Towards Fair Multiparty Computation in Scriptless Distributed Ledger Systems

Minze Xu, Yuan Zhang and Sheng Zhong*
State Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing 210023, China
Email: cnmzxu@gmail.com, zhangyuan@nju.edu.cn,
zhongsheng@nju.edu.cn

Abstract—**Fairness is one of the fundamental properties for multiparty computation (MPC) protocols. Although fair MPC protocols for general functions is shown to be impossible with a dishonest majority, a variant of fairness called "fairness with penalty" has been explored recently. A MPC protocol provides fairness with penalty if either all participants can get the output, or the dishonest parties who break the protocol after getting the output will be financially penalized. Fairness with penalty is enabled by previous works leveraging the emerging distributed ledger systems (DLS), e.g. Bitcoin and Ethereum. They utilize the scripting functionality provided by the DLSs to make automatic penalty practical without relying on any trusted third party. However, there is also a significant number of DLSs that do not provide the scripting functionality.**

In this paper, we propose the ROSE protocol which enables fairness with penalty while only requiring the underlying DLS can verify and broadcast digital signatures on transactions. This requirement can be fulfilled by almost all DLSs, including the scriptless DLSs. To the best of our knowledge, it is still unknown how to realize fairness with penalty on scriptless DLSs before our work. We also provide a implementation of ROSE. The experimental results show that applying ROSE only brings little computation and communication overhead.

Index Terms—**Fair MPC protocols, Claim-or-refund Functionality, Distributed Ledger Systems**

I. Introduction

Fairness is one of the fundamental properties for multiparty computation (MPC) protocols. Using a MPC protocol, a group of participants jointly compute a function and generate an output. The protocol is said to be *fair*, if either all participants get the output, or none of them gets it. The property of fairness is crucial for many applications, such as electronic auctions and digital goods exchange [26], [15], [16].

Although fair MPC for general functions is shown to be impossible [21] in the standard model with a dishonest majority, a variant model called "fairness with penalty" has been actively explored recently [11], [41], [40], [42], [19], [37], [1], [18]. Intuitively, fairness with penalty ensures that either all participants get the output of the protocol, or the dishonest party who aborts the protocol after learning the output is financially penalized for its deviation. This model makes sense because a proper penalty can give the participants strong incentives to honestly complete the protocol.

Fairness (with penalty) is achieved leveraging the emerging distributed ledger systems (DLS), e.g. Bitcoin [48] and Ethereum [57]. A new paradigm, called "MPC with distributed ledgers,"[11], [37] is designed to make automatic penalty practical without relying on any trusted third-party. In this paradigm, all participants make deposits using digital currencies before running the MPC protocol. If a dishonest participant aborts the protocol prematurely, its deposit may be used to compensate the honest parties.

We notice that, to the best of our knowledge, all existing fair MPC protocols rely on the scripting functionality provided by Bitcoin and Ethereum [11], [41], [12], [40], [42], [19], [37]—in particular, they rely on smart contracts based on the scripting functionality. However, there is also a significant number of DLS [49], [47], [52], [58] that do not provide this functionality. Consequently, the following question naturally arises:

> *Is fair MPC possible in DLS that do not provide the scripting functionality?*

In this paper, we answer this question affirmatively by proposing the Race-Of-Solving-and-Exposing (*ROSE*) protocol which enables fairness while only requiring that users of the DLS can transfer digital currencies among accounts represented by public keys of a digital signature scheme. Currencies can be transferred from the account pk to another one with a transaction signed by its corresponding secret key. Besides, the signatures are broadcasted to all users of the DLS together with the transactions. In short, we only require that the DLS can verify and broadcast digital signatures on transactions.

This requirement can be fulfilled by virtually all DLSs, for that verifying digital signatures is the most standard way for a DLS to validate transactions. The transactions and their digital signatures are recorded on the ledger and broadcasted to all users.

ROSE brings advantages in the following aspects:

-Compatibility. The most essential advantage brought by *ROSE* is to make fair MPC protocols be compatible with the DLSs which do not provide the scripting functionality. We investigate the percentage of amount and the market share of the DLS providing scripting functionality among the top 40 DLSs with the highest market share. The result is shown in Table I. As can be noted, only about a quarter of DLSs provide scripting functionality. So that applying *ROSE* can greatly improve

TABLE I
SUPPORTS FOR SCRIPTING FUNCTIONALITY.

Top	Amount	Percentage	Market share
10	3	33.3%	29.7%
20	4	20.0%	27.8%
30	7	23.3%	27.9%
40	10	25.0%	28.0%

Besides, although some DLSs provide scripting functionality, complicate scripts are not widely accepted for security and efficiency concerns[7]. For instance, we collect all the transactions of blocks with heights in the range from 710,000 to 715,000 in Bitcoin, and find that all the scripts in these transactions are of less than six lines. The percentage of scripts for different usage is shown in Table. II. But existing realization of fair MPC in Bitcoin needs to deploy scripts with at least tens of lines[11], [41], [40].

TABLE II
PERCENTAGE OF SCRIPTS FOR DIFFERENT USAGE.

Usage	Percentage
Data record	37.18%
Signature Verification	23.47 %
Hash Verification	39.35 %

On the contrary, *ROSE* can enable fairness with penalty while only requiring the verification and broadcast of digital signatures on transactions. It is compatible with almost every DLSs.

-Efficiency. Applying *ROSE* to realize fair MPC protocols also reduces the computation burden on the underlying DLS. Because running complicate scripts in a DLS needs to consume much resource. For instance, the scripts in Bitcoin and Ethereum need to be re-executed by all miners, but re-execution of a complicate script costs much computation resource.

ROSE does not need to deploy complicate scripts and the underlying DLS only needs to verify digital signatures. This helps to save the computation resource of the underlying DLS. Although it is inevitably to move some computation tasks to the protocol participants, but we show that *ROSE* only brings little computation overhead to fair MPC participants.

-Privacy. Another exciting advantage brought by *ROSE* is the improvement of the privacy protection on the participants of fair MPC protocols.

To enable fairness with penalty, the participants of a MPC protocol are required to make deposit with transactions in DLSs at first. But the privacy protection offered by mainstream DLSs, such as Bitcoin and Ethereum, is very preliminary. The transaction amount in these DLSs are public and are observable to all users. And some de-anonymization techniques [13], [30] also arise to find the users behind the anonymous accounts. Thus, to make deposit using these systems may reveal the amount of deposit and the identities of participants of the MPC task to all external adversaries.

It should be noticed that the amount of deposit should be set close to the "value" of the MPC task in order to incentivize the fairness. For instance, if Alice and Bob hope to exchange some digital goods worth \$100,000, they will set the deposit close to this price. If the deposit is much lower, one party can also take great advantage by aborting the exchange after having the output even though it loses the deposit. Therefore, disclosing the amount of deposit to external adversaries may be unacceptable for privacy-concerning participants.

This "deposit privacy" issue can be addressed by making the deposit using the DLSs adopting privacy-enhancing technologies, e.g. Monero [53], [59], [43] and ZCash [52], [47]. Transactions on the distributed ledger of these confidential currencies are obfuscated so that the involving parties and transaction amounts cannot be deciphered by external adversaries. Thus, the amount of deposit and the identities of involving parties is not observable to external adversaries if the deposit is made using the confidential DLSs.

Unfortunately, typical confidential currencies, such as Monero and ZCash, do not provide the scripting functionality. And introducing this functionality to the confidential currencies is also a challenging task. There has been some trials[14], [39], [24], but they have not been widely applied. *ROSE* can help to solve this dilemma. It is compatible with confidential DLSs, thus it can be applied to enhance the privacy of fair MPC protocols. We provide an implementation of *ROSE* for Monero in Section V to prove this claim.

A. Solution Overview

In this paper, we propose *ROSE* which enables fairness with penalty without relying on the scripting functionality. In the following, we briefly overview our solution.

Claim-or-refund functionality. In [11], Bentov et al. define the claim-or-refund functionality $\mathcal{F}^{\star}_{CR}$ and a prominent line of works [11], [37], [40] proves that $\mathcal{F}^{\star}_{CR}$, along with standard cryptographic primitives, e.g. oblivious transfer (OT), is sufficient to design fair MPC protocols for general functions.

We illustrate the claim-or-refund functionality $\mathcal{F}^{\star}_{CR}$ in Figure. 1. It is a two-party functionality which accepts deposit from the "sender" and conditionally transfers the deposit to the "receiver".

$\mathcal{F}^{\star}_{CR}$ proceeds in three phases. In the deposit phase, the sender makes a deposit (e.g. a few Bitcoins) while specifying the condition with a circuit ϕ and the expiring time period τ. In the claim phase, the receiver can claim the deposit by sending a valid "witness" w which is accepted by the circuit (i.e. $\phi(w) = 1$) to $\mathcal{F}^{\star}_{CR}$. The witness will be revealed to the sender and the deposit will be transferred to the receiver by $\mathcal{F}^{\star}_{CR}$. Otherwise, the refund phase starts after the time period τ passing, where the deposit is refunded to the sender.

Assumptions and our results. In this paper, we propose *ROSE* to realize the claim-or-refund functionality while only requiring that users of the underlying DLS can transfer digital currencies among accounts represented by public keys of a

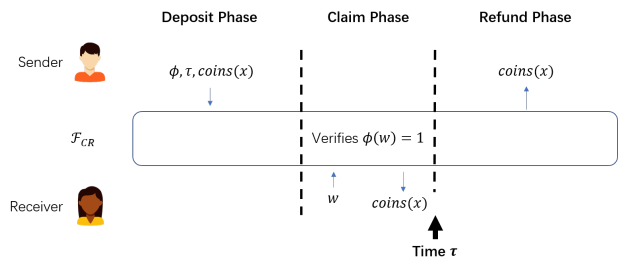


Fig. 1. The claim-or-refund functionality $\mathcal{F}^{\star}_{CR}$.

digital signature scheme. Currencies can be transferred from an account to another one given a transaction with a valid signature for the sender account. And the signature is broadcasted to all users by the DLS together with the transaction. In short, the DLS can verify and broadcast digital signatures on transactions.

This requirement is fulfilled by almost all DLS, for that verifying signatures is one of the most standard methods to validate transactions. And the signatures are recorded in the distributed ledger together with the transaction, so that all users can apparently see it.

Cryptographic primitives. At the heart of *ROSE* lies two cryptographic primitives, two-party adaptor signatures and verifiable time-lock puzzles.

Given a digital signature scheme, two-party adaptor signatures enable two users, who secretly share the corresponding secret key of a public key pk, to jointly generate a pre-signature $\tilde{\sigma}$ on a message m for an agreed circuit ϕ. Given a witness w which can be accepted by ϕ (i.e. $\phi(w) = 1$), the pre-signature $\tilde{\sigma}$ can be adapted into a valid signature σ for pk on m. Besides, the witness can be extracted from the knowledge of both σ and $\tilde{\sigma}$.

And a verifiable time-lock puzzle scheme allows one to lock a secret s into a puzzle Z, which cannot be solved before a scheduled time period τ passing. At the same time, the generator of the puzzle can produce a publicly verifiable proof π which proves the secret s meets some conditions. For instance, supposing two users secretly share the corresponding secret key of a public key for a digital signature scheme, then one user can lock its share in a puzzle and prove its honest behavior.

Roadmaps. Then we present the roadmaps of *ROSE* to realize the claim-or-refund functionality. We note that from a very basic level, the claim-or-refund functionality $\mathcal{F}^{\star}_{CR}$ provides the two sub-functionalities. It (1) establishes atomicity between the receiver claiming the deposit and sender knowing the witness, and (2) allows the sender to refund the deposit after a scheduled time period. We introduce how *ROSE* realizes the two sub-functionalities respectively in the following.

To ease understanding, we assme that the sender of $\mathcal{F}^{\star}_{CR}$ has stored some deposit in an account pk, and the corresponding secret key sk is secretly shared between the sender and the receiver. But we note that the deposit should be made after all preparations in the deposit phase being done, to avoid the receiver being able to lock the sender's deposit by aborting in the deposit phase.

To achieve sub-functionality (1), we have a key observation that the DLS provides an inherent atomicity between the validation of a transaction and the broadcast of the signature on it. The *ROSE* protocol builds a bridge between the inherent atomicity of the DLS and the atomicity in the claim phase leveraging two-party adaptor signatures. This idea is illustrated in Figure. 2.

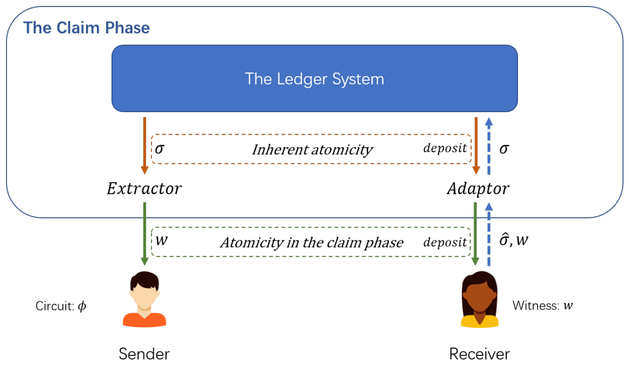


Fig. 2. Design for the claim phase of *ROSE*.

In the deposit phase, the sender and receiver jointly generate a pre-signature $\tilde{\sigma}$ on a transaction which transfers the deposit from the deposit account pk to the receiver for the desired circuit ϕ in $\mathcal{F}^{\star}_{CR}$. Then in the claim phase, if the receiver wants to claim the deposit, it adapts $\tilde{\sigma}$ into a valid signature σ with the witness w, and publishes the transaction with σ on the distributed ledger, then the deposit will be transferred to the receiver. Meanwhile, the signature is broadcasted to the sender, so the sender can extract the witness w with σ and $\tilde{\sigma}$.

To achieve sub-functionality (2) which allows the sender to refund the deposit after a scheduled time period τ, *ROSE* leverages another cryptographic primitive, the verifiable time-lock puzzles. In the deposit phase, the receiver locks its share of the secret key sk in a time-lock puzzle Z and produces a proof π, and sends them to the sender. Once the sender solving the puzzle, it can recover the full secret key with both shares and refund the deposit with it. This allows the sender to refund the deposit after a scheduled time period. The public verifiability of the proof ensures that the receiver cannot cheat, so that the sender can feel relieved to make the deposit.

Essentially, *ROSE* establishes a race between the sender solving the puzzle and the receiver exposing its witness, and the winner takes the deposit.

B. Our Contributions

The most essential contribution of our work is to first show that fairness with penalty can be realized without requiring any scripting functionality. Furthermore, concrete contributions of our work are summarized as follows.

Generalized construction. We realize the claim-or-refund functionality $\mathcal{F}^{\star}_{CR}$ with *ROSE*, which is compatible with almost all DLSs. A prominent line of works demonstrates wide applications of this functionality[11], [61], [41], [45], [40]. Our scriptless realization greatly extends the application scenarios of this functionality.

Novel cryptographic primitives. Our design for the *ROSE* protocol leverages the recent advances in adaptor signatures and time-release cryptography[4], [27], [54], [44]. We introduce and formulate the notion of verifiable time-lock puzzles. The primitives are of independent interest and we believe that they can be widely used in many other scenarios.

Efficient implementation. Besides the above theoretical contributions, we provide an efficient implementation of *ROSE* for Monero, and the experimental results show that applying *ROSE* only brings little computation and communication overhead to fair MPC protocols. We choose Monero for that it is a well-known confidential DLS, so that realizing fair MPC on it can enhance privacy protection on the deposit amount and the participants, but Monero provides no scripting functionality. This shows the advantage of *ROSE*.

C. Paper Organization

This rest of this paper is organized as follows. In Section II, we introduce backgrounds about the distributed ledger systems and the scripting functionality. Then we present the preliminaries for our work in Section III. The construction for *ROSE* is present in Section IV. We also provide theoretical analysis on the security of *ROSE* in this section. Our implementation of *ROSE* for Monero is shown in Section V. In Section VI, we discuss some related works. Finally, we conclude this paper in Section VII.

II. Backgrounds

A. Distributed Ledger Systems

Distributed ledger systems, like Bitcoin and Ethereum, allow a group of users to replicate a common ledger, which records all existing transactions among the users. And the consistency of ledgers across all users is ensured by a consensus protocol[46].

One basic function of DLS is to enable users to transfer digital currencies among accounts. Mostly, the accounts are represented by a public key for a digital signature scheme. And the digital currencies can be transferred from the sender account to the receiver account only with a valid signature for the public key which represents the sender account.

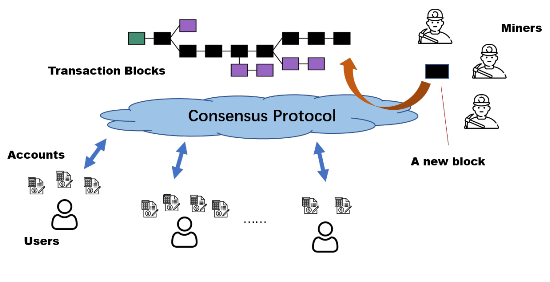


Fig. 3. Blockchain-based DLSs.

Currently, most DLSs are realized with the blockchain technology[62], [17]. The typical structure of blockchain-based DLSs is illustrated in Figure. 3. A group of users, namely the miners, determines which transactions should be included in the next block, i.e. be recorded in the public ledger. A transaction will be recorded to the public only if more than half of the miners agree, so the users can be assured that the transactions in the ledger are all valid and irreversible if they believe that majority of the miners are honest[36], [32], [31], [38].

B. Scripting Functionality

Besides the transference of digital currencies, some DLSs also provide scripting functionality, which enables users to control the transference of digital currencies automatically with some codes.

```
OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
```

Fig. 4. An example of a Bitcoin script. It declares that some bitcoins can be spent only with a transaction the signature on which can be verified by a public key whose hash equals to pubKeyHash.

The scripting functionality extends the power of DLSs, but the application of it in blockchain-based DLSs mainly brings two shortcomings. The first is the re-execution of scripts. As we state before, the validation of transactions in blockchain-based DLSs are verified by the miners, so while scripts are involved, the miners need to execute them to evaluate their results. Moreover, a script will be re-executed by at least half of the miners to verify the correctness of its result. So, complicate scripts will cost much more resource than simple ones for the re-execution.

Another main shortcoming is that there is currently no effective way to guarantee the security of scripts due to the very limited programming tools[63], [8]. But the irreversibility of the transactions on blockchain-based DLSs makes it difficult to fix bugs in the scripts after being deployed.

The shortcomings limit the use of complicate scripts in most DLSs. For instance, only some standard scripts, like hash verification and signature verification, are allowed in Bitcoin.

C. Confidential Distributed Ledger Systems

Primary blockchain-based DLSs, like Bitcoin and Ethereum, suffers inherent privacy issues for the publicity of the transactions. For instance, all the transactions are plainly recorded on the ledger of Bitcoin, so the flow of digital currencies from one account to another is observable to all users. There are many works pointing out that with some heuristic, although the users are hidden behind the anonymous accounts, the accounts can be deanonymized[30], [13]. Thus, the privacy protection provided by these DLS is very limited from current perspective.

To address above privacy issues, some confidential DLSs are devised, such as Monero[53], [59], [43] and Zcash[47], [52]. The transactions on these confidential DLSs are obfuscated, so that the outer users of a transaction cannot observe the participants and the amount of transferred coins of a transaction. For example, Monero leverages the ring signatures to enhance the privacy protection. In any transaction, the actual sender is hidden in a group of randomly chosen users. This design achieves k-anonymity for the real identity of the sender.

III. Preliminaries

We first introduce the notations we use in this paper. In the following of this paper, we use $[n]$ to denote the set of positive integers $\{1, 2, ..., n\}$. The notation $x \xleftarrow{\$} \mathcal{X}$ means the variable x is uniformly sampled from the set $\mathcal{X}$.

For any probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$, $\mathcal{A}(x; r)$ denotes the output of $\mathcal{A}$ running on the input x and randomness r and $\mathcal{A}(x)$ denotes the output of $\mathcal{A}$ running on x with uniform randomness r.

A. Adaptor Signatures

Adaptor signature is a novel primitive with important applications in DLSs. It is first formulated in [4], and then followed by many interesting and valuable works[28], [6], [27].

A digital signature scheme Π enables one to produce a publicly verifiable digital signature on a message. It is defined by specifying three PPT algorithms $(\mathsf{G}, \mathsf{S}, \mathsf{V})$:

- $(pk, sk) \leftarrow \mathsf{G}(1^\lambda)$: takes a security parameter as the input and outputs a pair of keys the public key pk and the secret key sk.
- $\sigma \leftarrow \mathsf{S}_{sk}(m)$: produces a signature σ on the message m.
- $b \leftarrow \mathsf{V}_{pk}(m, \sigma)$: verifies the signature. If the signature is valid, it outputs $b = 1$. Otherwise, it outputs $b = 0$.

By specifying a digital signature scheme Π and circuit classes $\mathcal{C}_\lambda$ for $\lambda \in \mathbb{N}$, an adaptor signature scheme $\mathsf{AS}_{\Pi,\mathcal{C}}$ provides four additional algorithms pS, pV, A and E:

- $\tilde{\sigma} \leftarrow \mathsf{pS}_{sk}(m, \phi)$: The pre-signing algorithm pS takes a secret key sk, a message $m \in \mathcal{M}_\lambda$ and a circuit $\phi \in \mathcal{C}_\lambda$ as the inputs, and outputs the pre-signature $\tilde{\sigma}$.
- $b \leftarrow \mathsf{pV}_{pk}(m, \tilde{\sigma}, \phi)$: The pre-verification algorithm pV takes a public key pk, a message m, a pre-signature $\tilde{\sigma}$ and a circuit ϕ as the inputs, then outputs the result $b \in \{0, 1\}$.
- $\sigma \leftarrow \mathsf{A}(\tilde{\sigma}, w)$: Given a pre-signature $\tilde{\sigma}$ and a witness w, the adaptor algorithm A adapts $\tilde{\sigma}$ into a signature σ.
- $w/ \perp \leftarrow \mathsf{E}(\sigma, \tilde{\sigma}, \phi)$: The extractor algorithm E extracts the witness w from a signature σ and a pre-signature $\tilde{\sigma}$. If the extraction fails, it outputs $\perp$.

An adaptor signature scheme enables the secret key owner to produce a pre-signature $\tilde{\sigma}$ on a message m for a circuit $\phi \in \mathcal{C}_\lambda$ using the pre-signing algorithm pS. The pre-signature $\tilde{\sigma}$ can be adapted into a signature σ by invoking the adaptor algorithm A. If w is accepted by the circuit (i.e. $\phi(w) = 1$), σ should be a valid signature on m. And the witness w can be extracted given $\tilde{\sigma}$ and σ with the extractor algorithm E.

We note that we only consider nontrivial circuit classes in adaptor signatures, which means that given a circuit $\phi \in \mathcal{C}_\lambda$, any PPT adversary $\mathcal{A}$ cannot find a witness w accepted by it with overwhelming probability.

The security of an adaptor signature scheme $\mathsf{AS}_{\Pi,\mathcal{C}}$ is defined from three aspects.

The *adaptive existential unforgeability* under *chosen message attack* for adaptor signatures (aEUF-CMA) requires that given a circuit $\phi \in \mathcal{C}$ and the oracle to query $\mathsf{S}_{sk}(m)$ and $\mathsf{pS}_{sk}(m, \phi)$ on chosen messages m, any PPT adversary $\mathcal{A}$ cannot forge a valid signature σ^* for a new message m^* even if it is given $\mathsf{pS}_{sk}(m^*, \phi)$

The *adaptability* requires that if $\mathsf{pV}_{pk}(m, \tilde{\sigma}, \phi) = 1$ for a given pre-signature $\tilde{\sigma}$, it can be adapted into a signature σ with a correct witness w.

The *extractability* requires that any PPT adversary cannot produce a valid signature σ from an adaptor signature $\tilde{\sigma} \leftarrow \mathsf{pS}_{sk}(m, \phi)$ while $\phi(\mathsf{E}(\sigma, \tilde{\sigma}, \phi)) = 0$.

B. Homomorphic Time-Lock Puzzles

The time-lock puzzles (TLP) allow one to encrypt a message for the future. Specifying a hardness parameter τ, time-lock puzzles enable one to lock a secret s into a puzzle Z which can be solved to recover s only after τ time of sequential computation.

Malavolta et al. propose a homomorphic time-lock puzzle in [44] based on the assumption of sequential squaring modulo a strong RSA integer. A time-lock puzzle scheme TLP is defined by specifying a secret space $\mathcal{S}$ and three algorithms defined as follows:

- $pp \leftarrow \mathsf{P}(1^\lambda, \tau)$: is a PPT algorithm which takes the security parameter λ and a hardness parameter τ as inputs, and outputs public parameters pp.
- $Z \leftarrow \mathsf{L}(pp, s)$: is a PPT algorithm that takes the secret s as the input and outputs a puzzle Z.
- $s := \mathsf{U}(pp, Z)$: is a deterministic algorithm that solves the puzzle Z to recover the secret s.

The correctness of a time-lock puzzle scheme essentially requires that an secret s can be correctly recovered by the solver algorithm U after τ times of sequential computations. The security requires that any adversary of polynomial parallelism cannot solve the puzzle before τ times of sequential computations.

Additionally, the linear homomorphism of a time-lock puzzle scheme enables one to evaluate a new puzzle for the linear combination of the secrets locked in the puzzles without solving the puzzles. Supposing the secret space $\mathcal{S}$ and randomness space $\mathcal{R}$ of a time lock scheme form two groups, then linear homomorphism provides a plus operation on puzzles and requires that

$$\begin{aligned} &\forall r_1, r_2 \in \mathcal{R}, \forall s_1, s_2 \in S, \forall pp \leftarrow \mathsf{P}(1^\lambda), \\ &Z_1 := \mathsf{L}(pp, s_1; r_1), Z_2 := \mathsf{L}(pp, s_2; r_2) \\ \Rightarrow &Z_1 + Z_2 = \mathsf{L}(pp, s_1 + s_2; r_1 + r_2). \end{aligned}$$

The implementation for a linearly homomorphic time-lock puzzle scheme is given by G. Malavolta et al. in [44].

C. Ideal Functionalities

To complete our work, we provide the formal description for the claim-or-refund functionality $\mathcal{F}^{\star}_{CR}$[11] in Figure. 5.

The claim-or-refund functionality $\mathcal{F}^{\star}_{CR}$ for the circuit class $\mathcal{C}$ running with parties P_s and P_r and the security parameter 1^{λ} proceeds as follows:

1) *Deposit Phase.* Upon receiving a deposit message $(\mathsf{deposit}, sid, s, r, \phi, \tau, x)$ and x coins from P_s, it records the message $(\mathsf{deposit}, sid, s, r, \phi, \tau, x)$ and sends it to both parties. Ignore any future deposit messages with the same sid from P_s to P_r.
2) *Claim Phase.* Upon receiving a claim message $(\mathsf{claim}, sid, s, r, \phi, \tau, x, w)$, it checks (1) if a deposit message $(\mathsf{deposit}, sid, s, r, \phi, \tau, x)$ was recorded and (2) if $\phi(w) = 1$. If so, it sends $(\mathsf{claim}, sid, s, r, \phi, \tau, x, w)$ and x coins to P_r, sends the witness w to P_s and deletes the message $(\mathsf{deposit}, sid, s, r, \phi, \tau, x)$.
3) *Refund Phase.* Upon time period τ passing, if the record $(\mathsf{deposit}, sid, s, r, \phi, \tau, x)$ was not deleted, it sends $(\mathsf{refund}, sid, s, r, \phi, \tau, x)$ and x coins to P_s, and deletes the message $(\mathsf{deposit}, sid, s, r, \phi, \tau, x)$.

Fig. 5. The Ideal Claim-or-Refund Functionality $\mathcal{F}^{\star}_{CR}$[11].

$\mathcal{F}^{\star}_{CR}$ interacts with two parties, the sender P_s and the receiver P_r, and proceeds in three phases. In the deposit phase, the sender specifies a circuit ϕ, a time period τ and the amount of deposit x. Then it comes to the claim phase. The receiver can claim the deposit by presenting a witness w which is accepted by ϕ. Otherwise, it is the refund phase while the time period τ passing. where the coins will be refunded to the receiver R.

As stated before, we only require that users of DLS can transfer digital currencies among accounts. The accounts are represented by public keys of a digital signature scheme Π, and currencies can be transferred from the sender account pk_S to the receiver account pk_R only if the transaction is signed by the corresponding secret key of pk_S. To formalize and abstract the exact property we require from the underlying DLS, we define the "ledger functionality" $\mathcal{F}_{\mathcal{L}}$ in Figure. 6. It enables one user, namely the sender, to pay some currencies (i.e. x coins) to an account pk and any one can take the coins by presenting a transaction with a signature for pk.

IV. The *ROSE* Protocol

In this section, we propose our construction for the *ROSE* protocol, which securely realizes the claim-or-refund functionality $\mathcal{F}^{\star}_{CR}$ on a DLS.

A. Building Blocks

The *ROSE* protocol mainly leverages two cryptographic tools, the verifiable time-lock puzzles and two-party adaptor signatures. We first introduce the two notions in this section.

1) Verifiable Time-lock Puzzles: A VTLP scheme enables the generator of a time-lock puzzle to prove that the secret locked in the puzzle meets some requirement, which is specified by a circuit. A VTLP scheme is defined by specifying

Given a digital signature scheme Π, $\mathcal{F}^{\star}_{\mathcal{L}}$ running with parties P_s, P_r, and the security parameter 1^{λ} proceeds as follows:

- Upon receiving a pay message (pay, s, pk, x) and x coins from P_s, it sends a message (pay, pk, x) to both parties and record a message (pk, x).
- Upon receiving a take message $(\mathsf{take}, b, pk, x, \sigma)$ from any party P_b, it verifies
(1) if (pk, x) is still recorded, and
(2) if $\Pi.\mathsf{V}_{pk}(m, \sigma) = 1$ where $m = (\mathsf{take}, r, pk, x)$.
If so, it transfers x coins to the party P_b, broadcasts the message $(\mathsf{take}, b, pk, x, \sigma)$ to both parties and deletes the record (pk, x).

Fig. 6. The Ledger Functionality $\mathcal{F}^{\star}_{\mathcal{L}}$

a secret space family $\{\mathcal{S}_{\lambda}\}_{\lambda \in \mathbb{N}}$, circuit classes $\mathcal{C}_{\lambda}$ and four algorithms:

- $pp \leftarrow \mathsf{P}(1^{\lambda}, \tau)$: is a PPT algorithm which takes the security parameter λ and a hardness parameter τ as inputs, and outputs public parameters pp.
- $(Z, \pi) \leftarrow \mathsf{L}(pp, \phi, s)$: is a PPT algorithm that takes a circuit $\phi \in \mathcal{C}$ and a secret $s \in \mathcal{S}_{\lambda}$ with $\phi(s) = 1$ as the inputs, then outputs a puzzle Z and a proof π.
- $b \leftarrow \mathsf{V}(pp, Z, \phi, \pi)$: is a PPT algorithm that takes a puzzle Z, a circuit $\phi \in \mathcal{C}$ and a proof π as the inputs, and outputs a bit $b \in \{0, 1\}$. If this algorithm accepts the proof, it outputs $b = 1$, and output $b = 0$ otherwise.
- $s := \mathsf{U}(pp, Z)$: is a deterministic algorithm that solves the puzzle Z to recover the secret s.

In addition to the puzzle, the generator algorithm L also outputs a proof π. An verifier algorithm V is also provided to verify the proof.

The correctness of a VTLP scheme is defined in Definition 1, which essentially tells that if the puzzle Z and the proof π which is correctly generated by the L given a circuit $\phi \in \mathcal{C}$ and a secret $s \in \mathcal{S}_{\lambda}$ with $\phi(s) = 1$, then (1) the puzzle can be solved by the algorithm U to obtain the secret s in a given time period; (2) the algorithm V accepts the corresponding proof π.

Definition 1 (Correctness)**.** *A VTLP scheme is correct if $\forall \tau \in \mathbb{N}, \forall \phi \in \mathcal{C}$, and $\forall s \in \mathcal{S}$ with $\phi(s) = 1$, the following conditions are satisfied:*

- *The probability ensemble indexed by $\lambda \in \mathbb{N}$*
$$Pr\left[\begin{array}{c|c} \mathsf{V}(pp, Z, Y, \pi) = 1 & pp \leftarrow \mathsf{P}(1^{\lambda}, \tau) \\ \wedge \mathsf{U}(pp, Z) = s & (Z, \pi) \leftarrow \mathsf{L}(pp, s, \phi) \end{array}\right]$$
is overwhelming.
- *For the above Z, the running time of $\mathsf{U}(pp, Z)$ are bounded by $p(\lambda, \tau)$ for a fixed polynomial p.*

The security of a VTLP scheme mainly involves two aspects. The indistinguishability essentially tells that for any algorithm with polynomial parallelism cannot distinguish two puzzles containing different secrets before τ times of sequential computation, where τ is the hardness parameter.

Definition 2 (Indistinguishability)**.** *A VTLP scheme meets the indistinguishability with gap $0 < \epsilon < 1$ if $\exists \widetilde{\mathcal{T}} \in \mathbb{N}^{\infty}[X]$, such that $\forall \mathcal{T} \in \mathbb{N}^{\infty}[X]$ with $\mathcal{T} \geq \widetilde{\mathcal{T}}$ and every polynomial-size adversary family $(\mathcal{A}_1, \mathcal{A}_2) = \{(\mathcal{A}_1, \mathcal{A}_2)_\lambda\}_{\lambda\in\mathbb{N}}$ where the depth of $\mathcal{A}_2$ is bounded from above by $\mathcal{T}^{\epsilon}(\lambda)$, the probability ensemble indexed by λ*

$$Pr\left[b' = b \;\middle|\; \begin{array}{c} pp \leftarrow \mathsf{P}(1^\lambda, \mathcal{T}(\lambda)) \\ (z, \phi, s_0, s_1) \leftarrow \mathcal{A}_1(1^\lambda, pp) \\ b \xleftarrow{\$} \{0,1\} \\ (Z, \pi) \leftarrow \mathsf{L}(pp, \phi, s_b) \\ b' \leftarrow \mathcal{A}_2(1^\lambda, pp, z, Z, \pi) \end{array} \right] \approx \frac{1}{2}$$

and $(s_0, s_1) \in \mathcal{S}_\lambda^2$ with $\phi(s_1) = \phi(s_2) = 1$.

The verifiability of a VTLP scheme is presented in Definition 3, which requires that PPT adversary $\mathcal{A}$ cannot produce a puzzle Z which can pass the verification while locking a incorrect puzzle.

Definition 3 (Verifiability)**.** *A VTLP scheme is verifiable if $\forall \tau \in \mathbb{N}$ for any PPT adversary $\mathcal{A}$, the probability ensemble*

$$Pr\left[\begin{array}{c} \mathsf{V}(pp, Z, \phi, \pi) = 1 \\ \wedge\phi(\mathsf{U}(pp, Z)) = 0 \end{array} \;\middle|\; \begin{array}{c} pp \leftarrow \mathsf{P}(1^\lambda, \tau) \\ (\phi, Z, \pi) \leftarrow \mathcal{A}(1^\lambda, pp) \end{array} \right]$$

which is indexed by λ is negligible.

Definition 4 (Security)**.** *An VTLP scheme* VP *is secure if it meets the indistinguishability in Definition 2 and the verifiability in Definition 3.*

2) Two-party Adaptor Signatures: Given an adaptor signature scheme $\mathsf{AS}_{\Pi,\mathcal{C}}$, two-party adaptor signatures (TPAS) introduce a two-party protocol $\Theta^{\mathsf{AS}_{\Pi,\mathcal{C}}.\mathsf{pS}}$ which *securely* realizes the twp-party pre-signature functionality $\mathcal{F}_{\mathsf{2pS}}$ in Figure. 7.

For a specific adaptor signature scheme $\mathsf{AS}_{\Pi,\mathcal{C}}.\mathsf{pS}$, two-party pre-signature functionality $\mathcal{F}_{\mathsf{2pS}}$ running with parties P_0, P_1, the security parameter 1^λ proceeds as follows:

1) **Setup**: It generates a secret key for Π by invoking $sk \leftarrow \Pi.\mathsf{G}(1^\lambda)$, and shares sk as sk_0 and sk_1 with a secret sharing algorithm. Then it sends sk_i to P_i for $i = 0, 1$ respectively.
2) **Pre-signing**: Upon receiving (sk_0, m_0, ϕ_0) from P_0 and (sk_1, m_1, ϕ_1) from party P_1, it sends $reject$ to both parties if $m_0 \neq m_1$ or $\phi_0 \neq \phi_1$. Otherwise, it recovers sk from sk_0 and sk_1, and evaluates $\tilde{\sigma} \leftarrow \mathsf{AS}_{\Pi,\mathcal{C}}.\mathsf{pS}_{sk}(m, \phi)$. Then it sends $\tilde{\sigma}$ to both P_0 and P_1.

Fig. 7. The two-party pre-signature functionality $\mathcal{F}_{\mathsf{2pS}}$.

B. The ROSE *Protocol*

Let Π be the digital signature scheme used by the ledger functionality $\mathcal{F}_{\mathcal{L}}$. To realize $\mathcal{F}_{CR}^{\star}$ for the circuit class $\mathcal{C}$, *ROSE* requires the above two cryptographic tools for specific circuit classes:

- A VTLP scheme VP for the circuit class $\mathcal{E} = \{\varepsilon_{\lambda,pk}\}$, where $\varepsilon_{\lambda,pk}(sk) = 1$ if and only if is a pair of valid public/secret keys generated by $\Pi.\mathsf{G}(1^\lambda)$. In the following, the circuit $\varepsilon_{\lambda,pk}$ is represented by pk for short.
- A TPAS scheme AS for the digital signature scheme Π and the circuit class $\mathcal{C}$.

The *ROSE* protocol is shown in Figure. 8. Similarly to the claim-or-refund functionality, *ROSE* also proceeds in three phases, the deposit phase, the claim phase and the refund phase. We suppose that the input for the sender is the required circuit ϕ that sepcifies the requirement of the witness for which the sender aims to exchange with the deposit, and the input for the receiver is the corresponding witness w, which meets $\phi(w) = 1$.

In the deposit phase, the sender and the receiver first jointly generate a key pair (pk, sk) and secretly shares the secret key sk. We note that this step is completed reversely in our description by first letting the sender and receiver respectively generate two key pairs (pk_S, sk_S) and (pk_R, sk_R). Then they aggregate the public keys pk_S and pk_R to produce a new public key pk and their secret keys serve as shares of the corresponding secret key sk for pk. Sharing the secret key in this way can reduce the communication cost in this step, and this property can be provided by almost all digital signature schemes.

After this step, the sender locks its share sk_R of the secret with the VTLP scheme and send the puzzle Z together with a proof π, which proves its honest behavior.

Then, the two parties invoke the $\Theta^{\mathsf{AS}_{\Pi,\mathcal{C}}}$ protocol to produce a pre-signature $\tilde{\sigma}$ on a transaction which transfers the deposit from the account pk to the receiver for the specific circuit ϕ. After all the preparations being done, the sender transfers the deposit to the account pk.

In the claim phase, the receiver can adapt the pre-signature $\tilde{\sigma}$ into a valid signature σ with the witness w if it wants to claim the deposit. Then, the deposit will be transferred to the receiver and the signature will be broadcasted to the sender by the DLS. So it can extract the witness w from $\tilde{\sigma}$ and σ.

Otherwise, if the receiver do not claim the deposit until the sender solves the puzzle Z, the sender can recover the full secret key with the two shares and takes the deposit back with the full secret key.

C. Theoretical Analysis

Theorem 1. *Let Π be the digital signature scheme used by the ledger functionality $\mathcal{F}_{\mathcal{L}}^{\star}$. Assuming the correctness and security of the VTLP scheme* VP*, and the adaptor signature scheme $\mathsf{AS}_{\Pi,\mathcal{R}}$ and the $\Theta^{\mathsf{AS}_{\Pi,\mathcal{C}}.\mathsf{pS}}$ protocol, the* ROSE *protocol* securely *realizes the claim-or-refund functionality $\mathcal{F}_{CR}^{\star}$ in the $\mathcal{F}_{\mathcal{L}}$-hybrid model.*

Proof Sketch. To prove this theorem, we first model the invocation of the two cryptographic primitives as two special ideal functionalities and rewrite *ROSE* in the hybrid model

The ROSE Protocol

	$Sender(\phi)$		$Receiver(w)$
	Deposit Phase		
1 :	$(pk_S, sk_S) \leftarrow \Pi.\mathsf{G}(1^\lambda)$	$\xleftrightarrow{pk \leftarrow \Pi.\mathsf{KAg}(pk_S, pk_R)}$	$(pk_R, sk_R) \leftarrow \Pi.\mathsf{G}(1^\lambda)$
2 :	Abort **if** $\mathsf{VP.V}(pp, Z, pk_R, \pi) \neq 1$.	$\xleftarrow{Z, \pi}$	$(Z, \pi) \leftarrow \mathsf{VP.G}(pp, pk_R, sk_R)$
3 :	Abort **if** $\mathsf{AS}_{\Pi,\mathcal{R}}.\mathsf{pV}_{pk}(\mathsf{Tx}, \tilde{\sigma}, \phi) \neq 1$.	$\xleftrightarrow[\tilde{\sigma} \leftarrow \Theta^{\mathsf{AS}_{\Pi,\mathcal{C}}.\mathsf{pS}}_{\langle sk_S, sk_R \rangle}(\mathsf{Tx}, \phi)]{\mathsf{Tx}=(\mathsf{take}, Receiver, pk, x)}$	Abort **if** $\mathsf{AS}_{\Pi,\mathcal{R}}.\mathsf{pV}_{pk}(\mathsf{Tx}, \tilde{\sigma}, \phi) \neq 1$.
4 :	Send $(\mathsf{pay}, s, pk, \mathsf{coins}(x))$ to $\mathcal{F}_\mathcal{L}$.		
	Claim Phase		
5 :	Upon receiving $(\mathsf{take}, pk, \sigma)$ from $\mathcal{F}^\star_\mathcal{L}$, extract $w := \mathsf{AS}_{\Pi,\mathcal{R}}.\mathsf{E}(\sigma, \tilde{\sigma}, \phi)$.		$\sigma \leftarrow \mathsf{AS}_{\Pi,\mathcal{R}}.\mathsf{A}(\tilde{\sigma}, w)$ Send $(\mathsf{take}, Receiver, pk, x, \sigma)$ to $\mathcal{F}^\star_\mathcal{L}$.
	Refund Phase		
6 :	$sk_R := \mathsf{VP.S}(pp, Z)$ $\mathsf{Tx}' = (\mathsf{take}, Sender, pk, x)$ $sk \leftarrow \Pi.\mathsf{SKA}(sk_S, sk_R)$ $\sigma' \leftarrow \Pi.\mathsf{S}_{sk}(\mathsf{Tx}')$ Send $(\mathsf{take}, Sender, pk, x, \sigma')$ to $\mathcal{F}^\star_\mathcal{L}$.		

Fig. 8. The *ROSE* protocol.

with acccess to the two ideal functionalities and the ledger functionality $\mathcal{F}_\mathcal{L}$. Then we construct an ideal simulator $\mathcal{S}$ in the ideal model with access to the claim-or-refund functionality $\mathcal{F}^\star_{CR}$, which simulates the behavior of the adversary $\mathcal{A}$ in the hybrid model interacting with *ROSE*, and prove that any PPT environment $\mathcal{Z}$ cannot distinguish whether it is interacting with the ideal simulator in the ideal model or it is interacting with the adversary in the hybrid model.

We model **verifiable time-lock puzzles** as a two-party functionality $\mathcal{F}_{\mathsf{VP}}$. It is parameterized with the security parameter 1^λ and the hardness paramter τ. Uponing receiving a public key pk_R from the sender of *ROSE* and a secret key sk_R from the receiver, it sends $reject$ to the sender of *ROSE* if sk_R is not the corresponding secret key of pk_R. Otherwise, it sends $accept$ to the sender immediately and sends sk_R to the sender after the time period τ passing.

From the security and correctness requirements for the VTLP scheme, we claim that the invocation of VP in *ROSE* (i.e. the lines 2 and 6 in Figure. 8) *securely* realizes $\mathcal{F}_{\mathsf{VP}}$.

We note that the invocation of the $\Theta^{\mathsf{AS}_{\Pi,\mathcal{C}}.\mathsf{pS}}$ protocol in *ROSE* (i.e. the line 3 in Figure. 8) is modeled by the two-party pre-signature functionality $\mathcal{F}_{\mathsf{2pS}}$ in Figure. 7. And by definition we can claim that the invocation of $\Theta^{\mathsf{AS}_{\Pi,\mathcal{C}}.\mathsf{pS}}$ *securely* realizes $\mathcal{F}_{\mathsf{2pS}}$.

Now, we can rewrite *ROSE* in the $(\mathcal{F}_{\mathsf{VP}}, \mathcal{F}_{\mathsf{AS}}, \mathcal{F}_\mathcal{L})$-hybrid model.

In the deposit phase, (1) the sender and the receiver first respectively generate (pk_S, sk_S) and (pk_R, sk_R) by invoking $\Pi.\mathsf{G}(1^\lambda)$, and exchange their public keys and aggregate them to obtain pk. Then (2) the sender sends pk_R to $\mathcal{F}_{\mathsf{VP}}$ and the receiver sends sk_R to $\mathcal{F}_{\mathsf{VP}}$. Now the sender will receive $reject$ or $accept$ from $\mathcal{F}_{\mathsf{VP}}$. If it receives $reject$, it aborts. Otherwise, it proceeds to the next step.

In the next step, (3) the sender sends $(sk_S, \mathsf{Tx}, \phi)$ to $\mathcal{F}_{\mathsf{2pS}}$ and the receiver sends $(sk_R, \mathsf{Tx}, \phi)$ to $\mathcal{F}_{\mathsf{2pS}}$, where Tx is a transaction which transfer the deposit from the account pk to the receiver. Either of them abors if it receives $reject$ from $\mathcal{F}_{\mathsf{2pS}}$. If neither of the parties aborts until now, the sender makes the deposit by sending the pay message to $\mathcal{F}_\mathcal{L}$.

In the claim phase, (4) the two parties proceeds as the claim phase in Figure. 8.

In the refund phase, (5) upon receiving sk_R from $\mathcal{F}_{\mathsf{VP}}$, the sender recovers the full secret key and takes the deposit back by sending a take message to $\mathcal{F}_\mathcal{L}$ and signing it with the full secret key.

Then we describes the ideal simulator $\mathcal{S}$ in the ideal model with access to the claim-or-refund functionality, which simulates the behavior of the adversary $\mathcal{A}$ interacting with the above *ROSE* in the $(\mathcal{F}_{\mathsf{VP}}, \mathcal{F}_{\mathsf{AS}}, \mathcal{F}_{\mathcal{L}})$-hybrid model. We consider two cases, where the adversary $\mathcal{A}$ corrupts the sender or the adversary $\mathcal{A}$ corrupts the receiver.

$\mathcal{A}$ **corrupts the sender.** If $\mathcal{S}$ receives any input from the environment $\mathcal{Z}$, it passes the input to the adversary. Then $\mathcal{S}$ generates a pair of keys (pk_R, sk_R) and exchanges pk_R with the adversary to obtain pk_S from $\mathcal{A}$ (step 1). Then $\mathcal{S}$ invokes $\mathsf{VP.G}(pp, pk_R, sk_R)$ to generate the puzzle Z and the proof π, and sends (Z, π) to the adversary (step 2).

If adversary does not abort and outputs a message $(sk_S, \mathsf{Tx}, \phi)$ in the next step, $\mathcal{S}$ recovers sk from sk_S and sk_R, and then invokes $\mathsf{AS}_{\Pi,\mathcal{C}}.\mathsf{pS}_{sk}(\mathsf{Tx}, \phi)$ to obtain the pre-signature $\tilde{\sigma}$ and sends $\tilde{\sigma}$ to $\mathcal{A}$ (step 3).

Now, if the adversary does not abort and outputs the take message which stores the deposit in the account pk together with x coins, $\mathcal{S}$ sends the deposit message with the x coins to $\mathcal{F}^{\star}_{CR}$. Then if $\mathcal{S}$ receives the witness w from $\mathcal{F}^{\star}_{CR}$, it adapts the pre-signature $\tilde{\sigma}$ into a full signature σ with w and sends σ to $\mathcal{A}$ (step 4).

Otherwise, if the time period τ passes, $\mathcal{S}$ will receives x coins from $\mathcal{F}^{\star}_{CR}$. Then if $\mathcal{A}$ outputs the take message with a correct signature, $\mathcal{S}$ transfers x coins to $\mathcal{A}$.

$\mathcal{A}$ **corrupts the receiver.** If $\mathcal{S}$ receives any input from the environment $\mathcal{Z}$, it passes the input to the adversary $\mathcal{A}$. Then $\mathcal{S}$ generates a pair of keys (pk_S, sk_S) and exchanges pk_S with the adversary to obtain pk_R from $\mathcal{A}$ (step 1).

Then if $\mathcal{A}$ outputs (Z, π), $\mathcal{S}$ invokes $\mathsf{VP.V}(pp, Z, pk_R, \pi)$ to verify the proof, and if verification rejects, $\mathcal{S}$ records $\mathcal{A}$ cheating (step 2).

Next, if $\mathcal{A}$ outputs a message $(sk_R, \mathsf{Tx}, \phi)$, $\mathcal{S}$ recovers sk from sk_S and sk_R, and then invokes $\mathsf{AS}_{\Pi,\mathcal{C}}.\mathsf{pS}_{sk}(\mathsf{Tx}, \phi)$ to obtain the pre-signature $\tilde{\sigma}$ and sends $\tilde{\sigma}$ to $\mathcal{A}$ (step 3).

Now, the deposit phase ends, and $\mathcal{S}$ receives the deposit message $(\mathsf{deposit}, sid, s, r, \phi^*, \tau, x)$ from $\mathcal{F}^{\star}_{CR}$. Now if $\phi^* \neq \phi$, $\mathcal{S}$ records $\mathcal{A}$ cheating.

In the claim phase, if $\mathcal{A}$ outputs a witness w with $\phi^*(w) \neq 1$, $\mathcal{S}$ records $\mathcal{A}$ cheating. If $\mathcal{A}$ never cheats, $\mathcal{S}$ sends w to $\mathcal{F}^{\star}_{CR}$ and receives x coins from $\mathcal{F}^{\star}_{CR}$. Then it transfers the coins to adv. Otherwise, it does nothing and wait for the refund phase.

We claim that any PPT environment $\mathcal{Z}$ cannot distinguish whether it is interacting with the above simulator $\mathcal{S}$ in the ideal model or it is interacting with the adversary $\mathcal{A}$ in the hybrid model.

$\square$

Together with the main theorem in [11], we have Theorem 2.

Theorem 2. *Let* Π *be the digital signature scheme used by the ledger functionality* $\mathcal{F}^{\star}_{\mathcal{L}}$*. Assuming the correctness and security of the VTLP scheme* VP*, and the adaptor signature scheme* $\mathsf{AS}_{\Pi,\mathcal{R}}$ *and the* $\Theta^{\mathsf{AS}_{\Pi,\mathcal{C}}\cdot\mathsf{pS}}$ *protocol, then for every* n*-party function* f*, there exists a protocol which securely computes* f *and provides fairness with penalty in the* $(\mathcal{F}_{OT}, \mathcal{F}_{\mathcal{L}})$*-hybrid model.*

V. Implementation of *ROSE* for Monero

In the previous section, we present the general construction of the *ROSE* protocol to realize the claim-or-refund functionality $\mathcal{F}^{\star}_{CR}$. In this section, we take Monero, a typical confidential DLS, as an example to show how to efficiently implement the *ROSE* protocol in a real-world DLS.

Specifically, we realize the claim-or-refund functionality $\mathcal{F}^{\star}_{CR}$ for the circuit class $\mathcal{C} = \{\phi_{p,g,y}\}_{p\in\mathbb{P},g,y\in\mathbb{Z}_p^*}$, where $\mathbb{P}$ is the set of prime numbers and $\phi_{p,g,y}(x) = 1 \Leftrightarrow g^x \equiv y \mod p$. This is sufficient to enable fairness with penalty.

A. Realization of The Ledger Functionality

Monero is a typical confidential DLS. It leverages the linkable ring signatures to enhance the confidentiality of transactions on the ledger.

The linkable ring signature scheme MLSAG used in Monero is present in Figure. 9.Here the group generation algorithm $\mathcal{G}$ outputs a group G for which the DDH assumption holds, an generator g of G and the order q of G on the input of security parameter 1^λ, and that the two hash functions $H : \{0,1\}^* \to G$ and $h : \{0,1\}^* \to Z_q$ are collision resistent. The parameter generation algorithm P invokes $\mathcal{G}$ to produce public parameters $pp = (G, g, q)$. The key generation algorithm G uniformly samples $x \in Z_q$ as the secret key and evaluates $y = g^x$ as the public key pk. On inputting n public keys $(y_i)_{i\in[n]}$ and the corresponding secret key x for one of the public keys, namely $y_\pi = g^x$ for some $\pi \in [n]$, the signing algorithm S produces a signature σ for the given message m.

The ledger functionality $\mathcal{F}_{\mathcal{L}}$ can be realized in Monero by publishing transactions. The pay message can be realized by transferring the deposit from the sender to an account represented by a public key pk. And similarly, the take message can be realized by publishing a transaction signed by the corresponding secret key sk, which transfers the deposit to the receiver.

We note that the account pk is just a temporary account, and is not controlled by any party, so the transaction for the take message can be signed with the ring size being only one, and reveals no information. Because the Monero ledger is public to all users, so the take message will be transformed to the sender too.

To realize the claim-or-refund functionality $\mathcal{F}^{\star}_{CR}$ for the circuit class $\mathcal{C} = \{\phi_{p,g,y}\}_{p\in\mathbb{P},g,y\in\mathbb{Z}_p^*}$ with the *ROSE* protocol based on Monero, we further have to construct a VTLP scheme for the circuit class $\mathcal{E} = \{\varepsilon_{g,Y}\}$, in which $\varepsilon_{g,Y}(x) = 1 \Leftrightarrow g^x = Y$. We also need a adaptor signature scheme AS for the circuit class $\mathcal{C}$ and a two party protocol $\Theta^{\mathsf{AS}_{\Pi,\mathcal{C}}\cdot\mathsf{pS}}$ which *UC-securely* realizes the functionality $\mathcal{F}_{\mathsf{pS}}$ defined in Figure. 7. We provide the constructions in the following part of this section.

B. Construction for Verifiable Time-Lock Puzzles

We construct a VTLP scheme VP for the circuit class $\mathcal{E} = \{\varepsilon_{G,g,y}\}$ in this part, where $\varepsilon_{G,g,y}(x) = 1 \Leftrightarrow g^x = y$ and the order of G is q.

```
$\mathsf{P}(1^\lambda)$ :
 1 : $(G, g, q) \leftarrow \mathcal{G}(1^\lambda)$;
 2 : **return** $pp = (G, g, q)$.
$\mathsf{G}(pp)$ :
 1 : **parse** $pp = (G, g, q)$;
 2 : $sk \leftarrow Z_q, pk := g^{sk}$;
 3 : **return** $(pk, sk)$.
$\mathsf{S}_{pp,\pi,sk}((pk_i)_{i\in[n]}, m)$ :

 1 : **parse** $pp = (G, g, q)$;
 2 : $I = H(pk_\pi)^{sk}$;
 3 : $\alpha \xleftarrow{\$} \mathbb{Z}_q, P_\pi = g^\alpha, Q_\pi = H(pk_\pi)^\alpha$;
 4 : **for** $i = \pi+1, \pi+2, ..., n, 1, ..., \pi-1$ :
 5 :     $c_i = h(m, P_{i-1}, Q_{i-1})$;
 6 :     $s_i \xleftarrow{\$} \mathbb{Z}_q$;
 7 :     $P_i = g^{s_i} \cdot y_i^{c_i}$;
 8 :     $Q_i = H(pk_i)^{s_i} \cdot I^{c_i}$;
 9 : $s_\pi = \alpha - c_\pi \cdot sk$;
10 : **return** $\sigma = (I, c_1, (s_i)_{i\in[n]})$.
$\mathsf{V}_{pp,(pk_i)_{i\in[n]}}(m, \sigma)$ :

 1 : **parse** $pp = (G, g, q), \sigma = (I, c_1, (s_i)_{i\in[n]})$;
 2 : **for** $i = 1, 2, ..., n$ :
 3 :     $P_i = g^{s_i} \cdot pk_i^{c_i}$;
 4 :     $Q_i = H(pk_i)^{s_i} \cdot I^{c_i}$;
 5 :     $c_{i+1} = h(m, P_i, Q_i)$;
 6 : **return** $(c_{n+1} == c_1)$.
```

Fig. 9. MLSAG in Monero[50]

Our construction is mainly inspired by the homomorphic time lock puzzles in [44]. From a very basic level, our construction produces a non-interactive zero-knowledge (ZK) proof π for the prover, who generates the time-lock puzzle Z, to prove that the secret x locked in the puzzle meets $g^x = y$ for a given $y \in G$.

We achieve this objective by first providing a public-coin ZK proof protocol $\Pi_{\mathcal{L}_G^{\mathsf{DL}}}$ for the language

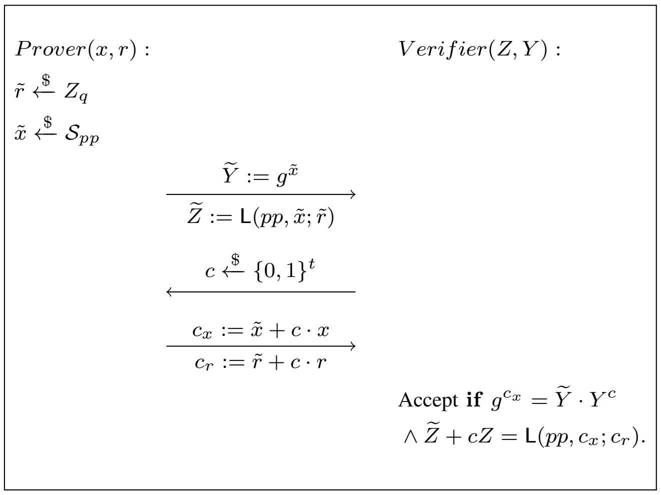
$$\mathcal{L}_G^{\mathsf{DL}} = \{(Z, Y) : \exists x, r, s.t. Z = \mathsf{HP.G}(pp, x; r) \wedge Y = g^x\},$$

where HP is a homomorphic time-lock puzzle scheme.

We have Theorem 3 for the security of the protocol $\Pi_{\mathcal{L}_G^{\mathsf{DL}}}$.

Theorem 3. *Supposing the time-lock puzzle scheme* HP *is correct and secure, and discrete logarithm is hard for the group G, the protocol $\Pi_{\mathcal{L}_G^{\mathsf{DL}}}$ meets completeness and extractability requirement. And it is zero-knowledge if the running time of the verifier is bounded by τ, where τ is the hardness parameter to generate the public parameter pp.*

Proof Sketch. We discuss about the three requirements respectively.



$Prover(x, r)$:
$Verifier(Z, Y)$:
$\tilde{r} \xleftarrow{\$} Z_q$
$\tilde{x} \xleftarrow{\$} \mathcal{S}_{pp}$
$\widetilde{Y} := g^{\tilde{x}}$
$\widetilde{Z} := \mathsf{L}(pp, \tilde{x}; \tilde{r})$
$c \xleftarrow{\$} \{0,1\}^t$
$c_x := \tilde{x} + c \cdot x$
$c_r := \tilde{r} + c \cdot r$
Accept **if** $g^{c_x} = \widetilde{Y} \cdot Y^c$
$\wedge\, \widetilde{Z} + cZ = \mathsf{L}(pp, c_x; c_r)$.

Fig. 10. The ZK proof protocol $\Pi_{\mathcal{L}_G^{\mathsf{DL}}}$.

Completeness. For that the time-lock puzzle scheme HP is linearly homomorphic, so if $Z := \mathsf{L}(pp, x^*; r^*)$ and $\widetilde{Z} := \mathsf{L}(pp, \tilde{x}; \tilde{r})$, then

$$\begin{aligned}\widetilde{Z} + cZ &= \mathsf{L}(pp, \tilde{x}; \tilde{r}) + \mathsf{L}(pp, cx^*; cr^*)\\ &= \mathsf{L}(pp, \tilde{x} + cx^*; \tilde{r} + cr^*)\\ &= \mathsf{L}(pp, c_x; c_r).\end{aligned}$$

apparently holds. And similarly, if $Y = g^{x^*}$, $g^{c_x} = g^{\tilde{x}+cx^*} = g^{\tilde{x}} + (g^{x^*})^c = \widetilde{Y} \cdot Y^r$ also holds. So the verifier will accept the proof.

Extractability. To prove that this ZK proof protocol meets the extractability, we construct a knowledge extractor $\mathcal{K}$ as follows. First, it invokes the prover oracle and challenges with $c = 1$. Now $\mathcal{K}$ has $c_x := \tilde{x} + x$ and $c_r := \tilde{r} + r$. Then it rewinds the prover and challenges it with $c = 0$ to get $c_x' := \tilde{x}$ and $c_r' := \tilde{r}$. Finally, $\mathcal{K}$ can extracts the knowledge of x and r by evaluating $x := c_x - c_x'$ and $r = c_r - c_r'$.

Zero-knowledge. To prove the zero-knowledge property, we aims to simulate the view of the verifier, which can be represented as $(\widetilde{Y}, \widetilde{Z}, c_x, c_r, r)$, where r is the randomness used by it. We simulate its view by uniformly sampling c_x, c_r and c, then let $\widetilde{Y} = g^{c_x} \cdot Y^{-c}$ and $\widetilde{Z} = \mathsf{L}(pp, c_x; c_r) - cZ$. Then we uniformly samples a randomness r and invokes the verifier with input $\widetilde{Y}$ and $\widetilde{Z}$, and the randomness r to obtain its challenge c^*. We repeat this process until $c^* = c$. Finally, we output $(\widetilde{Y}, \widetilde{Z}, c_x, c_r, r)$ as the simulation result.

For that the running time of the verifier is bounded by τ, it cannot tell whether the secret x' locked in $\widetilde{Z}$ meets that $\widetilde{Y} = g^{x'}$. So that its view is indistinguishable. □

The protocol $\Pi_{\mathcal{L}_G^{\mathsf{DL}}}$ can be converted into a non-interactive ZK proof system with Fiat-Shamir heuristic [29] in the random oracle model. Combined with the linearly homomorphic time-lock puzzle scheme given by G. Malavolta et al.[44], we can obtain a verifiable time-lock puzzle scheme VP, which meets the security and correctness requirements.

C. Construction for Two-party Adaptor Signatures

In Figure. 11, we present the adaptor signature scheme $\mathsf{AS}_{\mathsf{MLSAG},\mathcal{C}}$ for the MLSAG scheme and the circuit class $\mathcal{C} = \{\phi_{p,g,y}\}_{g,y\in\mathbb{Z}_p^*}$ for some prime p, where $\phi_{p,g,y}(x) = 1 \Leftrightarrow g^x \equiv y \mod p$.

We note that due to the particularity of linkable ring signatures, the algorithms pS and pV take an auxiliary input z. Supposing $y = g^w$ in $\phi_{p,g,y}$, the auxiliary input $z = H^w(pk)$, where H is the hash function used in the MLSAG scheme.

```
pS_{pp,sk}(m, pk, φ_{p,g,y}, z) :

  1 : parse $pp = (G, g, q)$;
  2 : $I = H^{sk}(pk)$;
  3 : $\alpha \xleftarrow{\$} \mathbb{Z}_q, \tilde{P} = g^{\alpha} \cdot y, \tilde{Q} = H^{\alpha}(pk) \cdot z$;
  4 : $c = h(m, \tilde{P}, \tilde{Q}), \tilde{s} = \alpha - c \cdot sk$;
  5 : return $\tilde{\sigma} = (I, c, \tilde{s})$.
pV_{pp,pk}(m, σ̃, φ_{p,g,y}, z) :

  1 : parse $pp = (G, g, q), \tilde{\sigma} = (I, c, \tilde{s})$;
  2 : $\tilde{P} = g^{\tilde{s}} \cdot pk^c \cdot y$;
  3 : $\tilde{Q} = H(pk)^{\tilde{s}} \cdot I^c \cdot z$;
  4 : else $c' = h(m, \tilde{P}, \tilde{Q})$;
  5 : return $(c' == c)$.
A_{pp}(π, σ̃, w) :

  1 : parse $\tilde{\sigma} = (I, c, \tilde{s})$;
  2 : $s = \tilde{s} + w$;
  3 : return $\sigma = (I, c, s)$.
E_{pp}(π, σ̃, σ) :

  1 : parse $\tilde{\sigma} = (I, c, \tilde{s}), \sigma = (I, c, s)$;
  2 : return $w = s - \tilde{s}$.
```

Fig. 11. The adaptor signature scheme $\mathsf{AS}_{\mathsf{MLSAG},\mathcal{C}}$.

The protocol $\Theta^{\mathsf{AS}}_{\mathsf{MLSAG},\mathcal{C}}$ in Figure. 12 *UC-securely* realizes the functionality in Figure. 7 for the $\mathsf{AS}_{\mathsf{MLSAG},\mathcal{C}}$ adaptor signature scheme.

Similarly to the $\mathsf{AS}_{\mathsf{MLSAG},\mathcal{C}}$.pS algorithm, this protocol has an auxiliary common input $z = H^w(pk)$ where w is the witness.

D. Performance Analysis

In this part, we briefly analyse the performance of our implementation for *ROSE*.

In the deposit phase of *ROSE*, the communication mainly occurs between the sender and the receiver. The communication cost in bits between the two parties in each step of *ROSE* is shown in Table. III, where λ is the security parameter. In this phase, the sender also publishes a transaction on the DLS to store the deposit, which requires about $\lambda + O(1)$ bits of communication cost.

```
Θ^{AS_{MLSAG,C}}_{⟨sk_S, sk_R⟩}(m, pk, φ_{p,g,y}, z):

S :                                                R :
$\alpha_S \xleftarrow{\$} Z_q$                      $\alpha_R \xleftarrow{\$} Z_q$
$P_S := g^{\alpha_S}$                               $P_R := g^{\alpha_R}$
$Q_S := H^{\alpha_S}(pk)$                           $Q_R := H^{\alpha_R}(pk)$

                 $\xleftarrow{P_R, Q_R}$

$P = P_S \cdot P_R \cdot y$
$Q = Q_S \cdot Q_R \cdot z$
$c := h(m, P, Q)$
$\tilde{s}' := \alpha_S - c \cdot sk_S$

                 $\xrightarrow[c, \tilde{s}']{I_S = H^{sk_S}(pk)}$

                                                   $I := I_S \cdot H^{sk_R}(pk)$
                                                   $\tilde{s} := \tilde{s}' + \alpha_R - c \cdot sk_R$
                                                   return $\tilde{\sigma} = (I, c, \tilde{s})$
```

Fig. 12. $\Theta^{\mathsf{AS}_{\mathsf{MLSAG},\mathcal{C}}}$ protocol.

TABLE III
COMMUNICATION COST IN THE DEPOSIT PHASE.

Step	Sender to receiver	receiver to sender
Invoking VTLP	0	5λ
Invoking TPAS	2λ	3λ
Total	2λ	8λ

In the claim phase and the refund phase, the sender and the receiver only needs to communicates with the DLS. The communication cost in bits is shown in Table. IV. In the table, we refer to the sender and receiver as S and R for short respectively.

TABLE IV
COMMUNICATION COST IN THE CLAIM AND REFUND PHASE.

Phase	S to DLS	R to DLS	DLS to S
Claim phase	0	$2\lambda + O(1)$	$2\lambda + O(1)$
Refund phase	$2\lambda + O(1)$	0	0
Total	$2\lambda + O(1)$	$2\lambda + O(1)$	$2\lambda + O(1)$

We also perform experiments to evaluate the computation cost of our implementation for *ROSE* and results are presented in the following. The protocol is implemented using C++ and the Crypto++ and CryptoTools libraries[23], [51] on the Ubuntu OS. And the experiments are performed on a server running the Intel(R) Xeon(R) Gold 5122 CPU.

We first present the experimental results on the efficiency of the VTLP scheme. We investigate time consumption of the parameter generation algorithm P, the locking algorithm L, the verification algorithm V and the unlocking algorithm U of the VTLP scheme running under different security parameters and hardness parameters, and the result is shown in Figure. 13.

From the results, we find that the hardness parameter mainly influences the time consumption of the unlocking algorithm U.

This meets the requirement of our design. And for other three algorithms, the hardness parameter has no apparent affect on their running time. Their efficiency is quite satisfying under all security parameters.

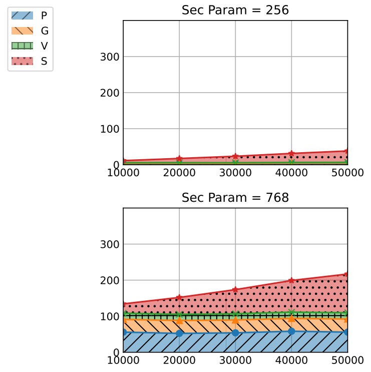


Fig. 13. Time consumption of the four algorithms of the VTLP scheme running under different security parameters. In each graph, the x-axis represents the hardness parameters, and the y-axis represents the time consumption in milliseconds. The time consumption of each algorithm is represented by the space between two lines.

Then we show the time consumption of the pre-verification algorithm of the adaptor signature scheme used in *ROSE* in Figure. 14. We note that the adaptor algorithm and the extractor algorithm are also used in *ROSE*, but their time consumption is negligible under all security parameters, so we omit them here.

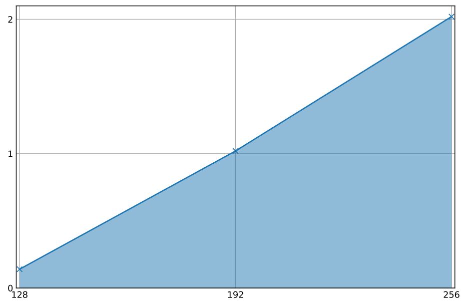


Fig. 14. Time consumption of verification algorithm of the adaptor signature scheme running under different security parameters. The x-axis represents the security parameters, and the y-axis represents the time consumption of the algorithm in milliseconds.

We also investigate the respective running time of the sender and receiver in the two-party pre-signature protocol. The results are collected under different security parameters and is shown in Figure. 15.

From all the above experimental results, we can find that the cryptographic primitives used in *ROSE* is of high efficiency. So applying *ROSE* to enable fairness with penalty in Monero only brings little communication and computation overhead.

VI. Related Works

In this section, we discuss about some related works of this paper.

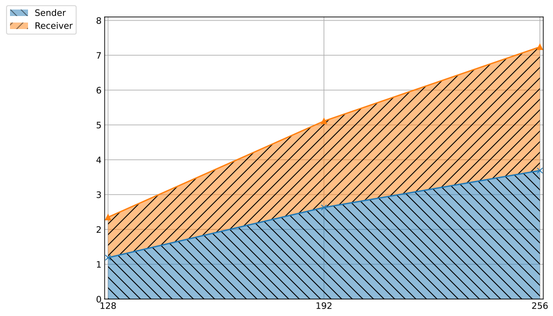


Fig. 15. Time consumption of the sender and the receiver in the two-party pre-signature protocol under different security parameters. The x-axis represents the security parameters, and the y-axis represents the time consumption in milliseconds. The time consumption of the sender and receiver is represented by the space between two lines.

Fairness in MPC has been studied by many works. Fair MPC protocols for general functions with dishonest majority is proven impossible in standard cryptographic model[21]. But it is still possible to design fair MPC protocols for restricted class of function [2], [33], [3], [60], or in restricted settings[10], [22], [20]. Variant models of fairness are also be defined and studied[9], [34], [35]. Bentov et al. study fairness with penalty, and leverage DLSs to make automatic penalty practical without relying on any trusted third-part[11]. In the presence of DLSs, Choudhuri et al. show that complete fairness can also be enabled[19], but stronger assumptions (witness encryption or trusted hardware) needs to be introduced.

Claim-or-refund functionality is first proposed by Bentov et al. [11] to abstract the exact property required from the DLSs to design fair MPC protocols. A prominent line of works [11], [37], [40] shows that $\mathcal{F}^{\star}_{CR}$, along with standard cryptographic primitives, e.g. oblivious transfer (OT), is sufficient for designing fair MPC protocols for general functions. While in these works, $\mathcal{F}^{\star}_{CR}$ is realized by deploying smart contracts on DLSs relying on the scripting functionality. In this paper, we propose the first construction to realize $\mathcal{F}^{\star}_{CR}$ based on DLSs without requiring any scripting functionality, which allows this tool to be compatible with more DLSs.

One main advantage of *ROSE* over previous solutions is its ability to enhance the privacy protection on the participants and deposit amount for that it can realize fair MPC on confidential DLSs, most of which do not provide scripting functionalities. Some existing works, like [39] and [14], present novel designs to **enable scripting functionality on confidential DLSs**. They propose another approach to for privacy protection in fair MPC protocols.

Our construction leverages recent advances in **adaptor signatures**. This tool is applied by many works[5], [28], [55], [25] for various "scriptless" applications in DLSs. We first utilize this tool to enable fair MPC protocols for general functions in a "scriptless" way.

Time-lock puzzles are also frequently used in DLS scenarios. [44] propose the first time-lock puzzle scheme providing homomorphism, which greatly inspires our work. In [56], Thyagarajan et al. define verifiable timed signatures and instantiate it for many celebrated digital signature schemes. Verifiable timed signatures enable one to lock a digital signature in a time-lock puzzle and provide a publicly verifiable proof for its validity, and can be used to enable fairness with penalty in DLS without relying on the time-lock smarts contracts. In this paper, we extend their definition and define verifiability for general circuits. We also provide a concrete instantiation for the discrete logarithm circuits and enable fairness with penalty without relying on any scripting functionality, including the time-lock smart contracts.

VII. Conclusion

In this paper, we propose *ROSE* to realize the claim-or-refund functionality $\mathcal{F}^{\star}_{CR}$ on DLSs. The main novelty of our construction is that it does not require any scripting functionality from the underlying DLSs. It only requires that the users of the underlying DLS can transfer digital currencies among accounts represented by public keys of a digital signature scheme. And digital currencies can be transferred from the account pk to another one with a transaction signed by its corresponding secret key. Besides, the signatures are broadcasted to all users of the DLS together with the transactions. Furthermore, our result first shows that fair MPC protocols for general functions can be realized without requiring scripting functionality from DLSs.

Besides, we also present an efficient implementation of *ROSE* on Monero, which is one of the most well-known confidential DLSs. We theoretically prove that our implementation meets the security requirements and conduct experiments to evaluate efficiency of our implementation. The experimental results show that applying our implementation for *ROSE* to enable fair MPC protocol in Monero only brings negligible overhead in communication and computation.

References

[1] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek, "Secure multiparty computations on bitcoin," in *2014 IEEE Symposium on Security and Privacy*. IEEE, 2014, pp. 443–458.

[2] G. Asharov, "Towards characterizing complete fairness in secure two-party computation," in *Theory of Cryptography Conference*. Springer, 2014, pp. 291–316.

[3] G. Asharov, A. Beimel, N. Makriyannis, and E. Omri, "Complete characterization of fairness in secure two-party computation of boolean functions," in *Theory of Cryptography Conference*. Springer, 2015, pp. 199–228.

[4] L. Aumayr, O. Ersoy, A. Erwig, S. Faust, K. Hostáková, M. Maffei, P. Moreno-Sanchez, and S. Riahi, "Generalized bitcoin-compatible channels." *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 476, 2020.

[5] ——, "Generalized channels from limited blockchain scripts and adaptor signatures," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2021, pp. 635–664.

[6] L. Aumayr, P. Moreno-Sanchez, A. Kate, and M. Maffei, "Blitz: Secure multi-hop payments without two-phase commits," in *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.

[7] W. Banasik, S. Dziembowski, and D. Malinowski, "Efficient zero-knowledge contingent payments in cryptocurrencies without scripts," in *European symposium on research in computer security*. Springer, 2016, pp. 261–280.

[8] M. Bartoletti and L. Pompianu, "An empirical analysis of smart contracts: platforms, applications, and design patterns," in *International conference on financial cryptography and data security*. Springer, 2017, pp. 494–509.

[9] C. Baum, B. David, and R. Dowsley, "Insured mpc: Efficient secure computation with financial penalties," in *International Conference on Financial Cryptography and Data Security*. Springer, 2020, pp. 404–420.

[10] A. Beimel, Y. Lindell, E. Omri, and I. Orlov, "1/p-secure multiparty computation without honest majority and the best of both worlds," in *Annual Cryptology Conference*. Springer, 2011, pp. 277–296.

[11] I. Bentov and R. Kumaresan, "How to use bitcoin to design fair protocols," in *Annual Cryptology Conference*. Springer, 2014, pp. 421–439.

[12] I. Bentov, R. Kumaresan, and A. Miller, "Instantaneous decentralized poker," in *International conference on the theory and application of cryptology and information security*. Springer, 2017, pp. 410–440.

[13] A. Biryukov and S. Tikhomirov, "Deanonymization and linkability of cryptocurrency transactions based on network analysis," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019, pp. 172–184.

[14] S. Bowe, A. Chiesa, M. Green, I. Miers, P. Mishra, and H. Wu, "Zexe: Enabling decentralized private computation," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 947–964.

[15] M. Campanelli, R. Gennaro, S. Goldfeder, and L. Nizzardo, "Zero-knowledge contingent payments revisited: Attacks and payments for services," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 229–243.

[16] Y. Chen, X. Tian, Q. Wang, J. Jiang, M. Li, and Q. Zhang, "Safe: A general secure and fair auction framework for wireless markets with privacy preservation," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[17] U. W. Chohan, "Cryptocurrencies: A brief thematic review," *Available at SSRN 3024330*, 2017.

[18] A. R. Choudhuri, V. Goyal, and A. Jain, "Founding secure computation on blockchains," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2019, pp. 351–380.

[19] A. R. Choudhuri, M. Green, A. Jain, G. Kaptchuk, and I. Miers, "Fairness in an unfair world: Fair multiparty computation from public bulletin boards," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 719–728.

[20] K.-M. Chung, T.-H. H. Chan, T. Wen, and E. Shi, "Game-theoretic fairness meets multi-party protocols: The case of leader election," in *Annual International Cryptology Conference*. Springer, 2021, pp. 3–32.

[21] R. Cleve, "Limits on the security of coin flips when half the processors are faulty," in *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, 1986, pp. 364–369.

[22] D. Dachman-Soled, "Revisiting fairness in mpc: Polynomial number of parties and general adversarial structures," in *Theory of Cryptography Conference*. Springer, 2020, pp. 595–620.

[23] W. Dai, "Crypto++ library," 2010.

[24] P. Das, L. Eckey, T. Frassetto, D. Gens, K. Hostáková, P. Jauernig, S. Faust, and A.-R. Sadeghi, "Fastkitten: Practical smart contracts on bitcoin," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 801–818.

[25] A. Deshpande and M. Herlihy, "Privacy-preserving cross-chain atomic swaps," in *International Conference on Financial Cryptography and Data Security*. Springer, 2020, pp. 540–549.

[26] S. Dziembowski, L. Eckey, and S. Faust, "Fairswap: How to fairly exchange digital goods," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 967–984.

[27] A. Erwig, S. Faust, K. Hostáková, M. Maitra, and S. Riahi, "Two-party adaptor signatures from identification schemes." in *Public Key Cryptography (1)*, 2021, pp. 451–480.

[28] M. F. Esgin, O. Ersoy, and Z. Erkin, "Post-quantum adaptor signatures and payment channel networks," in *European Symposium on Research in Computer Security*. Springer, 2020, pp. 378–397.

[29] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Conference on the theory and application of cryptographic techniques*. Springer, 1986, pp. 186–194.

[30] A. Gaihre, S. Pandey, and H. Liu, "Deanonymizing cryptocurrency with graph learning: The promises and challenges," in *2019 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2019, pp. 1–3.

[31] P. Gaži, A. Kiayias, and D. Zindros, "Proof-of-stake sidechains," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 139–156.

[32] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 3–16.

[33] S. D. Gordon, C. Hazay, J. Katz, and Y. Lindell, "Complete fairness in secure two-party computation," *Journal of the ACM (JACM)*, vol. 58, no. 6, pp. 1–37, 2011.

[34] Y. Ishai, J. Katz, E. Kushilevitz, Y. Lindell, and E. Petrank, "On achieving the "best of both worlds" in secure multiparty computation," *SIAM journal on computing*, vol. 40, no. 1, pp. 122–141, 2011.

[35] Y. Ishai, E. Kushilevitz, Y. Lindell, and E. Petrank, "On combining privacy with guaranteed output delivery in secure multiparty computation," in *Annual International Cryptology Conference*. Springer, 2006, pp. 483–500.

[36] A. Kiayias, E. Koutsoupias, M. Kyropoulou, and Y. Tselekounis, "Blockchain mining games," in *Proceedings of the 2016 ACM Conference on Economics and Computation*, 2016, pp. 365–382.

[37] A. Kiayias, H.-S. Zhou, and V. Zikas, "Fair and robust multi-party computation using a global transaction ledger," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2016, pp. 705–734.

[38] A. Kiayias and D. Zindros, "Proof-of-work sidechains," in *International Conference on Financial Cryptography and Data Security*, 2019, pp. 21–34.

[39] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *2016 IEEE symposium on security and privacy (SP)*. IEEE, 2016, pp. 839–858.

[40] R. Kumaresan and I. Bentov, "Amortizing secure computation with penalties," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 418–429.

[41] R. Kumaresan, T. Moran, and I. Bentov, "How to use bitcoin to play decentralized poker," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 195–206.

[42] R. Kumaresan, V. Vaikuntanathan, and P. N. Vasudevan, "Improvements to secure computation with penalties," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 406–417.

[43] R. W. Lai, V. Ronge, T. Ruffing, D. Schröder, S. A. K. Thyagarajan, and J. Wang, "Omniring: Scaling private payments without trusted setup," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 31–48.

[44] G. Malavolta and S. A. K. Thyagarajan, "Homomorphic time-lock puzzles and applications," in *Annual International Cryptology Conference*. Springer, 2019, pp. 620–649.

[45] E. V. Mangipudi, K. Rao, J. Clark, and A. Kate, "Towards automatically penalizing multimedia breaches," in *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2019, pp. 340–346.

[46] R. Maull, P. Godsiff, C. Mulligan, A. Brown, and B. Kewell, "Distributed ledger technology: Applications and implications," *Strategic Change*, vol. 26, no. 5, pp. 481–489, 2017.

[47] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from bitcoin," in *2013 IEEE Symposium on Security and Privacy*. IEEE, 2013, pp. 397–411.

[48] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2019.

[49] S. Noether, "Ring signature confidential transactions for monero." *IACR Cryptol. ePrint Arch.*, vol. 2015, p. 1098, 2015.

[50] S. Noether, A. Mackenzie *et al.*, "Ring confidential transactions," *Ledger*, vol. 1, pp. 1–18, 2016.

[51] P. Rindal, "CryptoTools," https://github.com/ladnir/cryptoTools/.

[52] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE Symposium on Security and Privacy*. IEEE, 2014, pp. 459–474.

[53] S.-F. Sun, M. H. Au, J. K. Liu, and T. H. Yuen, "Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero," in *European Symposium on Research in Computer Security*. Springer, 2017, pp. 456–474.

[54] E. Tairi, P. Moreno-Sanchez, and M. Maffei, "A2l: Anonymous atomic locks for scalability in payment channel hubs," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 1834–1851.

[55] ——, "Post-quantum adaptor signature for privacy-preserving off-chain payments," in *International Conference on Financial Cryptography and Data Security*. Springer, 2021, pp. 131–150.

[56] S. A. K. Thyagarajan, A. Bhat, G. Malavolta, N. Döttling, A. Kate, and D. Schröder, "Verifiable timed signatures made practical," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1733–1750.

[57] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[58] A. Yakovenko, "Solana: A new architecture for a high performance blockchain v0. 8.13," *Whitepaper*, 2018.

[59] T. H. Yuen, S.-f. Sun, J. K. Liu, M. H. Au, M. F. Esgin, Q. Zhang, and D. Gu, "Ringct 3.0 for blockchain confidential transaction: Shorter size and stronger security," in *International Conference on Financial Cryptography and Data Security*. Springer, 2020, pp. 464–483.

[60] E. Zhang, M. Li, S.-M. Yiu, J. Du, J.-Z. Zhu, and G.-G. Jin, "Fair hierarchical secret sharing scheme based on smart contract," *Information Sciences*, vol. 546, pp. 166–176, 2021.

[61] Z. Zhao and T.-H. H. Chan, "How to vote privately using bitcoin," in *International Conference on Information and Communications Security*. Springer, 2015, pp. 82–96.

[62] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *International Journal of Web and Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.

[63] W. Zou, D. Lo, P. S. Kochhar, X.-B. D. Le, X. Xia, Y. Feng, Z. Chen, and B. Xu, "Smart contract development: Challenges and opportunities," *IEEE Transactions on Software Engineering*, 2019.

Appendix