_i}$.
2. Choose $\rho_k, \rho_\gamma \xleftarrow{R} \mathbb{Z}_q$ and computes $Y_1 = g_1^{\rho_s} g_2^{\rho_\gamma}, Y_2 = R^{\rho_s}$.
3. Compute the hash value of the transcripts, i.e., $x = H(Y_1 || Y_2 || A_{0,1} || ... || A_{(l-1),4})$.
4. For $i = \{0, ..., l-1\}$, compute $z_{i,1} = s_i x + \rho_{s_i}$, $z_{i,2} = \beta_i x + \rho_{\beta_i}$, $z_{i,3} = \beta_i (x - z_{i,1}) + \rho_i$.
5. Compute $\gamma = \sum_{i=0}^{l-1} \beta_i \cdot 2^i$, $z_1 = \rho_k - xk$, $z_2 = \rho_\gamma - x\gamma$.
6. Parse π as $(z_{i,1}, z_{i,2}, z_{i,3}, i = \{0, ..., l-1\}, z_1, z_2)$ and send it to $\mathcal{V}$.

Verification of π. Upon receiving π from $\mathcal{P}$, $\mathcal{V}$ performs the following:

1. For $i = \{0, ..., l-1\}$, compute $A_{i,1} = \frac{g_1^{z_{i,1}} g_2^{z_{i,2}}}{u_i^x}$, $A_{i,2} = \frac{T^{z_{i,2}}}{v_i^x}$, $A_{i,3} = \frac{g_2^{z_{i,3}}}{u_i^{x - z_{i,1}}}$, $A_{i,4} = \frac{T^{z_{i,3}}}{v_i^{x - z_{i,1}}}$.
2. Compute $Y_1 = U^x g_1^{z_1} g_2^{z_2}$, $Y_2 = (g_1^h Q^r)^x R^{z_1}$.
3. Finally, $\mathcal{V}$ returns Accept if $x = H(Y_1 || Y_2 || A_{0,1} || A_{0,2} ... || A_{(l-1),4})$ holds.

Figure 5.3: The construction of π

- $\mathbf{O}_{\mathbf{VESig}}$ **Query**. $\mathcal{A}$ requests a verifiably encrypted signature on m. $\mathcal{F}$ queries its signing oracle with m and obtains $\sigma = (r, s)$. Let $R = (r, y)$ where y is computed by plugging r into the ECDSA elliptic curve. $\mathcal{F}$ computes $c = \mathcal{E}.\mathsf{Enc}(s, APK)$ and generates a proof π. $\mathcal{F}$ gives $\sigma' = (R, c, \pi)$ to $\mathcal{A}$.

- **Output**. Finally, $\mathcal{A}$ outputs a valid verifiably encrypted signature $\sigma'^* = (R^*, c^*, \pi^*)$ on message m^*. $\mathcal{F}$ computes $s^* = \mathcal{E}.\mathsf{Dec}(\sigma'^*, ASK)$ and $r^* = F(R^*)$. By the soundness of π^*, we have $R^{s^*} = g^{h^*}Q^{r^*}$. That is, $R^* = g^{h^*s^{*-1}}Q^{r^*s^{*-1}}$. Applying F function to both sides of the equation, we have $F(R^*) = r^* = F(g^{h^*s^{*-1}}Q^{r^*s^{*-1}})$. Define $\sigma^* = (r^*, s^*)$. It is easy to see that $\mathsf{Ver}(m^*, \sigma^*, PK) = 1$. Finally, $\mathcal{F}$ outputs (σ^*, m^*) and wins its own game.

Theorem 2. *The verifiably encrypted ECDSA signature scheme is secure against extraction if (a) the soundness and SHVZK of π hold; (b) the unforgeability of the ECDSA signature scheme $\mathcal{S}$ holds; and (c) the twisted ElGamal encryption scheme $\mathcal{E}$ is IND-CPA secure.*

Proof. Let $\mathcal{B}$ be verifiably encrypted signature extractor algorithm. $\mathcal{B}$ makes at total q_S queries to O_{VESSig}, q_A queries to O_{Adj}, and finally, outputs a valid ECDSA signature σ^* on message m^*. Note that $\mathcal{B}$ must have queried O_{VESSig} with m^*, otherwise $\mathcal{B}$ breaks the unforgeability of ECDSA. Let $\mathsf{Pr}_i[\mathsf{Succ}]$ indicate the success probability of $\mathcal{B}$ wins in $\mathbf{Game}_i$. The proof for opacity is described as below:

- $\mathbf{Game}_0$ is the same as the opacity model defined in Section 3.7.

- $\mathbf{Game}_1$ is the same as the opacity model defined in Section 3.7 but only $\mathcal{B}$ aborts if $m^* \neq m_i$, $i \overset{R}{\leftarrow} \{1, ..., q_S\}$. As mentioned, $\mathcal{B}$ must have queried O_{VESSig} with m^* before outputting σ^*. The probability that $m^* = m_i$ is $\frac{1}{q_S}$. Therefore,

$$\mathsf{Pr}_1[\mathsf{Succ}] = \frac{1}{q_S} \times \mathsf{Pr}_0[\mathsf{Succ}]$$

- $\mathbf{Game}_2$ is derived by modifying the responses to the O_{Adj} queries. For any adjudication query on (σ', m), where $\sigma' = (R, c, \pi)$, run the soundness extractor to obtain s, compute $r = F(R)$, and finally return $\sigma = (r, s)$. By the soundness of π, $\mathbf{Game}_1$ and $\mathbf{Game}_2$ are indistinguishable. Let Sound denote the event breaking the soundness of proof π, we have:

$$|\mathsf{Pr}_2[\mathsf{Succ}] - \mathsf{Pr}_1[\mathsf{Succ}]| \leq \mathsf{Pr}[\mathsf{Sound}]$$

- $\mathbf{Game}_3$ is derived by modifying responses to the i-th $\mathsf{O}_{\mathsf{VESSig}}$ query. Chooses $k_i \overset{R}{\leftarrow} \mathbb{Z}_q$, compute $R_i = g_i^k$, $w_i = \mathcal{E}.\mathsf{Enc}(0, APK)$, and generate a proof π_i using the zero-knowledge simulator. By the SHVZK of π, π_i is indistinguishable from a real proof. For the i-th query to O_{VESSig}, response with $\sigma'_i = (R_i, c_i, \pi_i)$.

  We claim that $|\mathsf{Pr}_3[\mathsf{Succ}] - \mathsf{Pr}_2[\mathsf{Succ}]$ is also negligible. Let $\mathcal{B}'$ attacking the IND-CPA secure encryption scheme $\mathcal{E}$. $\mathcal{B}'$ runs $\mathcal{B}$ as follows: $\mathcal{B}'$ sets $m_0 = \sigma_i, m_1 = 0$ as the challenge messages. Upon receiving the challenge ciphertext c^*, $\mathcal{B}'$ forwards $\sigma'_i = (R_i, c^*, \pi_i)$ to $\mathcal{B}$ as the response to the i-th O_{VESSig} query. Here, if $c^* = \mathsf{Enc}(APK, \sigma)$, the view of $\mathcal{B}$ is the same as in $\mathbf{Game}_2$. While if $c^* = \mathsf{Enc}(APK, 0)$, the view of $\mathcal{B}$ is the same as in $\mathbf{Game}_3$. Finally, if $\mathcal{B}$ outputs a valid signature on m^* such that $m_i = m^*$, $\mathcal{B}'$ outputs $b = 0$. Otherwise, $b = 1$. Let ZK denote the event breaking the SHVZK of proof π and IND denote the event breaking the IND-CPA security of ECDSA, we have :

$$|\mathsf{Pr}_3[\mathsf{Succ}] - \mathsf{Pr}_2[\mathsf{Succ}]| \leq |\mathsf{Pr}[\mathsf{ZK}]| + |\mathsf{Pr}[\mathsf{IND}]|$$

  Moreover, given that σ^* is independent of m_0, m_1, by the unforgeability of ECDSA, $\mathsf{Pr}_3[Succ]$ is also negligible.

To sum up, we have:

$$\begin{aligned}\mathsf{Pr}_1[\mathsf{Succ}] &\leq |\mathsf{Pr}_1[\mathsf{Succ}] - \mathsf{Pr}_2[\mathsf{Succ}]| + |\mathsf{Pr}_2[\mathsf{Succ}] \\ &\quad -\mathsf{Pr}_3[\mathsf{Succ}]| + |\mathsf{Pr}_3[\mathsf{Succ}]| \\ &\leq |\mathsf{Pr}_1[\mathsf{Sound}| + |\mathsf{Pr}[\mathsf{ZK}]| + |\mathsf{Pr}[\mathsf{IND}]| \\ &\quad +|\mathsf{Pr}_3[\mathsf{Succ}]|\end{aligned}$$

Since that $|\mathsf{Pr}[\mathsf{Sound}]|$, $|\mathsf{Pr}[\mathsf{ZK}]|$, $|\mathsf{Pr}[\mathsf{IND}]|$, and $|\mathsf{Pr}_3[\mathsf{Succ}]|$ are negligible, we have $\mathsf{Pr}_1[\mathsf{Succ}]$ is negligible. As a result, $\mathsf{Pr}_0[\mathsf{Succ}] = q_S \cdot \mathsf{Pr}_1[\mathsf{Succ}]$ is negligible. The verifiably encrypted ECDSA scheme is secure against extraction.

5.4 Efficiency Analysis

Let $\mathbf{G}$ denote an element in $\mathbb{G}$ and $\mathbf{Z}$ denote an element in $\mathbb{Z}_q$. By $\mathbf{E}$ we mean an exponentiation operation. Table 5.1 summarizes the costs of the verifiably encrypted ECDSA scheme in terms of space complexity, time complexity, and estimated runtime based on the benchmark of $\mathbf{E}$. For time complexity, we only consider the most expensive operation, namely, exponentiation operation over $\mathbb{G}$. Using Secp256k1 parameters, which offers security at 256-bit level, the average runtime of an exponentiation operation is 1.983 ms (MacBook Pro 2017 with Intel Core i5-7267U CPU and 16 GB of RAM). The estimated runtime is calculated by multiplying the maximum number of exponentiation operations and the average runtime of a single exponentiation operation.

Table 5.1: Efficiency of the Verifiably Encrypted ECDSA[54]

	Space Complexity	Time Complexity	Estimated Runtime
Sig	$2\mathbf{Z}$	$1\mathbf{E}$	$1.983\mathbf{ms}$
Ver	$1\mathbf{Z}$	$2\mathbf{E}$	$3.966\mathbf{ms}$
VESSig^1	$513\mathbf{G}$	$512\mathbf{E}$	$1015.296\mathbf{ms}$
VESSig^2	$777\mathbf{Z}$	$1539\mathbf{E}$	$3051.837\mathbf{ms}$
VESVer	$1\mathbf{Z}$	$2310\mathbf{E}$	$4580.730\mathbf{ms}$
Adj	$2\mathbf{Z}$	$256\mathbf{E}$	$507.648\mathbf{ms}$

5.5 Summary

This chapter introduces a practical verifiably encrypted ECDSA signature scheme based on Σ-protocol, which gives rise to a practical escrow protocol for ECDSA-based cryptocurrencies. The performance of the escrow protocol for Bitcoin resulting from our verifiably encrypted ECDSA signature scheme is analyzed and compared in Chapter 7. However, the main drawback of our scheme is its high bandwidth requirement for off-chain communication. This is due to its use of a rather inefficient bit-by-bit encryption and its corresponding zero-knowledge proof in the construction of VES for ECDSA. Furthermore, it is only applicable to cryptocurrencies adopting ECDSA to authorise its transaction (such as Bitcoin). As an improvement, Chapter 6 introduces a generic construction of verifiably encrypted signature scheme covering both ECDSA and its variants (which we refer to as EdDAS-like signatures) with significantly fewer bandwidth costs.

Chapter 6

Verifiably Encrypted EdDSA-like Signature Scheme

We define EdDSA-like signature, an abstraction of many variants of ECDSA, and describe verifiably encrypted EdDSA-like signature. Similar to the verifiably encrypted ECDSA, the verifiably encrypted EdDSA-like signature scheme can be thought of as an encryption of a standard EdDSA-like signature using the twisted ElGamal encryption scheme, together with a zero-knowledge proof that the plaintext is a valid signature. We describe EdDSA-like signature in Section 6.1. Section 6.2 and Section 6.3 presents the concrete construction of the verifiably encrypted EdDSA-like signature scheme. The security analysis and efficiency analysis are given in Section 6.4 and Section 6.5, respectively. Looking ahead, our construction uses an improved way to encrypt signature component s, and a more efficient zero-knowledge proof. Thus, the space complexity of the verifiably encrypted EdDSA-like signature is better than the verifiably encrypted ECDSA presented in the last chapter. Since ECDSA is also an EdDSA-like signature, the result presented in this chapter can be used to improve the space complexity of our escrow protocol for Bitcoin described in Chapter 4.

6.1 EdDSA-like Signature

Definition 10. *(EdDSA-like signature scheme). Let $\mathbb{G}$ be a cyclic group of order q. Suppose there is a DL-based signature scheme* $(\mathsf{KeyGen}, \mathsf{Sig}, \mathsf{Ver})$ *on group $\mathbb{G}$, i.e., $SP \leftarrow$* $\mathsf{KeyGen}(1^\lambda)$*, $\sigma \leftarrow \mathsf{Sig}(m, SK, PK; k)$, $0/1 \leftarrow \mathsf{Ver}(m, c, PK, SK)$. We say such a scheme EdDSA-like if*

1. *$\sigma = (\sigma_1, \sigma_2) \wedge \sigma_2 \in \mathbb{Z}_q$.*
2. *There exists probabilistic polynomial algorithms $\mathsf{Sig}_1, \mathsf{Sig}_2$ such that $\sigma_1 \leftarrow \mathsf{Sig}_1([m]; k)$, $\sigma_2 \leftarrow \mathsf{Sig}_2\ (SK, m; k)$.*
3. *σ can be verified in the form of $A = B^{\sigma_2}$ where $(A, B) = \mathsf{D}(SP, PK, \sigma_1, m)$ for some deterministic polynomial-time algorithm D. We call D the base generation algorithm for the EdDSA-like signature.*

EdDSA-like signature schemes include, but are not limited to, EdDSA, Schnorr signature, ECDSA, ECGDSA, GOST, and SM2. It is obvious that EdDSA, and Schnorr signature scheme are EdDSA-like. We show that ECDSA is also EdDSA-like. Given an EdDSA-like signature $\sigma = (r, s)$, σ can be verified by recovering group element R from its x-coordinate r and checking if $g_1^h Q^r = R^s$. Hence, ECDSA is also EdDSA-like. Similarly, ECGDSA, GOST, and SM2 are EdDSA-like.

6.2 The Construction of Verifiably Encrypted EdDSA-like Signature

Let $\mathcal{E} = (\mathcal{E}.\mathsf{KeyGen}, \mathcal{E}.\mathsf{Enc}, \mathcal{E}.\mathsf{Dec})$ denote twisted ElGamal encryption (with 32-bit message space) with system parameters $SP_{\mathcal{E}}$. Let $\mathcal{S} = (\mathcal{S}.\mathsf{KeyGen}, \mathcal{S}.\mathsf{Sig}, \mathcal{S}.\mathsf{Ver})$ be an EdDSA-like signature scheme with system parameters $SP_{\mathcal{S}}$ and D the base generation algorithm for $\mathcal{S}$. The system parameter SP of the verifiably encrypted signature scheme is defined as $SP = \{SP_{\mathcal{S}}, SP_{\mathcal{E}}\}$. A generic two-phase verifiably encrypted EdDSA-like signature scheme (KeyGen, $\mathsf{AdjKeyGen}$, Sig, Ver, VESSig, VESVer, Adj) is defined as below.

- $\mathsf{KeyGen}, \mathsf{Sig}, \mathsf{Ver}$. Same as $\mathcal{S}.\mathsf{KeyGen}, \mathcal{S}.\mathsf{Sig}, \mathcal{S}.\mathsf{Ver}$.
- $\mathsf{AdjKeyGen}(SP)$. Run $(T, t) \leftarrow \mathcal{E}.\mathsf{KeyGen}(SP_{\mathcal{E}})$. Let $APK = T, ASK = t$.
- $\mathsf{VESSig}(m, SK, PK, APK)$. Run $(\sigma_1, \sigma_2) \leftarrow \mathcal{S}.\mathsf{Sig}(m,\ SK_{\mathcal{S}}, PK; k)$. Assume $|\sigma_2| = n$. Let $l = n/32$. Choose $\beta_0, ..., \beta_{l-1} \in Z_q$ and compute $c \leftarrow \mathcal{E}.\mathsf{Enc}$ $(\sigma_2,\ T;\ \beta_0, ..., \beta_{l-1})$. That is, $c = (u_i, v_i)_{i=0}^{l-1}$ where $\sigma_2 = \sum_{i=0}^{l-1} \sigma_{2,i} \cdot 2^{32 \cdot i}$, $\sigma_{2,i} \in \{0, ..., 2^{32} - 1\}$, $u_i = g_1^{\sigma_{2,i}} g_2^{\beta_i}$, $v_i = T^{\beta_i}$. Generate a non-interactive zero-knowledge proof π to show that c is the encryption of σ_2 under APK. Details of the construction of π will be presented in Section 6.3. The verifiably encrypted signature is set as $\sigma' = (\sigma_1, c, \pi)$.
- $\mathsf{VESVer}(m, \sigma', PK, APK)$. Output 1 if the validity of π holds.
- $\mathsf{Adj}(m, \sigma', ASK, APK)$. Run $\sigma_2 \leftarrow \mathcal{E}.\mathsf{Dec}(c, APK, ASK)$ and output $\sigma = (\sigma_1, \sigma_2)$.

6.3 The Construction of π

Conceptually, π is a combination of Σ-protocol and Bulletproofs that proves the ciphertext is indeed the encryption on an EdDSA-like signature. Note that instead of encrypting the whole EdDSA-like signature (σ_1, σ_2), we only encrypt σ_2. Our construction of π can be formalized as:

$$\pi = \mathrm{PK}\left\{ \begin{array}{ll} (\sigma_2, \beta_0, ..., \beta_{l-1}) : & \\ & c = \mathcal{E}.\mathsf{Enc}(\sigma_2, APK; \beta_0, ..., \beta_{l-1}) \\ \wedge & \qquad \mathcal{S}.\mathsf{Ver}(m, (\sigma_1, \sigma_2), PK) = 1 \end{array} \right\}.$$

By the concrete construction of the twisted ElGamal encryption scheme, $c = \mathcal{E}.\mathsf{Enc}(s, APK; \beta_0, ..., \beta_{l-1})$ is equivalent to $u_i = g_1^{\sigma_{2,i}} g_2^{\beta_i}$, $v_i = T^{\beta_i}$, $\sigma_{2,i} \in \{0, ..., 2^{32} - 1\}$, where

$\sigma_2 = \sum_{i=0}^{l-1} \sigma_{2,i} \cdot 2^{32 \cdot i}$. And $\mathcal{S}.\mathsf{Ver}(m, (\sigma_1, \sigma_2), PK) = 1$ is equivalent to $A = B^{\sigma_2}$, where $A, B = \mathsf{D}(SP, PK, \sigma_1, m)$. Therefore, the concrete relationship to be proven in π is:

$$\pi = \text{PK}\left\{\begin{array}{lc} (\sigma_{2,0}, ..., \sigma_{2,l-1}, \beta_0, ..., \beta_{l-1}) : & \\ & u_i = g_1^{\sigma_{2,i}} g_2^{\beta_i} \wedge \sigma_{2,i} \in [0, 2^{32}-1] \\ \wedge & v_2 = T^{\beta_i} \wedge A = B^{\sum_{i=0}^{l-1} \sigma_{2,i} \cdot 2^{32 \cdot i}} \end{array}\right\}.$$

We can further decompose π into the following proofs:

$$\pi_\Lambda = \text{PK}\left\{\begin{array}{l} (\sigma_{2,0}, ..., \sigma_{2,l-1}, \beta_0, ..., \beta_{l-1}) : \\ \quad u_i = g_1^{\sigma_{2,i}} g_2^{\beta_i} \wedge \sigma_{2,i} \in [0, 2^{32}-1]\} \end{array}\right\}$$

$$\pi_\Sigma = \text{PK}\left\{\begin{array}{l} (\sigma_2, \gamma) : \\ \qquad \prod_{i=0}^{l-1} u_i^{2^{32 \cdot i}} = g_1^{\sigma_2} g_2^{\gamma} \wedge \prod_{i=0}^{l-1} v_i^{2^{32 \cdot i}} = T^{\gamma} \wedge \ A = B^{\sigma_2} \end{array}\right\},$$

using witness $\gamma = \sum_{i=0}^{l-1} \beta_i \cdot 2^i$.

We realize the verifiably encrypted signature scheme by invoking aggregated Bulletproof for Pedersen commitment to generate a proof π_Λ and Σ-protocol to generate a proof π_Σ [25]. The construction of proof π_Σ and π_Λ is shown in Figure 6.1 and Figure 6.2. For the security proof of π_Λ, please refer to [10]. We omit the security proof of π_Σ, which is essentially the same as π_2 provided in Appendix B. Using Fiat-Shamir heuristic [21] (i.e. replacing the challenge with the hashes of the transcripts), a non-interactive Σ-protocol π is given in Figure 6.3.

$\mathcal{P}(g_1, g_2, \{u_i\}_{i=0}^{l-1}; \{\sigma_{2,i}\}_{i=0}^{l-1}, \gamma)$ $\qquad$ $\mathcal{V}(g_1, g_2, \{u_i\}_{i=0}^{l-1})$

Let $\mathbf{h}, \mathbf{k}$ two sets of independent generators in $\mathbb{G}^n$.
Choose $\alpha, \rho \overset{R}{\leftarrow} \mathbb{Z}_q, \rho_{\mathbf{L}}, \rho_{\mathbf{R}} \overset{R}{\leftarrow} \mathbb{Z}_q^n$

Set $\sigma_{\mathbf{L}} \in \{0,1\}^n$ $s.t.$ $\langle \sigma_{\mathbf{L}[32\cdot j:32\cdot(j+1)-1]}, \mathbf{2}^n \rangle = \sigma_{2,i}$
for $j = \{0, ..., l-1\}$, and $\sigma_{\mathbf{R}} = \sigma_{\mathbf{L}} - \mathbf{1}^n$.
Compute $A = g_2^{\alpha} \mathbf{h}^{\sigma_{\mathbf{L}}} \mathbf{k}^{\sigma_{\mathbf{R}}}, S = g_2^{\rho} \mathbf{h}^{\rho_{\mathbf{L}}} \mathbf{k}^{\rho_{\mathbf{R}}}$.

$\xrightarrow{A,S}$

$y, z \overset{R}{\leftarrow} \mathbb{Z}_q$

$\xleftarrow{y,z}$

Define polynomials $l(X), r(X), t(X)$ as
$l(X) = (\sigma_{\mathbf{L}} - z \cdot 1^n) + \rho_{\mathbf{L}} \cdot X$,
$r(X) = y^n \circ (\sigma_{\mathbf{R}} + z \cdot 1^n + \rho_{\mathbf{R}} \cdot X) +$
$\sum_{j=0}^{l-1} z^{2+j} \cdot (\mathbf{0}^{32\cdot j} || \mathbf{2}^{32} || \mathbf{0}^{32\cdot(l-j-1)})$,
Define $t(x)$ the inner product of $l(X), r(X)$ as
$t(X) = \langle l(X), r(X) \rangle = t_0 + t_1 X + t_2 X^2$,
$(i.e.,\ t_0 = \sigma_{2,0} \cdot z^2 + \sigma_{2,1} \cdot z^3 ... + \sigma_{2,l-1} \cdot 2^{l+1} + \delta(y,z)$,
where $\delta(y,z) = (z - z^2)\langle \mathbf{1}^n, \mathbf{y}^n \rangle - \sum_{j=0}^{l-1} z^{j+3} \cdot \langle \mathbf{1}^{32}, \mathbf{2}^{32} \rangle)$)
Choose $\alpha_i \overset{R}{\leftarrow} \mathbb{Z}_q, \tau_1, \tau_2 \overset{R}{\leftarrow} \mathbb{Z}_q$
Compute $T_1 = g_1^{t_1} g_2^{\tau_1}, T_2 = g_1^{t_2} g_2^{\tau_2}$.

$\xrightarrow{T_1,T_2}$

$x \overset{R}{\leftarrow} \mathbb{Z}_q$

$\xleftarrow{x}$

Set $\mathbf{k}' = (k_1, k_2^{y^{-1}}, ..., k_n^{y^{-n+1}})$.
Compute $\mathbf{l} = l(x), \mathbf{r} = r(x), \hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle$,
$\tau = \tau_2 \cdot x^2 + \tau_1 \cdot x + \sum_{j=0}^{l-1} \beta_j \cdot z^{j+2}$,
$\mu = \alpha + \rho \cdot x$,
$P = A \cdot S^x \cdot \mathbf{h}^{-z} \cdot \mathbf{k}'^{z\cdot\mathbf{y}^n} \prod_{j=0}^{l-1} \mathbf{k}'^{z^{j+2}\cdot\mathbf{2}^{32}}_{[32\cdot j:32\cdot(j+1)-1]}$.
Run Bünz's inner product argument on$(\mathbf{h}, \mathbf{k}', P \cdot g_2^{-\mu}, \hat{t}; \mathbf{l}, \mathbf{r})$
and obtain a proofπ_{ipa}[1].
Set $\pi = (A, S, T, \hat{t}_x, \tau, \mu, \pi_{ipa})$.

$\xrightarrow{\pi}$

Compute
$y = H(A, S)$,
$z = H(A, S, y)$,
$x = H(A, S, y, z, T_1, T_2)$.
Define $\mathbf{u} = (u_0, ..., u_{l-1})$.
Accept if
π_{ipa} holds,
and $g_1^{\hat{t}} g_2^{\tau} = \mathbf{u}^{z^2\cdot\mathbf{z}^l} g_1^{\delta(y,z)} T_1^x T_2^{x^2}$.

Figure 6.1: The construction of π_Λ

$\mathcal{P}(\{u_i\}_{i=0}^{l-1}, \{v_i\}_{i=0}^{l-1}, g_1, g_2, T, B; \sigma_2, \gamma)$		$\mathcal{V}(g_1, g_2, T, A, B)$
Choose $\rho_{\sigma_2}, \rho_\gamma \overset{R}{\leftarrow} \mathbb{Z}_q$. Compute $Y_1 = g_1^{\rho_{\sigma_2}} g_2^{\rho_\gamma}$, $Y_2 = T^{\rho_\gamma}$, $Y_3 = B^{\rho_{\sigma_2}}$.		
	$\xrightarrow{Y_1, Y_2, Y_3}$	
		$x \overset{R}{\leftarrow} \{0,1\}^*$
	$\xleftarrow{x}$	
Compute $z_1 = \rho_{\sigma_2} - x\sigma_2$, $z_2 = \rho_\gamma - x\gamma$.		
	$\xrightarrow{z_1, z_2}$	
		Compute $U = \prod_{i=0}^{l-1} u_i^{2^{32 \cdot i}}, V = \prod_{i=0}^{l-1} v_i^{2^{32 \cdot i}}$. Accept if $Y_1 = U^x g_1^{z_1} g_2^{z_2}, Y_2 = V^x T^{z_2}$, $Y_3 = A^x B^{z_1}$

Figure 6.2: The construction of π_Σ

Construction of π. $\mathcal{P}$ performs the following:

1. Let $\mathbf{h}, \mathbf{k}$ two sets of independent generators in $\mathbb{G}^n$. Choose $\alpha, \rho \overset{R}{\leftarrow} \mathbb{Z}_q$ and $\rho_{\mathbf{L}}, \rho_{\mathbf{R}} \overset{R}{\leftarrow} \mathbb{Z}_q^n$. Let $\sigma_{\mathbf{L}} \in \{0,1\}^n$ $s.t. \langle \sigma_{\mathbf{L}[32\cdot j:32\cdot(j+1)-1]}, \mathbf{2}^n \rangle = \sigma_{2,j}$ for $j = \{0, ..., l-1\}$. Set $\sigma_{\mathbf{R}} = \sigma_{\mathbf{L}} - 1^n$. Compute $A = g_2^{\alpha} \mathbf{h}^{\sigma_{\mathbf{L}}} \mathbf{k}^{\sigma_{\mathbf{R}}}$ and $S = g_2^{\rho} \mathbf{h}^{\rho_{\mathbf{L}}} \mathbf{k}^{\rho_{\mathbf{R}}}$.

2. Computes $y = H(A, S)$, $z = H(A, S, y)$.

3. Define $l(X), r(X), t(X)$ as
   $l(X) = (\sigma_{\mathbf{L}} - z \cdot 1^n) + \rho_{\mathbf{L}} \cdot X$,
   $r(X) = y^n \circ (\sigma_{\mathbf{R}} + z \cdot 1^n + \rho_{\mathbf{R}} \cdot X) + \sum_{j=0}^{l-1} z^{2+j} \cdot (\mathbf{0}^{32\cdot j} || \mathbf{2}^{32} || \mathbf{0}^{32\cdot(l-j-1)})$,
   $t(X) = \langle l(X), r(X) \rangle = t_0 + t_1 X + t_2 X^2$,
   $i.e., t_0 = \sigma_{2,0} \cdot z^2 + \sigma_{2,1} \cdot z^3 ... + \sigma_{2,l-1} \cdot 2^{l+1} + \delta(y,z)$, $\delta(y,z) = (z - z^2)\langle \mathbf{1}^n, \mathbf{y}^n \rangle - \sum_{j=0}^{l-1} z^{j+3} \cdot \langle \mathbf{1}^{32}, \mathbf{2}^{32} \rangle$

4. Choose $\tau_1, \tau_2 \overset{R}{\leftarrow} \mathbb{Z}_q$, and compute $T_1 = g_1^{t_1} g_2^{\tau_1}$, $T_2 = g_1^{t_2} g_2^{\tau_2}$.

5. Choose $\rho_{\sigma_2}, \rho_{\gamma} \overset{R}{\leftarrow} \mathbb{Z}_q$. Compute $Y_1 = g_1^{\rho_{\sigma_2}} g_2^{\rho_{\gamma}}$, $Y_2 = T^{\rho_{\gamma}}$, $Y_3 = B^{\rho_{\sigma_2}}$.

6. Compute $x = H(A, S, y, z, T_1, T_2, Y_1, Y_2, Y_3)$.

7. Define $\mathbf{k}' = (k_1, k_2^{y^{-1}}, ..., k_n^{y^{-n+1}})$. Compute $\mathbf{l} = l(x)$, $\mathbf{r} = r(x)$, $\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle$, $\tau = \tau_2 \cdot x^2 + \tau_1 \cdot x + \sum_{j=0}^{l-1} \beta_{j+1} \cdot z^{j+2}$, $\mu = \alpha + \rho \cdot x$, and $P = A \cdot S^x \cdot \mathbf{h}^{-z} \cdot \mathbf{k}'^{z \cdot \mathbf{y}^n} \prod_{j=0}^{l-1} \mathbf{k}'^{z^{j+2} \cdot \mathbf{2}^{32}}_{[32\cdot j:32\cdot(j+1)-1]}$.

8. Run Bünz's inner product argument on $(\mathbf{h}, \mathbf{k}', P \cdot g_2^{-\mu}, \hat{t}; \mathbf{l}, \mathbf{r})$ and obtain a proof π_{ipa}.

9. Output $\pi = (A, S, T_1, T_2, Y_1, Y_2, Y_3, \hat{t}, \tau, \mu, \pi_{ipa})$.

Veification of π**.** Upon receiving π from $\mathcal{P}$, $\mathcal{V}$ performs the following:

1. Verify π_{ipa}.

2. Compute $y = H(A, S)$, $z = H(A, S, y)$, and $x = H(A, S, y, z, T_1, T_2, Y_1, Y_2, Y_3)$.

3. Define $\mathbf{u} = (u_0, ..., u_{l-1})$.

4. Compute $U = \prod_{i=0}^{l-1} u_i^{2^{32\cdot i}}$, $V = \prod_{i=0}^{l-1} v_i^{2^{32\cdot i}}$. Finally, output Accept if $Y_1 = U^x g_1^{z_1} g_2^{z_2}$, $Y_2 = V^x T^{z_2}$, $Y_3 = A^x B^{z_1}$, and $g_1^{\hat{t}} g_2^{\tau} = \mathbf{u}^{z^2 \cdot \mathbf{z}^l} g_1^{\delta(y,z)} T_1^x T_2^{x^2}$,

Figure 6.3: The Construction of π

6.4 Security Analysis

Theorem 3. *The verifiably encrypted EdDSA-like signature scheme is secure against existential forgery if (a) the EdDSA-like signature scheme $\mathcal{S}$ is existentially unforgeable, and (b) the soundness and SHVZK of π hold.*

Proof. Let $\mathcal{A}$ be a verifiably encrypted EdDSA-like signature forger algorithm. $\mathcal{A}$ makes at most q_S queries to O_{VESSig}, and finally, outputs a verifiably encrypted signature σ'^* on message m^*. We construct a forger algorithm $\mathcal{F}$ for the underlying EdDSA-like scheme. Given the public key PK, $\mathcal{F}$ simulates the challenger and interacts with $\mathcal{A}$ as follows:

- **Setup**. $\mathcal{F}$ runs $\mathsf{AdjKeyGen}$ and generates an adjudicator's key pair (APK, ASK). $\mathcal{A}$ is given (APK, ASK, PK).
- **O$_{\mathsf{VESSig}}$ Query**. $\mathcal{A}$ requests a verifiably encrypted signature on m. $\mathcal{F}$ picks a random σ_1, runs soundness extractor to obtain σ_2, and generates a proof π using zero-knowledge simulator. $\mathcal{F}$ gives $\sigma' = (\sigma_1, c, \pi)$ to $\mathcal{A}$.
- **Output**. Finally, $\mathcal{A}$ outputs a valid verifiably encrypted signature $\sigma'^* = (\sigma_1^*, c^*, \pi^*)$ on message m^*. $\mathcal{F}$ computes $\sigma_2^* = \mathcal{E}.\mathsf{Dec}(\sigma'^*, ASK)$. Set $\sigma^* = (\sigma_1^*, \sigma_2^*)$. Finally, $\mathcal{F}$ outputs (σ^*, m^*) and wins its own game.

Theorem 4. *The verifiably encrypted EdDSA-like signature scheme is secure against extraction if (a) the soundness and SHVZK of π hold; (b) the unforgeability of the EdDSA-like signature scheme $\mathcal{S}$ holds; and (c) the twisted ElGamal encryption scheme $\mathcal{E}$ is IND-CPA secure.*

Proof. Let $\mathcal{B}$ be verifiably encrypted signature extractor algorithm. $\mathcal{B}$ makes at total q_S queries to O_{VESSig}, q_A queries to O_{Adj}, and outputs a valid EdDSA-like signature σ^* on message m^*. Note that $\mathcal{B}$ must have queried O_{VESSig} with m^*, otherwise $\mathcal{B}$ breaks the unforgeability of transformed EdDSA-like signature scheme. Let $\mathsf{Pr}_i[\mathsf{Succ}]$ indicate the success probability of $\mathcal{B}$ wins in $\mathbf{Game}_i$. The proof for opacity is described as below:

- $\mathbf{Game}_0$ is the same as the opacity model defined in Section 3.7.

- $\mathbf{Game}_1$ is the same as the opacity model defined in Section 3.7 but only $\mathcal{B}$ aborts if $m^* \neq m_i$, $i \overset{R}{\leftarrow} \{1, ..., q_S\}$. As mentioned, $\mathcal{B}$ must have queried O_{VESSig} with m^* before outputting σ^*. The probability that $m^* = m_i$ is $\frac{1}{q_S}$. Therefore,

$$\mathsf{Pr}_1[\mathsf{Succ}] = \frac{1}{q_S} \times \mathsf{Pr}_0[\mathsf{Succ}]$$

- $\mathbf{Game}_2$ is derived by modifying the responses to the O_{Adj} queries. For any adjudication query on (σ', m) where $\sigma' = (\sigma_1, c, \pi)$, run the soundness extractor to obtain σ_2. Finally, return $\sigma = (\sigma_1, \sigma_2)$. Let Sound denote the event breaking the soundness of proof π, we have:

$$|\mathsf{Pr}_2[\mathsf{Succ}] - \mathsf{Pr}_1[\mathsf{Succ}]| \leq \mathsf{Pr}[\mathsf{Sound}]$$

- $\mathbf{Game}_3$ is derived by modifying responses to the i-th $\mathsf{O}_{\mathsf{VESSig}}$ query. Randomly choose $\sigma_{1,i}$. Compute $c_i = \mathcal{E}.\mathsf{Enc}(0, APK)$ and generate a proof π_i using the zero-knowledge simulator. By the SHVZK of π_i, π_i is indistinguishable from a real proof. For the i-th query to O_{VESSig}, return $\sigma'_i = (\sigma_{1,i}, c_i, \pi_i)$.

  We claim that $|\mathsf{Pr}_3[\mathsf{Succ}] - \mathsf{Pr}_2[\mathsf{Succ}]|$ is also negligible. Let $\mathcal{B}'$ attacking the IND-CPA secure encryption scheme $\mathcal{E}$. $\mathcal{B}'$ runs $\mathcal{B}$ as follows: $\mathcal{B}'$ sets $m_0 = \sigma_{2,i}, m_1 = 0$ as the challenge messages. Upon receiving the challenge ciphertext c^*, $\mathcal{B}'$ forwards $\sigma'_i = (\sigma_{1,i}, c^*, \pi_i)$ to $\mathcal{B}$ as the response to the i-th O_{VESSig} query. Here, if $c^* = \mathsf{Enc}(APK, \sigma)$, the view of $\mathcal{B}$ is the same as in $\mathbf{Game}_2$. While if $c^* = \mathsf{Enc}(APK, 0)$, the view of $\mathcal{B}$ is the same as in $\mathbf{Game}_3$. Finally, if $\mathcal{B}$ outputs a valid signature σ^* on

m^* such that $m_i = m^*$, $\mathcal{B}'$ outputs $b = 0$. Otherwise, output $b = 1$. Let ZK denote the event breaking the SHVZK of proof π and IND denote the event breaking the IND-CPA of $\mathcal{E}$, we have

$$|\mathsf{Pr}_3[\mathsf{Succ}] - \mathsf{Pr}_2[\mathsf{Succ}]| \leq \mathsf{Pr}[\mathsf{ZK}] + \mathsf{Pr}[\mathsf{IND}]$$

Moreover, by the unforgeability of EdDSA-like signature, $\mathsf{Pr}_3[Succ]$ is also negligible.

To sum up, we have:

$$\begin{aligned}\mathsf{Pr}_1[\mathsf{Succ}] &\leq |\mathsf{Pr}_1[\mathsf{Succ}] - \mathsf{Pr}_2[\mathsf{Succ}]| + |\mathsf{Pr}_2[\mathsf{Succ}] \\ &\quad -\mathsf{Pr}_3[\mathsf{Succ}]| + |\mathsf{Pr}_3[\mathsf{Succ}]| \\ &\leq \mathsf{Pr}_1[\mathsf{Sound}] + \mathsf{Pr}[\mathsf{ZK}] + \mathsf{Pr}[\mathsf{IND}] \\ &\quad +\mathsf{Pr}_3[\mathsf{Succ}]\end{aligned}$$

Since that $\mathsf{Pr}[\mathsf{Sound}]$, $\mathsf{Pr}[\mathsf{ZK}]$, $\mathsf{Pr}[\mathsf{IND}]$, and $\mathsf{Pr}_3[\mathsf{Succ}]$ are negligible, we have $\mathsf{Pr}_1[\mathsf{Succ}]$ is negligible. As a result, $\mathsf{Pr}_0[\mathsf{Succ}] = q_S \cdot \mathsf{Pr}_1[\mathsf{Succ}]$ is negligible. The verifiably encrypted EdDSA-like signature scheme is secure against extraction.

6.5 Efficiency Analysis

Suppose that $|\sigma_2| = n$. Let $\mathbf{G}$ denote an element in $\mathbb{G}$ and $\mathbf{Z}$ denote an element in $\mathbb{Z}_q$. By $\mathbf{E}$ we mean an exponentiation operation. Table 6.1 summarizes the costs of the verifiably encrypted EdDSA-like signature scheme in terms of space complexity, time complexity, and estimated runtime based on the benchmark of $\mathbf{E}$. For time complexity, we only consider the most expensive operation, namely, exponentiation operation over $\mathbb{G}$. The space complexity and time complexity of the scheme are shown in Table 6.1.

Table 6.1: Efficiency of the Verifiably Encrypted EdDSA-like Signature Scheme

	Space Complexity	Time Complexity
Sig	$2\mathbf{Z}$	$1\mathbf{E}$
Ver	$1\mathbf{Z}$	$2\mathbf{E}$
VESSig	$(2log_2(n) + \frac{n}{16} + 4)\mathbf{G} + 7\mathbf{Z} + 1\mathbf{G}/\mathbf{Z}$	$(12n + \frac{3n}{32} + 12)\mathbf{E}$
VESVer	$1\mathbf{Z}$	$(\frac{65}{32}n + 2log_2(n) + 13)\mathbf{E}$
Adj	$1\mathbf{Z}$	$28\mathbf{E}$

6.6 Summary

This chapter introduces an efficient verifiably encrypted EdDSA-like signature scheme based on Σ-protocol and Bulletproofs for Pedersen commitment. For the case of ECDSA, the bandwidth consumption of our verifiably encrypted EdDSA-like signature scheme are 29 times more efficient compared with the verifiably encrypted ECDSA signature scheme in the previous chapter. The performance of the escrow protocol for Bitcoin resulting from our verifiably encrypted EdDSA-like signature scheme is analyzed in Chapter 7.

Chapter 7

Performance

Overall, an escrow protocol for cryptocurrencies can be divided into two phases, namely, the on-chain phase and the off-chain phase. Taking Bitcoin as an example, for the on-chain phase, a Bitcoin lock transaction and a Bitcoin unlock transaction are conducted. In the off-chain phase, Alice generates a verifiably encrypted ECDSA signature $\sigma'_A = (c, \pi, r)$ on transaction T and sends it to Bob. Bob verifies σ'_A. In the withdrawal phase, if Bob performs his duties but Alice refuses to publish her part of multi-signature on T, Bob forwards σ'_A to the adjudicator, who will construct the signature σ_A on T on behalf of Alice.

In this chapter, we evaluate the performance of our escrow protocol on Bitcoin. In particular, we focus on the performance of the on-chain phase in Section 7.1. In Section 7.2, we compare the off-chain performance of the escrow protocols based on our verifiably encrypted signature schemes presented in Chapter 5 and Chapter 6.

7.1 On-chain Phase

As mentioned, simply a lock transaction and an unlock transaction are conducted on the chain. The on-chain script of the escrow protocol for Bitcoin is shown in Table 7.1.

We evaluate the on-chain performance in Testnet, an alternative Bitcoin blockchain to be used for testing. The experiment results are shown in Table 7.2. As shown in the table,

Table 7.1: The Script of the Lock and Unlock Transactions of Our Escrow Protocol for Bitcoin

Bitcoin Lock Script	P2SH Script
OP_IF	OP_HASH160
OP_2	script hash
{{PK_A}}	OP_ EQUAL
{{PK_B}}	Bitcoin Unlock Script (App 1)
OP_2	
OP_ CHECKMULTISIG	OP_ 0
OP_ ELSE	{{signature_A}}
{{blocknum t}}	{{signature_B}}
OP_ CHECKLOCKTIMEVERIFY	OP_1
OP_ DROP	Bitcoin Unlock Script (App 2)
{{PK_A}}	
OP_ CHECKSIG	{{signature_A}}
OP_ENDIF	OP_ 0

Table 7.2: Performance of the On-Chain Phase

Lock Transaction (6 confirmation)			Unlock Transaction (6 confirmation)		
Time	Size	Fee	Time	Size	Fee
$\approx$ 70 min	$\approx$ 220 B	0.0001 BTC	$\approx$ 70 min	$\approx$ 400 B	0.0002 BTC

after the release of each transaction, the 1st confirmation reaches in around 8 minutes, and the 6th confirmation arrives in 70 minutes on average. Indeed, the script size of our protocol is 21% (i.e. 78 bytes) larger than a standard `P2SH Multisig` script. But still, the confirmation time is about the same as a standard Bitcoin transaction. In short, our escrow protocol for Bitcoin takes approximately 140 minutes to complete a single exchange. This is about two times to that of a standard transaction, and is the same as the baseline solution relying on the trusted platform in which Alice transfers her coins to the platform first, and the platform releases the fund to either Alice and Bob later. The concrete transactions as a result of our protocol are given in Figure 7.1 and Figure 7.2.

```
Lock Transaction:
Tx:  935033c7e588315f5a7a3deafc6984ed8 3630fdacc81a0b4345bee4
cf86cf0f5
Block:  541377    Size:  223 bytes
Speed:  immediate mine in next block
First confirmation:  6 min
Sixth confirmation:  66 min
```

Figure 7.1: An Escrow Lock Transaction

```
Unlock Transaction:
Unlock TX (Only use Alice signature after block 541380)
Tx:  dd15f268f4f67123d3c1a50756db0a5ee
857727407c1e9eacf6e7e30c0eb2f30
Block:  541381    Size:  372 bytes
Speed:  immediate mine in next block
First confirmation:  24 min
Sixth confirmation:  69 min
```

Figure 7.2: An Escrow Unlock Transaction

7.2 Off-chain Phase

The efficiency of the off-chain phase depends mainly on the VES. In particular, when no party or Bob deviates from the protocol, simply a VES signature is generated and verified. An additional adjudication is required if Alice deviates from the protocol.

In Chapter 5 and Chapter 6, we introduce two VES schemes for ECDSA. One is dedicated to ECDSA. The other applies all EdDSA-like signature schemes. We evaluate their performance on Linux DESKTOP-IDMG9KA 4.4.0-19041-Microsoft. The experiment results are shown in Table 7.3.

Escrow via verifiably encrypted ECDSA takes 4262 ms and 40.63 KB when no party/Bob deviates from the protocol, and 4505 ms and 57.16 KB when Alice deviates from the protocol.

Escrow via verifiably encrypted EdDSA-like, on the other hand, takes 265 ms and 1.5 KB when no party/Bob deviates from the protocol, and 306 ms and 2 KB when Alice deviates from the protocol.

By comparison, the bandwidth consumption of the escrow protocols resulting from verifiably encrypted ECDSA is around 27 times larger than that of the verifiably encrypted EdDSA-like. However, the latter requires the TTP to locally store a check-up table of size $O(2^{16})$.

Table 7.3: Performance of the Off-chain Phase

Situation	No party deviates from the protocol		Alice deviates from the protocol		Bob deviates from the protocol	
	Time	Size	Time	Size	Time	Size
Via Verifiably Encrypted ECDSA	≈ 4262 ms	40.63 KB	≈ 4505 ms	57.16 KB	≈ 4262 ms	40.63 KB
Via Verifiably Encrypted EdDSA-like	≈ 265 ms	1.5 KB	≈ 306 ms	2 KB	≈ 265 ms	1.5 KB

Chapter 8

Conclusion

8.1 Conclusion

In this thesis, we develop the framework to construct escrow protocol via VES. To realize our framework, we propose an efficient verifiably encrypted ECDSA scheme and prove its security under an enhanced security model. Furthermore, we develop the framework to categorize EdDSA-like signature, which includes a set of variants of ECDSA, e.g. EdDSA and Schnorr signature, and managed to develop a generic VES construction for all EdDSA-like signature schemes. The resulting verifiably encrypted EdDSA-like signature scheme yields an escrow protocol for all popular cryptocurrencies. Finally, we conduct thorough complexity analysis of our escrow protocols and evaluate the feasibility on Bitcoin mainnet.

8.2 Future Work

Replacing (Twisted) ElGamal Encryption Scheme. In our verifiably encrypted EdDSA-like signature scheme, σ_2 is encrypted by every 32-bit. In the case of ECDSA where $|\sigma_2|$=256, a total 8 pairs of ciphertexts and range proofs are needed. We are currently thinking to replace the ElGamal encryption scheme with the CL encryption scheme [13] to improve communication and computation efficiency. CL encryption scheme is a linearly homomorphic scheme whose message space is the whole set of $\mathbb{Z}_q$. Hence σ_2 can be encrypted with a single CL ciphertext and no additional range proof is required.

Payment Channel Network with Atomicity. Payment Channel Network (PCN) [37] is introduced to mitigate the scalability issue in cryptocurrencies. The main idea is to establish an off-chain linkage and enable multiple transactions through a single on-chain transaction. However, same as the standard cryptocurrency transaction, it lacks an atomicity guarantee (i.e. both or none of the payer and payee receive goods/digital coins). We are currently thinking to use verifiably encrypted EdDSA-like signature schemes or adaptor signature schemes [20] to enforce fairness in PCN.

Reference

[1] N. Asokan, Matthias Schunter, and Michael Waidner. “Optimistic fair exchange of digital signatures”. In: *IEEE Journal on Selected Areas in Communications* 18.4 (2000), pp. 593–610.

[2] N. Asokan, Matthias Schunter, and Michael Waidner. “Optimistic Fair Exchange of Digital Signatures (Extended Abstract)”. In: *Advances in Cryptology - EUROCRYPT ’98, International Conference on the Theory and Application of Cryptographic Techniques*. Vol. 1403. Lecture Notes in Computer Science. Springer, 1998, pp. 591–606.

[3] N. Asokan, Matthias Schunter, and Michael Waidner. “Optimistic Protocols for Fair Exchange”. In: *CCS ’97, Proceedings of the 4th ACM Conference on Computer and Communications Security*. ACM, 1997, pp. 7–17.

[4] Waclaw Banasik, Stefan Dziembowski, and Daniel Malinowski. “Efficient Zero-Knowledge Contingent Payments in Cryptocurrencies Without Scripts”. In: *Computer Security - ESORICS 2016 - 21st European Symposium on Research in Computer Security*. Vol. 9879. Lecture Notes in Computer Science. Springer, 2016, pp. 261–280.

[5] Feng Bao, Robert H. Deng, and Wenbo Mao. “Efficient and Practical Fair Exchange Protocols with Off-Line TTP”. In: *Security and Privacy - 1998 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 1998, pp. 77–85.

[6] Binance. *Binance*. Last accessed 14 Aug 2021. URL: `https://www.binance.com/en`.

[7] BitcoinWiki. *Zero Knowledge Contingent Payment*. Last accessed 28 May 2018. URL: `https://en.%20bitcoin.it/wiki`.

[8] Dan Boneh et al. “Aggregate and Verifiably Encrypted Signatures from Bilinear Maps”. In: *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques*. Vol. 2656. Lecture Notes in Computer Science. Springer, 2003, pp. 416–432.

[9] Sean Bowe. *Pay-to-sudoku*. Published 2016. URL: `https://github.com/zcash/pay-to-sudoku`.

[10] Benedikt Bünz et al. "Bulletproofs: Short Proofs for Confidential Transactions and More". In: *2018 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2018, pp. 315–334.

[11] Theresa Calderon et al. "Rethinking Verifiably Encrypted Signatures: A Gap in Functionality and Potential Solutions". In: *Topics in Cryptology - CT-RSA 2014 - The Cryptographer's Track at the RSA Conference 2014, San Francisco, CA, USA, February 25-28, 2014. Proceedings*. Ed. by Josh Benaloh. Vol. 8366. Lecture Notes in Computer Science. Springer, 2014, pp. 349–366.

[12] Matteo Campanelli et al. "Zero-Knowledge Contingent Payments Revisited: Attacks and Payments for Services". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 229–243.

[13] Guilhem Castagnos and Fabien Laguillaumie. "Linearly Homomorphic Encryption from DDH". In: *Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015*. Ed. by Kaisa Nyberg. Vol. 9048. Lecture Notes in Computer Science. Springer, 2015, pp. 487–505.

[14] Yu Chen et al. "PGC: Decentralized Confidential Payment System with Auditability". In: *Computer Security - ESORICS 2020 - 25th European Symposium on Research in Computer Security*. Vol. 10446. Lecture Notes in Computer Science. Springer, 2017, pp. 275–287.

[15] CoinMarketCap. *CoinMarketCap*. Last accessed 14 Aug 2019. URL: `https://coinmarketcap.com/`.

[16] Yevgeniy Dodis, Pil Joong Lee, and Dae Hyun Yum. "Optimistic Fair Exchange in a Multi-user Setting". In: *Public Key Cryptography - PKC 2007, 10th International Conference on Practice and Theory in Public-Key Cryptography*. Vol. 4450. Lecture Notes in Computer Science. Springer, 2007, pp. 118–133.

[17] Yevgeniy Dodis and Leonid Reyzin. "Breaking and repairing optimistic fair exchange from PODC 2003". In: *Proceedings of the 2003 ACM workshop on Digital rights management*. ACM, 2003, pp. 47–54.

[18] Vasily Dolmatov and Alexey Degtyarev. *GOST R 34.10-2012: Digital Signature Algorithm*. RFC 7091. Dec. 2013. DOI: `10.17487/RFC7091`.

[19] Electroneum. *Electroneum*. Last accessed 14 Aug 2019. URL: `https://electroneum.com/zh/`.

[20] Andreas Erwig et al. “Two-Party Adaptor Signatures from Identification Schemes”. In: *Public-Key Cryptography - PKC 2021 - 24th IACR International Conference on Practice and Theory of Public Key Cryptography*. Ed. by Juan A. Garay. Vol. 12710. Lecture Notes in Computer Science. Springer, 2021, pp. 451–480.

[21] Amos Fiat and Adi Shamir. “How to Prove Yourself: Practical Solutions to Identification and Signature Problems”. In: *Advances in Cryptology - CRYPTO ’86, Santa Barbara, California, USA, 1986, Proceedings*. Vol. 263. Lecture Notes in Computer Science. Springer, 1986, pp. 186–194.

[22] Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. “Threshold-Optimal DSA/ECDSA Signatures and an Application to Bitcoin Wallet Security”. In: *Applied Cryptography and Network Security - 14th International Conference, ACNS*. Vol. 9696. Lecture Notes in Computer Science. Springer, 2016, pp. 156–174.

[23] github. *Escrow protocol via Multisig*. Last accessed 14 Nov 2021. URL: `https://github.com/Macil/multisig-escrow`.

[24] Steven Goldfeder et al. “Escrow Protocols for Cryptocurrencies: How to Buy Physical Goods Using Bitcoin”. In: *Financial Cryptography and Data Security - 21st International Conference,FC*. Vol. 10322. Lecture Notes in Computer Science. Springer, 2017, pp. 321–339.

[25] Jens Groth and Markulf Kohlweiss. “One-Out-of-Many Proofs: Or How to Leak a Secret and Spend a Coin”. In: *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Vol. 9057. Lecture Notes in Computer Science. Springer, 2015, pp. 253–280.

[26] Christian Hanser, Max Rabkin, and Dominique Schröder. “Verifiably Encrypted Signatures: Security Revisited and a New Construction”. In: *Computer Security - ESORICS 2015 - 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part I*. Ed. by Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl. Vol. 9326. Lecture Notes in Computer Science. Springer, 2015, pp. 146–164.

[27] Ethan Heilman et al. “TumbleBit: An Untrusted Bitcoin-Compatible Anonymous Payment Hub”. In: *24th Annual Network and Distributed System Security Symposium, NDSS*. The Internet Society, 2017.

[28] https://commerce.coinbase.com/. *Coinbase Commerce*. Last accessed 24 Jul 2021. URL: `https://commerce.coinbase.com/`.

[29] Qiong Huang, Duncan S. Wong, and Willy Susilo. "Efficient Designated Confirmer Signature and DCS-Based Ambiguous Optimistic Fair Exchange". In: *IEEE Trans. Information Forensics and Security* 6.4 (2011), pp. 1233–1247.

[30] Qiong Huang et al. "Efficient Optimistic Fair Exchange Secure in the Multi-user Setting and Chosen-Key Model without Random Oracles". In: *Topics in Cryptology - CT-RSA 2008*. Vol. 4964. Lecture Notes in Computer Science. Springer, 2008, pp. 106–120.

[31] Xinyi Huang et al. "Preserving Transparency and Accountability in Optimistic Fair Exchange of Digital Signatures". In: *IEEE Trans. Information Forensics and Security* 6.2 (2011), pp. 498–512.

[32] Huobi. *Huobi Global*. Last accessed 14 Aug 2021. URL: `https://www.huobi.com/zh-cn/`.

[33] *Information Technology — Security Techniques — Cryptographic Techniques Based on Elliptic Curves — Part 2: Digital Signatures*. Standard. International Organization for Standardization/International Electrotechnical Commission, 2002.

[34] Simon Josefsson and Ilari Liusvaara. *Edwards-Curve Digital Signature Algorithm (EdDSA)*. RFC 8032. 2017. DOI: `10.17487/RFC8032`.

[35] Kee Sung Kim and Ik Rae Jeong. "Efficient verifiably encrypted signatures from lattices". In: *Int. J. Inf. Sec.* 13.4 (2014), pp. 305–314. URL: `https://doi.org/10.1007/s10207-014-0226-0`.

[36] Steve Lu et al. "Sequential Aggregate Signatures and Multisignatures Without Random Oracles". In: *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Vol. 4004. Lecture Notes in Computer Science. 2006, pp. 465–485.

[37] Giulio Malavolta et al. "Concurrency and Privacy with Payment-Channel Networks". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 455–471.

[38] Torben P. Pedersen. "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing". In: *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference*. Ed. by Joan Feigenbaum. Vol. 576. Lecture Notes in Computer Science. Springer, 1991, pp. 129–140.

[39] Ronald L. Rivest et al. "Responses to NIST's Proposal". In: *Commun. ACM* 35.7 (1992), pp. 41–54.

[40] Markus Rückert. “Verifiably Encrypted Signatures from RSA without NIZKs”. In: *Progress in Cryptology - INDOCRYPT 2009, 10th International Conference on Cryptology in India*. Vol. 5922. Lecture Notes in Computer Science. Springer, 2009, pp. 363–377.

[41] Markus Rückert, Michael Schneider, and Dominique Schröder. “Generic Constructions for Verifiably Encrypted Signatures without Random Oracles or NIZKs”. In: *Applied Cryptography and Network Security, 8th International Conference, ACNS 2010, Beijing, China, June 22-25, 2010. Proceedings*. Ed. by Jianying Zhou and Moti Yung. Vol. 6123. Lecture Notes in Computer Science. 2010, pp. 69–86.

[42] Claus-Peter Schnorr. “Efficient Signature Generation by Smart Cards”. In: *J. Cryptol.* 4.3 (1991), pp. 161–174.

[43] Adi Shamir. “How to Share a Secret”. In: *Commun. ACM* 22.11 (1979), pp. 612–613.

[44] Daniel Shanks. “Class number, a theory of factorization, and genera”. In: *Proc. of Symp. Math. Soc* 20 (1971), pp. 415–440.

[45] Sean Shen, Sean Shen, and XiaoDong Lee. *SM2 Digital Signature Algorithm*. Internet-Draft draft-shen-sm2-ecdsa-02. Work in Progress. Internet Engineering Task Force, 2014. 40 pp.

[46] Markus Stadler. “Publicly Verifiable Secret Sharing”. In: *Advances in Cryptology - EUROCRYPT ’96, International Conference on the Theory and Application of Cryptographic Techniques*. Ed. by Ueli M. Maurer. Vol. 1070. Lecture Notes in Computer Science. Springer, 1996, pp. 190–199.

[47] Fenghe Wang and Shaoquan Shi. “Lattice-Based Encrypted Verifiably Encryption Signature Scheme for the Fair and Private Electronic Commence”. In: *IEEE Access* 7 (2019), pp. 147481–147489.

[48] Yang Wang, Man Ho Au, and Willy Susilo. “Perfect Ambiguous Optimistic Fair Exchange”. In: *Information and Communications Security - 14th International Conference, ICICS*. Vol. 7618. Lecture Notes in Computer Science. Springer, 2012, pp. 142–153.

[49] Yang Wang, Man Ho Allen Au, and Willy Susilo. “Revisiting Optimistic Fair Exchange Based on Ring Signatures”. In: *IEEE Trans. Information Forensics and Security* 9.11 (2014), pp. 1883–1892.

[50] Yang Wang et al. “Collusion-Resistance in Optimistic Fair Exchange”. In: *IEEE Trans. Information Forensics and Security* 9.8 (2014), pp. 1227–1239.

[51] Bitcoin Wiki. *Bitcoin Timelock*. Last accessed 24 Jul 2019. URL: `https://en.bitcoin.it/wiki/Timelock`.

[52] Wikipedia. *Bitfinex*. Last accessed 4 Jul 2021. URL: `https://en.wikipedia.org/wiki/Bitfinex`.

[53] Wikipedia. *Blockchain*. Last accessed 14 Aug 2019. URL: `https://en.wikipedia.org/wiki/Blockchain`.

[54] Xiao Yang et al. "Practical Escrow Protocol for Bitcoin". In: *IEEE Trans. Inf. Forensics Secur.* 15 (2020), pp. 3023–3034.

[55] Zdnet. *Billions were stolen in blockchain hacks last year*. Last accessed 4 Jul 2021. URL: `https://www.zdnet.com/article/billions-were-stolen-in-blockchain-hacks-in-2020/`.

Appendix A

Security Proof of Σ-protocol π_1

We now prove that Σ- protocol π_1 is complete, sound, and SHVZK.

- **Completeness**. By $u_i = g_1^{s_i} g_2^{\beta_i}$, $v_i = T^{\beta_i}$, and $s_i \in \{0,1\}$, it is easy to verify that $A_{i,1} u_i^x = g_1^{\rho_{s_i}} g_2^{\rho_{\beta_i}} (g_1^{s_i} g_2^{\beta_i})^x = g_1^{z_{i,1}} g_2^{z_{i,2}}$, $A_{i,2} v_i^x = T^{\rho_{\beta_i}} T^{\beta_i x} = T^{z_{i,2}}$, $A_{i,3} u_i^{x - z_{i,1}} = g_1^{(1-s_i)s_i x} g_2^{z_{i,3}} = g_2^{z_{i,3}}$, $A_{i,4} v_i^{x - z_{i,1}} = T^{\beta_i (x - z_{i,1}) + \rho_i} = T^{z_{i,3}}$. Hence, Σ-protocol π_1 is complete.

- **Soundness**. Let $z_{i,1}, z_{i,2}, z_{i,3}$ be the response to challenge x, and $z'_{i,1}, z'_{i,2}, z'_{i,3}$ be the response to challenge x'. Combining the first and second verification equations of the transcripts, we have $(u_i^{x-x'}, v_i^{x-x'}) = (g_1^{z_{i,1} - z'_{i,1}} g_2^{z_{i,2} - z'_{i,2}}, T^{z_{i,2} - z'_{i,2}})$. Defining $s_i = \frac{z_{i,1} - z'_{i,1}}{x - x'}$ and $\beta_i = \frac{z_{i,2} - z'_{i,2}}{x - x'}$, we have $u_i = g_1^{s_i} g_2^{\beta_i}$, $v_i = T^{\beta_i}$. Combining the third and fourth verification equations of the transcripts, we have $(u_i^{x - x' - (z_{i,1} - z'_{i,1})}, v_i^{x - x' - (z_{i,1} - z'_{i,1})})$ $= (g_2^{z_{i,3} - z'_{i,3}}, T^{z_{i,3} - z'_{i,3}})$. Since $s_i(x - x') = z_{i,1} - z'_{i,1}$, $u_i = g_1^{s_i} g_2^{\beta_i}$, and $v_i = T^{\beta_i}$, we have $(u_i^{x - x' - (z_{i,1} - z'_{i,1})}, v_i^{x - x' - (z_{i,1} - z'_{i,1})}) = (u_i^{(1-s_i)(x-x')}, v_i^{(1-s_i)(x-x')}) = (g_1^{s_i(1-s_i)(x-x')}$ $g_2^{\beta_i(1-s_i)(x-x')}, T^{\beta_i(1-s_i)(x-x')})$. In other words, $g_1^{s_i(1-s_i)(x-x')} g_2^{\beta_i(1-s_i)(x-x')} = u_i^{x - x' - (z_{i,1} - z'_{i,1})}$ $= g_2^{z_{i,3} - z'_{i,3}}$. Equating the exponents (to base g_1), we have $s_i(1 - s_i)(x - x') = 0$, implying that $s_i \in \{0,1\}$. Therefore the witness s_i and β_i can be computed from two valid transcripts. Σ-protocol π_1 is sound.

- **SHVZK**. Given x, u_i, v_i, the simulator chooses $z_{i,1}, z_{i,2}, z_{i,3} \xleftarrow{R} \mathbb{Z}_q$ and computes

$A_{i,1} = u_i^{-x} g_1^{z_{i,1}} g_2^{z_{i,2}}$, $A_{i,2} = v_i^{-x} T^{z_{i,2}}$, $A_{i,3} = u_i^{z_{i,1}-x} g_2^{z_{i,3}}$, $A_{i,4} = v_i^{z_{i,1}-x} T^{z_{i,3}}$. We see that a polynomial-time simulator is capable of outputting valid transcripts without the witness. Therefore, Σ-protocol π_1 is SHVZK.

Appendix B

Security Proof of Σ-protocol π_2

We now prove that Σ-protocol π_2 is complete, sound, and SHVZK.

- **Completeness**. Suppose that $U = g_1^s g_2^\gamma$, we have $U^x g_1^{z_1} g_2^{z_2} = (g_1^s g_2^\gamma)^x g_1^{\rho_s - xs} g_2^{\rho_\gamma - x\gamma}$ = $g_1^{\rho_s} g_2^{\rho_\gamma} = Y_1$. Also, suppose that $g_1^h Q^r = R^s$, we have $(g_1^h Q^r)^x R^{z_1} = R^{xs} R^{\rho_s - xs} = R^{\rho_s} = Y_2$. Hence, Σ-protocol π_2 is complete.

- **Soundness**. Define z_1, z_2 as the response to challenge x, and z_1', z_2' as the response to challenge x'. Combining the verification equations of the transcripts, we have $U^{x'-x} = g_1^{z_1 - z_1'} g_2^{z_2 - z_2'}$ and $(g_1^h Q^r)^{x'-x} = R^{z_1 - z_1'}$. Defining $s = \frac{z_1 - z_1'}{x' - x}$ and $\gamma = \frac{z_2 - z_2'}{x' - x}$, we have $U = g_1^s g_2^\gamma$ and $g_1^h Q^r = R^s$. Therefore the witness s and γ can be computed from two valid proof transcripts. Σ-protocol π_2 is sound.

- **SHVZK**. Given U and $g_1^h Q^r$, the simulator chooses $z_1, z_2 \overset{R}{\leftarrow} \mathbb{Z}_p$ and computes $Y_1 = U^x g_1^{z_1} g_2^{z_2}$, $Y_2 = (g_1^h Q^r)^x R^{z_1}$. We see that a polynomial-time simulator can output valid transcripts without the witness. Therefore, Σ-protocol π_2 is SHVZK.