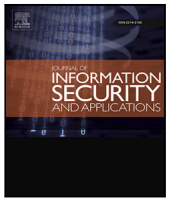
Journal of Information Security and Applications 83 (2024) 103794

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa



ZeroMT: Towards Multi-Transfer transactions with privacy for account-based blockchain☆

Emanuele Scala [a,*], Changyu Dong [b], Flavio Corradini [a], Leonardo Mostarda [c]

[a] Department of Computer Science, University of Camerino, Italy
[b] Institute of AI and Blockchain, Guangzhou University, China
[c] Department of Mathematics and Computer Science, University of Perugia, Italy

ARTICLE INFO

Keywords:
Zero-knowledge
Blockchain
Confidential transactions
Homomorphic encryption
Multi-transfer

ABSTRACT

The public blockchain lacks data confidentiality. Although a level of anonymity seems guaranteed, it is still possible to link transactions and disclose related information. A solution to the privacy problem is to use cryptography in transactions, however this can lead to increased costs and slowdown in network throughput. Recent works experiment with advanced cryptography, in particular Zero-Knowledge proofs (ZK-proofs) can be supplied within a transaction to prove its validity, without revealing sensitive information. We analyze solutions that adopt ZK-proofs, such as Confidential Transactions (CTs). Several challenges emerge depending on both the zero-knowledge system and the balance model considered (UTXO, hybrid or account model). For ZK-proofs, systems that do not introduce additional trust are required. On the other hand, the account model is the most flexible for addressing security challenges. Moreover, CTs do not fully exploit the potential of ZK-proofs, since each transaction comes with one or more ZK-proof for a single transfer. Within this paper, we present ZeroMT, a novel *multi-transfer* private payment scheme for account-based blockchains. Drawing inspiration from Zether, our approach extends their work to develop a payment model that supports multiple payees within a single transaction. This also benefits scalability: ZeroMT enriches the CTs with the aggregation property, i.e., the batch verification of multiple transfers from a single and aggregate proof. We show that in our extended model the overdraft-safety and privacy security properties still hold. We provide an implementation and evaluation of ZeroMT, which shows the benefits of aggregating multiple transfers.

1. Introduction

Bitcoin [1] introduces the first decentralized finance system based on the Unspent Transaction Outputs (UTXOs) model on top of blockchain theory. In short, a user who owns coins is linked to the corresponding unspent outputs, which can be spent by designating them to a recipient as new outputs of a transaction. Ethereum [2] has enhanced the blockchain by implementing a decentralized computation system with the so-called smart contracts. With these contracts, Ethereum is able to move the balance representation of a user from the UTXO model to one that is more flexible and semantically close to reality, the account model. In this model, a numerical quantity owned by a user or contract address serves as a measure of balance. The balances are updated when funds are transferred between two addresses, resulting in a debit to the sender's account and a credit to the receiver's account. While these solutions promise a level of anonymity for users, it is well-known that there are kinds of analyses on the transaction graph that can link parties to a transaction and even reveal their real-world identities [3–6]. Moreover, Bitcoin and Ethereum transactions are publicly announced and propagated to all nodes of the blockchain. Since there are no mechanisms to ensure confidentiality, all information related to the transactions is publicly visible, resulting in privacy concerns for users in many applications. From these issues, a large body of research is underway for the design of privacy solutions such as *Confidential Transactions* (CTs), with the aim of guaranteeing user anonymity and data confidentiality in transactions. The relevant solutions in CTs are based on the *Zero-Knowledge proofs* (ZK-proofs), as a mechanism to validate transactions without providing the sensitive information to the blockchain network. In general, these solutions define specific statements for which the cryptographic ZK-proof is constructed, and then each transaction is supplied with that proof to the ledger maintainers. An important feature of the proof is that it reveals nothing except the validity of the statements, and therefore can be verified without any

☆ This article is the extension of our previous works published in AINA 2022 (Corradini et al., 2022) and AINA 2023 (Scala et al., 2023).

* Corresponding author.
E-mail addresses: emanuele.scala@unicam.it (E. Scala), changyu.dong@gzhu.edu.cn (C. Dong), flavio.corradini@unicam.it (F. Corradini), leonardo.mostarda@unipg.it (L. Mostarda).

https://doi.org/10.1016/j.jisa.2024.103794

Available online 29 May 2024

leakage of information. Several privacy-preserving protocols have been proposed following the ZK-proofs paradigm as an underlying mechanism. The UTXO-based proposals use ZK-proofs to prove the validity of the equality relating input and output amounts of a transaction [7–10]. The most common technique concerns the construction of the specific computation made private through an arithmetic circuit, for example, using *zero-knowledge succinct non-interactive argument of knowledge* (zk-SNARK) schemes. The account-based proposals adopt the strategy of keeping amounts encrypted in transactions, and balances are updated homomorphically [11,12]. In this setting, common techniques rely on arguments of knowledge systems for *range proofs*, to prove that certain values are non-negative, and for proving algebraic statements over DLOG relations [13,14]. Although most protocols divide on one of two models, there exist hybrid solutions that build on UTXO logic on top of the account model, mainly using private computation circuits based on zk-SNARKs for transaction correctness [15,16].

1.1. Limitations derived from ZK-proofs of CT protocols

First, we recall the meaning of *trustless* and *succinctness*. *Trustless* in our context means a scheme should not rely on trusted hardware and/or trusted executors to perform private transfers. Existing schemes typically involve a *Trusted Execution Environment* (TEE) or a trusted third-party in the protocol working out of the main-chain together with the other entities of the system. Some other schemes may not be completely trustless when making strong security assumptions that require additional trust. As an example, zk-SNARKs constructions require a trusted third-party to generate a Common Reference String (CRS) [17]. This CRS is then distributed among all participants to enable the generation and verification of proofs. However, malicious users compromising the CRS could break the soundness of the entire protocol by producing valid false proofs. To overcome this problem, a secure multi-party computation for the distribution of the CRS has been proposed in [18], but it is difficult and costly to manage among a large set of participants. Moreover, a dishonest user could participate with multiple identities by imposing his unfair influence on the resulting CRS [16]. Another drawback is that CRS models are circuit-dependent, which means that a generation of a new CRS must be done for any change to the circuit.

The *succinctness* definition is strictly related to the proof size and verifier time of the proof system. In particular, a proof system is succinct if the proof is compact and is of constant size and the verifier runs in time $\mathcal{O}(|x|)$ on the size of the input x [8]. Groth16 [19] is the closest proof system to this definition and all the solutions that adopt it benefit from succinctness. However, the pre-processing of the CRS still remains a known drawback.

Known alternatives to the CRS-based proof systems are the *zero-knowledge argument of knowledge* systems of Bootle et al. [20] and its improvement proposed in Bulletproofs [13]. They present the *Inner Product Argument* (IPA), a method of satisfying an inner product relation (used in Bulletproofs as a subroutine for range proofs), which depends on weaker hardness assumptions (the soundness is ensured under the discrete logarithm assumption in standard groups).

Nonetheless, this class of proof systems is used in CTs to achieve the trustless property, but at the expense of succinctness, since the proofs generated are not of constant size. Although the challenge of succinctness remains, in our paper we follow the direction of ZK-proofs based on DLOG assumptions which eliminate the need for a trusted setup. A further motivation is that there are recent works trying to optimize the verifier time and proof size of the IPA protocol. Among these works, Bowe et al. [21] propose the *amortized succinctness* from a *polynomial commitment scheme*, used in an argument of knowledge for commitments employed in the IPA. With this method, the burden of the verifier is amortized by performing costly operations outside the verifier's process and adopting batching techniques for the ZK-proofs through *accumulators*. Bünz et al. [22] establish a generalized result from the previous one, showing that any polynomial commitment schemes under DLOG assumptions can support accumulation schemes. To conclude this line of research, there are optimizations on the IPA based on pairing-friendly groups, landing into the *inner-pairing product* [23–26]. However, such schemes may incur costly pairing operations for the verifier [27].

1.2. Limitations and security requirements of CT protocols

Several limitations arise when approaching the balance model to the privacy-preserving protocol. It is difficult to deal with confidentiality when coins with fixed denominations, even encrypted, are considered in a UTXO transaction [9], given the possibility of deriving other information such as the number of coins exchanged. Moreover, UTXO-based protocols such as Zerocash [8] and Monero [10] suffer from continuous growth in the size of the UTXO set [11,28], due to the method of mixing addresses in transactions. This incurs storage problems considering peers on the network do not have a concise UTXO representation, which is possible in Bitcoin instead. Quisquis [11] aims at solving the aforementioned issues by adopting a hybrid balance model together with the *updatable public keys* primitive, which allows the update of all the output addresses with each transaction. However, this mechanism is not resistant to *front-running* attacks [12,28], and the hybrid balance models can lead to increased gas costs for storage when integrated into practice in smart contract platforms [12].

Pure account-based privacy mechanisms can provide a natural way to solve some of the above issues. Transactions can deal directly with the amounts to be transferred to/from an account as a numerical quantity, hence fixed coin denominations do not pose confidentiality issues. There are no unspent outputs for each user to be stored in a global state, each transaction draws directly from the total owned by users, leading to an efficient way of managing balances. Moreover, in account-based blockchains such as Ethereum, smart contracts can be used to address security requirements flexibly at the protocol level, without any changes to the underlying system. In this direction, it is worth to mention Zether [12], an account-based private payment mechanism built for smart contract protocols, which solves security challenges of *front-running* and *replay attacks*. Other important security requirements can be addressed with respect to the soundness and zero-knowledge properties of the adopted proof system, respectively the *overdraft-safety* and *privacy* security properties [12]. The overdraft-safety definition ensures that an adversary cannot withdraw more money than it has from an account. The definition of privacy ensures that no information is leaked to an adversary about the payments of honest users. Given the advantages of the model discussed above, we follow the direction of account-based CT protocols. Moreover, we treat the ZK-proofs as non-blackbox, since in our analysis the security requirements are closely tied to the security properties of the proof system.

1.3. New directions of CT protocols: multi-transfer transactions

A new direction of CT protocols, from which we develop our central contribution, is the definition of a new payment method in which it is possible to simultaneously spend multiple amounts and update the balances of multiple payees within a single transaction. We identify such a payment model with the name of *multi-transfer*. A concept of "redistribution of wealth" was previously introduced in the work of Quisquis [11] and exhibits similarities to the multi-transfer model: a transaction takes place between a set of participants who can move coins to many recipients, on behalf of the authorization of the "true" sender. Other works, although based on the traditional payment model, propose the multi-transfer as a future extension and argue that this new payment system could be used to reduce the number of transactions submitted by users [12,16,29]. However, accommodating multiple payees in a confidential transaction presents several challenges in terms of both scalability and security. In the following, we highlight the main challenges of a multi-transfer model in CT settings:

- Most of the cost of a confidential transaction comes from the ZK-proofs. Traditional CTs provide one or more ZK-proofs, e.g., one for each statement, for a single transfer. Having multiple payees could increase the number of ZK-proofs within the transaction. This leads to higher blockchain space when the proofs are stored in blocks, and higher fees when the proofs are verified by ledger maintainers. To address these issues, we propose a method to reduce costs in account-based privacy solutions. In particular, the transaction fees can be amortized by performing multiple transfers in a single transaction equipped with an aggregate ZK-proof. Hence, by proving the validity of multiple transfers within one proof, the cost that a transaction would have for a single transfer is now spread across multiple transfers. Moreover, this also reduces the number of transactions to be validated on-chain and saves blockchain space.
- The choice of the balance model imposes a limit on the number of transfers that can be made in a single transaction. In UTXO model we have the worst flexibility, since increasing the number of inputs of a transaction results in a significant decrease in performance. This because the statements to prove behind a transfer are strictly close to the underlying blockchain mechanisms, e.g., it is not know how to prove efficiently multiple Merkle paths [16]. In contrast, the account model is more flexible and the only limit on the number of transfers is imposed by the maximum gas consumption of a transaction.
- The choice of the zero-knowledge proof system determines overall performance. In particular, the verification time and the proof size of an aggregate ZK-proof grow with the number of transfers and, therefore, with the size of the statements for which the proof is constructed. This is the case of trustless ZK-proofs, which offer security benefits due to weaker assumptions but come at the cost of lower performance. Our choice of trustless ZK-proofs is still optimal in the case of multi-transfer, but could be improved thanks to some amortization strategies, especially on verification time. We do not cover such optimization in this paper, but we provide a discussion in Section 9.
- Moving from the single receiver payment model to the multi-transfer model is not a trivial task from a security perspective. Privacy solutions design specialized zero-knowledge proof systems for static statements that need to be extended. This process could cause the violation of the security of the ZK-proof. Furthermore, extending the payment model introduces the potential risk of compromising the security of the high-level protocol. In our *multi-transfer scheme* we are careful to comply with the security definitions of the reference protocol and the proof system.

1.4. Our contribution

In this paper we propose ZeroMT, a new *multi-transfer* private payment scheme for CT protocols. Our methodology covers both theoretical and practical aspects. From a theoretical point of view, non-blackbox zero-knowledge proofs are employed and designed to comply with the fundamental security requirements in the case of multi-transfer. The bridge between low-level (ZK-proofs) and high-level (cryptographic scheme) security is built on the basis of rigorous definitions and theorems. We end up with a generalized and secure cryptographic scheme for multi-transfer, suitable for account-based blockchain and that can be easily integrated into CT protocols. From a practical point of view, our methodology is supported by the development of a library, that is modular and implements both the interactive and non-interactive zero-knowledge protocols underlying our ZK-proofs. The library also combines the several modules in order to implement the multi-transfer scheme. Moreover, the multi-transfer finds many applications in complex smart contract scenarios where transactions involve more than just two users. We present real-world case studies in Section 8.

In summary, for the benefits of the account model discussed in the previous sections, our starting point is Zether [12], which we move from the traditional payment model to a model in which there are multiple payees in one transaction. Hence, we design the multi-transfer scheme based on the absence of trusted setup and standard cryptographic assumptions, and we show that overdraft-safety and privacy hold in the extended model. We also remark that proving and verifying statements for multiple transfers within a single transaction is more convenient compared to proving and verifying statements for each individual transfer separately. In other words, we construct our proof based on zero-knowledge relations suitable for proving the validity of multiple transfers in one proof. This enriches the CT protocols with the new property of *aggregation*, i.e., batch verification of multiple transfers from a single aggregate proof. As part of our contribution, we implement ZeroMT using arkworks Rust libraries [30]. Finally, we provide concrete evaluations in terms of ZK-proof performance (proof size, prover and verifier time) and transaction costs. This paper is the extension of our two previous works [29,31]: in the first, we present the notion of multi-transfer from a concrete case study; in the second, we initiate the design of our multi-transfer ZK-proof. These papers have been enhanced in the following way: we define the complete cryptographic scheme of the multi-transfer; we provide rigorous security definitions that bridge the cryptographic scheme with the ZK-proof and prove them; we improve the evaluation and comparison with concurrent works; we release a modular and open-source library, through which we evaluate the proposed scheme.

1.5. Organization

The paper is organized as follows: Section 2 gives the cryptographic background; Section 3 provides a summary of the main Zether's concepts; Section 4 introduces the Multi-Transfer payment mechanism, the related cryptographic scheme and security definitions; Section 5 goes deep on the interactive zero-knowledge proof system; Section 6 presents related work and comparison; Section 7 outlines the implementation and evaluation of our system along with a comparison of concurrent works; Section 8 elaborates applications of the multi-transfer scheme; Section 9 outlines the limitations of the multi-transfer scheme; Section 10 are the conclusions and future directions.

2. Preliminaries

In what follows, PPT denotes *probabilistic polynomial-time*, $\lambda \in \mathbb{N}$ denotes the security parameter, and $s \xleftarrow{\$} S$ indicates a random variable s uniformly sampled from the set S.

2.1. ElGamal homomorphic encryption

$\mathcal{G}$ is a *group-generation* function that takes 1^λ as input and produces $(\mathbb{G}, p, g)$, where $\mathbb{G}$ describes a *cyclic group*, p (prime number) is the order of the group (with bit-length $p = \lambda$) and $g \in \mathbb{G}$ is a group element generator. *ElGamal encryption* is a public key cryptosystem that is secure under the Decisional Diffie–Hellman (DDH) assumption [32]. Informally, the DDH problem corresponds to the inability of any adversary to distinguish a uniform group element $h' = g^z$ from another element of the form $h = g^{x \cdot y}$ given the tuple $(\mathbb{G}, p, g, g^x, g^y)$, for uniformly chosen $x, y, z \in \mathbb{Z}_p$.

Considering groups where the DDH problem is intractable, the ElGamal encryption scheme has the public parameters $(\mathbb{G}, p, g, y)$, where y is the public key of the form $y = g^x$, for a uniformly chosen private key x from the set of inverses $\mathbb{Z}_p^\star$ of $\mathbb{Z}_p$. Let r be a uniformly random number chosen in $\mathbb{Z}_p^\star$, the ciphertext for a message m is derived as follows: $Enc_y(m) = (C_L, C_R) = (m \cdot y^r, g^r)$. Only if the private key x is known, it is possible to compute $C_R^x = (g^r)^x = y^r$, and then decrypt the ciphertext using the inverse such that $m = C_L \cdot (y^r)^{-1}$.

In our paper, we use the specific instance of the ElGamal scheme of Zether [12]. In particular, the message to be encrypted is a value $a \in \mathbb{Z}_p$, which is transformed into a group element by calculating g^a. Consequently, the encryption of that value is $Enc_y(a) = (C_L, C_R) = (g^a \cdot y^r, g^r)$. Moreover, this makes ElGamal encryption *additively homomorphic*. Given two distinct values a and a', encrypting them under the same public key y and respectively under the randomness r and r', we have the two ciphertexts $(C_L = g^a \cdot y^r, C_R = g^r)$, and $(C'_L = g^{a'} \cdot y^{r'}, C'_R = g^{r'})$. By applying the operation $(C_L, C_R) \cdot (C'_L, C'_R)$, it is possible to derive the ciphertext associated with $a + a'$.

2.2. Zero-knowledge proofs

Let $\mathcal{R}$ be a polynomial-time decidable relation $\mathcal{R} = \{(\sigma, x, w)\}$, for a common reference string σ, some instances x and witnesses w, and $\mathcal{L}$ the corresponding NP language such that $\mathcal{L} = \{x \mid \exists w : (\sigma, x, w) \in \mathcal{R}\}$. In an interactive proof, a prover $\mathcal{P}$ engages in a two-party protocol with a verifier $\mathcal{V}$. The goal is for $\mathcal{P}$ to convince $\mathcal{V}$ that an instance x belongs to the language $\mathcal{L}$ according to the specified relation $\mathcal{R}$. We posit a triple of PPT algorithms $\Pi = (\text{Setup}, \mathcal{P}, \mathcal{V})$ where:

Setup: a PPT algorithm that on input λ security parameter, generates the reference string $\sigma \leftarrow \text{Setup}(1^\lambda)$. Here, we consider σ a *common random string* without a trapdoor and can be generated with random oracles.

Prover: $\mathcal{P}$ is a PPT algorithm that on input (σ, x, w), for a tuple of instances and witnesses $(x, w) \in \mathcal{R}$, outputs a cryptographic proof π.

Verifier: $\mathcal{V}$ is a PPT algorithm that on input (σ, x, π) outputs 0 or 1, namely it rejects or accepts the ZK-proof π.

In the context of Π, an interactive proof involves a series of message exchanges between $\mathcal{P}$ and $\mathcal{V}$, forming a *transcript*. Based on this transcript, $\mathcal{V}$ can either accept or reject the conversation. A proof is considered *zero-knowledge* when it discloses no information other than the validity of the statements being proved. An interactive proof for Π is *Special Honest-Verifier Zero-Knowledge* (SHVZK) if it has the following properties:

Definition 2.1 (*Perfect Completeness [33]*)**.** If the prover follows the protocol then the verifier will accept with probability 1.

Definition 2.2 (*Special Soundness [33]*)**.** There exists a polynomial-time algorithm $\mathcal{E}$, called *extractor*, which is given as input a pair of accepting transcripts (α, c, ω) and (α, c', ω') with α the first prover's message, $c \neq c'$ two distinct verifier's challenges, ω and ω' the final prover's messages, and it always computes a witness w satisfying $(x, w) \in \mathcal{R}$.

The extractor $\mathcal{E}$ is allowed to rewind the protocol between $\mathcal{P}$ and $\mathcal{V}$ in such a way that after the prover sends the first message α, it restarts the protocol with a new challenge c', and $\mathcal{E}$ extracts the witness. This implies that we have a *proof of knowledge* for the witness w, demonstrating that $(x, w) \in \mathcal{R}$.

Definition 2.3 (*Spacial Honest-Verifier Zero-Knowledge [33]*)**.** There must be a PPT simulator $\mathcal{S}$ that takes as input the public instances and the verifier's internal randomness, and outputs transcripts such that the distribution over transcripts generated by $\mathcal{S}$ is identical to the distribution over the transcripts produced by the honest verifier and prover.

In a *public-coin* SHVZK protocol, all the messages sent by the verifier to the prover are sampled uniformly at random, hence they are independent of the prover's messages. The verifier's random choices (challenges) are made public.

Definition 2.4 (*Σ-protocol*)**.** A Σ-protocol is a 3-round public-coin interactive proof.

Σ-protocols are commonly expected to have the properties of perfect completeness, special soundness and special honest-verifier zero-knowledge. The 3-round structure allows the possibility to integrate an SHVZK interactive Σ-protocol with other interactive proofs. Moreover, using the Fiat–Shamir heuristic [34], it is possible to convert an interactive zero-knowledge proof into a *non-interactive* zero-knowledge proof (NIZK). For a Σ-protocol with messages (α, c, ω), an honest prover replaces the verifier's challenge c with a *random oracle*. The non-interactive version inherits the security properties of the interactive protocol. In practice, the Fiat–Shamir transformation is implemented by replacing the verifier's challenges in the interactive proof with the hash of messages from previous interactions with the current challenge. By hashing messages that are valid as a signature, the transformation can turn a public-coin interactive proof into a signature scheme [35,36].

3. Background on Zether

In this section, we summarize the basics of Zether [12]. Zether's CT protocol is built for account-based blockchain and ensures compatibility with Ethereum as well as other smart contract platforms. The payment mechanism comprises a setup algorithm, user-accessible algorithms, and the Zether Smart Contract (ZSC). The setup generates the public parameters for the Zether proof systems, called Σ-Bullets, and the signature scheme, both based on trustless subroutines. After the setup, the global parameters for the smart contract are initialized and the ZSC is deployed. This smart contract keeps track of all the Zether accounts and their respective encrypted balances. Zether accounts are identified by their ElGamal public key $y \in \mathbb{G}$ in the smart contract. Such accounts are generated using the `CreateAddress` user algorithm, which derives a randomly sampled private key $sk \xleftarrow{\$} \mathbb{Z}_p$ and the respective public key $y = g^{sk}$, where g is a generator of the group $\mathbb{G}$. A Zether balance for a user with public key y is stored in ZSC in the form of ElGamal encryption $(C_L = g^b y^r, C_R = g^r)$, where b is the current balance value and $r \xleftarrow{\$} \mathbb{Z}_p$ the randomness used in the encryption.

Accounts can interact with the ZSC through their public key, to fund other accounts, transfer currency or obtain their balance. To do so, accounts have access to the following user algorithms: (i) `Fund`, by which a user can transfer some ETH to the account having public key y. The ETH amount is converted into ZTH amount that is (homomorphically) added to the encrypted y's balance; (ii) `Burn`, by which a user having public key y requests the entire associated balance in ETH. A conversion is made from ZTH to ETH; (iii) `Transfer`, by which an account with public key y confidentially transfers a quantity of ZTH to a recipient account with public key $\bar{y}$; (iv) `Lock`, by which a user requests to lock an account to an Ethereum address; (v) `Unlock`, by which a user requests to unlock an account having a public key y. Each user algorithm produces a raw transaction, denoted with tx_{fund}, tx_{burn}, tx_{trans}, tx_{lock} and $\text{tx}_{\text{unlock}}$ respectively, that can be submitted directly to the ZSC by the user. For each of them, the ZSC implements a function that verifies the ZK-proof according to specific statements associated with the transaction. Once the transaction has been processed, the function outputs with success or failure, and in the case of success, the ZSC applies the changes to the state of the accounts.

Front-running and replay attacks. Front-running situations occur when a sort of race condition is not handled. For example, suppose Alice wants to withdraw her balance with a `Burn` transaction. She needs to submit the tx_{burn} transaction with a ZK-proof that the current encrypted balance is well-formed with respect to her public key. However, suppose that a tx_{trans} transaction of another user Bob, who wants to transfer some ZTH to Alice, gets processed before tx_{burn}. In the event of a change in the ciphertext, Alice's transaction will be rejected due to the resulting invalidity of the ZK-proof. A similar situation can also be encountered for `Transfer` transactions. Zether solves the front-running by introducing *pending* state for incoming transfers, *epochs* and *rollover* mechanisms. All incoming transfers for an account are placed

in a pending state and subsequently incorporated into the account at each epoch. The epoch length is chosen so that any transfer or burn transaction can be processed before the account changes state. The rollover mechanism is triggered by one account every time this account sends a message to the ZSC. At this moment, the ZSC ensures that all pending transfers since the last rollover of that account are finalized and funds are available for spending. Zether accounts are decoupled from Ethereum addresses since they have their own public keys. Hence, replay attack protection cannot rely on native Ethereum support. Moreover, an attacker can steal the ZK-proofs and put them into new transactions. To prevent any kind of replay attack, Zether associates a *nonce* with every Zether account. An account forming a new transaction must sign the last nonce which will be incremented when the transaction is processed.

Zether transfer method. Consider a scenario where a sender intends to confidentially transfer a certain amount, denoted with $b^\star$, of ZTH from an account associated with the public key y to a recipient account with the public key $\bar{y}$. To initiate the transfer process, the sender encrypts the desired transfer amount $b^\star$ using both the public key y of the sender's account and the public key $\bar{y}$ of the recipient's account. This encryption generates the ciphertexts $(C = g^{b^\star} \cdot y^r, D)$ and $(\bar{C} = g^{b^\star} \cdot \bar{y}^r, D)$, where both ciphertexts share the same randomness and $D = g^r$. To apply the transfers, the ZSC should add (C^{-1}, D^{-1}) and $(\bar{C}, D)$ with homomorphic group operations to the y's balance and $\bar{y}$'s balance respectively. The outgoing transfer for the $\bar{y}$ is inserted in a pending state for front-running reasons. Let denote with (C_{Ln}, C_{Rn}) the remaining balance of y after deducting the transfer amount $b^\star$ from the actual balance (C_L, C_R). The sender along with the $\mathrm{tx}_{\mathrm{trans}}$ transaction should provide a ZK-proof proving: (i) the knowledge of the secret key sk associated with the public key $y = g^{sk}$; (ii) the knowledge of the randomness r such that $D = g^r$; (iii) that the ciphertexts (C, D) and $(\bar{C}, D)$ encrypt an equal amount b^* and are well-formed; (iv) the balance cannot overdraft: $C_{Ln} = g^{\hat{b}} \cdot C_{Rn}^{sk}$, where $\hat{b}$ is the remaining balance value; (v) that both the transfer amount $b^\star$ and the remaining balance $\hat{b}$ are not negative.

These statements are used to construct the ZK-proof, which will be verified by the ZSC, for the validity of a single transfer. In the next section, we introduce revised statements for the validity of multiple transfers (in a single transaction) to multiple payees.

4. ZeroMT: Multi-transfer payment mechanism

In this section, we present our new payment mechanism in CT settings, in which multiple payees are considered in a single transaction. The aim is to design a Multi-Transfer scheme in which a payer submits a single transaction with the effect of multiple confidential balance updates. For proving the validity of a multi-transfer transaction without revealing transfer amounts and balances, we design a *zero-knowledge proof system* (presented in Section 5) for statements behind the multi-transfer. Here, the goals are to preserve the security and privacy properties of the original Zether payment mechanism and to provide a way of reducing transaction verification costs.

4.1. Differences between ZeroMT and Zether

ZeroMT can be seen as an extension of the Zether transfer method, with the aim of reducing the transaction costs on the public blockchain. For this purpose, ZeroMT introduces an *amortization strategy* where the transaction costs can be reduced by performing multiple transfers in a single transaction equipped with an *aggregate* ZK-proof. Hence, the cost that a private transaction would have for a single transfer is now spread across multiple transfers. We identify such a payment mechanism with the name of *multi-transfer*, a generalization of traditional one-to-one payments into a novel one-to-many payment model. We also remark that Zether cannot be directly used to make multi-transfer transactions. Indeed, the only way to do that with Zether is by carrying out one-to-one transfers separately. This because Zether Σ-Bullets proof system is not designed for multi-transfer ZK-relations, where there can be statements over multiple encrypted and range values for one ZK-proof. This is where our ZeroMT proof system differs substantially: we obtain a ZK-proof for multi-transfer ZK-relations using the aggregation techniques of Bulletproofs and Σ-protocols theory; Zether, on the other hand, does not support any type of aggregation. To better clarify this point, consider the following example for range proofs (a similar observation can be argued for ZK-proofs built with Σ-protocols). Range proofs are used to prove that transfer amounts and the sender's remaining balance are non-negative, e.g., for some value $v \in [0, MAX]$ where MAX is an upper bound of the form 2^n and n is the bit length of v. It turns out that a single range proof for the aggregation of m values is more efficient than having separate range proofs for each of the m values. Indeed, considering two values v_1, v_2 within 64-bit ranges, and group elements and scalars each represented by 32 bytes, then two separate transactions would require a total of $2 \times 672 = 1344$ bytes for range proofs. In contrast, using aggregation, a single transaction has a more efficient range proof, totaling only 736 bytes. While Zether does not natively support one-to-many transactions, ZeroMT is compatible with one-to-one transactions without any modification or incurring additional overhead. This ease of scaling to the traditional payment model is due to the dynamic and modular design of the ZeroMT proof system that underpins the multi-transfer scheme. Therefore, users have the flexibility to decide, depending on the context in which they operate, whether to perform one-to-one or one-to-many transactions. For instance, it is preferable to aggregate transfers for multiple recipients in scenarios such as multi-party off-chain protocols (see Section 8). This results in cost savings for transactions and provides atomicity to the batch of aggregate transfers.

4.2. Multi-transfer zero-knowledge relation

The intuition behind the concept of multi-transfer is shown in Fig. 1 and presented below. Each involved party i owns an ElGamal public key y_i and a balance that is encrypted in the form $b[y_i] = (C_L, C_R) = (g^{b_i} \cdot y_i^r, g^r)$. The actual value of the balance can only be accessed by the owner of the secret key sk_i from which y_i is derived. In the following, we differentiate the payer's public key from those of the payees by denoting them with $\bar{y}$. In order to transfer amounts to each of the designated n payees, the payer should encrypt each amount $\mathbf{a} = (a_1, \ldots, a_n)$ under both her/his public key y and the public keys $\bar{\mathbf{y}} = (\bar{y}_1, \ldots, \bar{y}_n)$ of the payees. As a result, the payer obtains two lists of ciphertexts: $\mathbf{C} = (C_1, \ldots, C_n)$ and $\bar{\mathbf{C}} = (\bar{C}_1, \ldots, \bar{C}_n)$, where for each $i \in [1, n]$, $C_i = g^{a_i} y^r$ is the encryption of the amount a_i under y, $\bar{C}_i = g^{a_i} \bar{y}_i^r$ is the encryption of the amount a_i under $\bar{y}_i$, and $r \xleftarrow{\$} \mathbb{Z}_p^*$. Moreover, each element C_i and $\bar{C}_i$ is coupled with the same random part $D = g^r$. Note that, the sum of all a_i amounts will be debited from the payer, while each a_i amount will be credited to the corresponding $\bar{y}_i$ payee. Let denote $\hat{b}$ the remaining balance of the payer after the deduction of the transfer amounts in $\mathbf{a}$, and with (C_{Ln}, C_{Rn}) the corresponding encryption. In order to prove the validity of many transfers at once, the payer should provide a ZK-proof to prove:

(i) the knowledge of the secret key sk such that $g^{sk} = y$ corresponds to the public key used in the encryption;

(ii) the knowledge of the randomness r such that $g^r = D$ is the random component of the encryption;

(iii) that the balance cannot be overdraft:

$$C_{Ln} = g^{\hat{b}} \cdot C_{Rn}^{sk},$$

where $C_{Ln} = C_L / \prod_{i=1}^n C_i$ and $C_{Rn} = C_R / \prod_{i=1}^n D$;

(iv) that each ciphertext $(C_1 \ldots C_n, D)$ and $(\bar{C}_1 \ldots \bar{C}_n, D)$ encrypts an equal amount a_i at the same index $i \in [1, n]$ and is well-formed;

(v) that each individual amount $(a_1, \ldots, a_n)$ and the balance $\hat{b}$, after deducting these amounts, are not negative.

The conjunction of the above statements forms (informally) the *multi-transfer zero-knowledge relation* that we denote with $\mathcal{R}_{multi}$. In

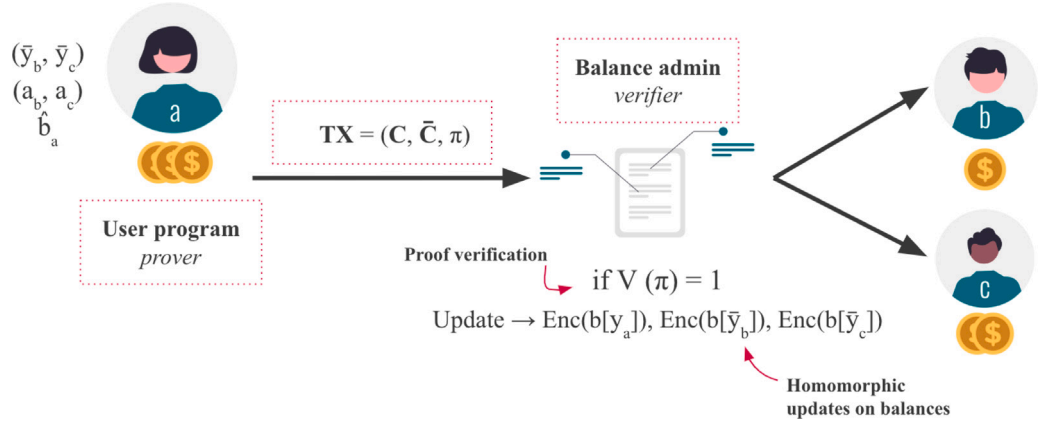


Fig. 1. Multi-transfer concept. The payer (a) runs a user program (acting as a prover) to create a multi-transfer transaction (TX). This transaction allows the payer to transfer money privately to payees (b) and (c). The ZK-proof π is also generated to prove the statements behind the multi-transfer relation. The balance admin (e.g. a smart contract acting as a verifier) verifies the ZK-proof π and updates the balances of all the users with the right amounts.

Section 5, we treat zero-knowledge relations $\mathcal{R}$ formally as a set of instances $\mathbf{x} = (x_1, \ldots, x_n)$, witnesses $\mathbf{w} = (w_1, \ldots, w_m)$ and algebraic statements for which $f(\mathbf{x}, \mathbf{w})$ is true, expressed with the notation:

$$\mathcal{R} : \{(\mathbf{x}; \mathbf{w}) : f(\mathbf{x}, \mathbf{w})\}$$

to specify how a ZK-proof is constructed and highlight that the elements in $\mathbf{x}$ are public while those in $\mathbf{w}$ are only known to the prover.

4.3. Multi-transfer scheme

We now present our Multi-Transfer (MT) scheme. In our design, we focus on transfers and ignore the existence of epochs, as well as pending state and rollover mechanisms. One reason is to simplify the presentation of our scheme, a second reason is that a multi-transfer transaction does not alter the mechanisms to overcome front-running and replay attack situations. On the contrary, our scheme modifies the underlying ZK-proof of Zether, therefore we must consider the *overdraft-safety* and *privacy* security properties. We define a Multi-Transfer scheme as a triple of PPT algorithms $\Pi_{MT} = (\texttt{Setup}, \texttt{MultiTrans}, \texttt{Verify})$:

- $\sigma \leftarrow \texttt{Setup}(1^\lambda)$. `Setup` takes as input the security parameter (in unary representation), internally runs a group-generation algorithm $\mathcal{G}(1^\lambda)$ for the public parameters of the encryption scheme, as well as generates the parameters for the NIZK scheme. All the public parameters are identified under the common string σ, shared by all algorithms but which we consider implicit.
- $\text{tx}_{\text{multi}} := (y, \bar{\mathbf{y}}, \mathbf{C}, \bar{\mathbf{C}}, D, \pi) \leftarrow \texttt{MultiTrans}(y, \bar{\mathbf{y}}, sk, \mathbf{a}, \hat{b})$. `MultiTrans` generates a multi-transfer transaction, given as inputs the public keys (of the payer) y and (of the destination accounts) $\bar{\mathbf{y}}$, the secret key for which $g^{sk} = y$, the transfer amounts $\mathbf{a}$ and the balance that will be deducted equal to $\hat{b}$.
- $0/1 \leftarrow \texttt{Verify}(\text{tx}_{\text{multi}})$. `Verify` verifies the multi-transfer transaction tx_{multi}. In particular, it verifies the ZK-proof π against the multi-transfer relation. If π is valid, `Verify` sets $b[y] = b[y] \circ (C_{tot}^{-1}, D^{-1})$ and $b[\bar{y}_i] = b[\bar{y}_i] \circ (\bar{C}_i, D)$ for each $i \in [1, n]$, where $b[y]$ is the payer current balance and $b[\bar{y}_i]$ the balance of the ith recipient.

The Π_{MT} scheme comes with an oracle smart contract $\mathcal{O}_{SC}$ which maintains a table of accounts $b[\,]$ representing the global state, and also calls `Verify` on every received tx_{multi} transaction. The overall scheme is summarized below.

Multi-Transfer payment scheme: Π_{MT}

Setup

INPUTS: security parameter λ

1. $\sigma_g \leftarrow \mathcal{G}(1^\lambda)$
2. $\sigma_{nizk} \leftarrow \text{Setup}_{nizk}(1^\lambda)$
3. Let $\sigma = (\sigma_g, \sigma_{nizk})$
4. Initialize a global state $b[\,] : \mathbb{G} \rightarrow \mathbb{G}^2$
5. Initialize the oracle $\mathcal{O}_{SC}$ with parameters $\sigma, b[\,], \text{MAX}$

MultiTrans

INPUTS:

- public key of the sender y
- public keys of the receivers $\bar{\mathbf{y}} = (\bar{y}_1, ..., \bar{y}_n)$
- private key of the sender sk
- transfer amounts $\mathbf{a} = (a_1, ..., a_n)$
- sender remaining balance $\hat{b}$

OUTPUTS: tx_{multi}

1. Let $(C_L, C_R) = b[y]$
2. Set $r \xleftarrow{\$} \mathbb{Z}_p$
3. Set $C_i = g^{a_i} y^r \ \forall i \in [n]$
4. Set $\bar{C}_i = g^{a_i} \bar{y}^r \ \forall i \in [n]$
5. Set $D = g^r$
6. Set $w = (\mathbf{a}, \hat{b}, sk, r)$
7. Let $\mathbf{C} = (C_1, ..., C_n)$
8. Let $\bar{\mathbf{C}} = (\bar{C}_1, ..., \bar{C}_n)$
9. $\pi = \textbf{Prove}_{\text{nizk}}(\mathcal{R}_{multi}[C_L, C_R, y, \bar{\mathbf{y}}, \mathbf{C}, \bar{\mathbf{C}}, g, D; w])$

Verify

INPUTS: tx_{multi}

OUTPUTS: 0/1 (rejects or accepts)

1. Let $(C_L, C_R) = b[y]$

2. Set $C_{tot} = \prod_{i=1}^{n} C_i$

3. Require: $\mathbf{Verify}_{\text{nizk}}(\mathcal{R}_{multi}[C_L, C_R, y, \bar{\mathbf{y}}, \mathbf{C}, \bar{\mathbf{C}}, g, D], \pi) = 1$; otherwise: return 0

4. $b[y] = b[y] \circ (C_{tot}^{-1}, D^{-1})$

5. For each $i \in [1, n]$ set $b[\bar{y}_i] = b[\bar{y}_i] \circ (\bar{C}_i, D)$

6. return 1

Here, the ZK-proof π appears in a NIZK version through the triple $(\text{Setup}_{nizk}, \text{Prove}_{nizk}, \text{Verify}_{nizk})$, and the `MultiTrans` algorithm assumes the role of the prover, while the `Verify` algorithm takes the role of the verifier. The NIZK argument satisfies the relation $\mathcal{R}_{multi}$ defined in Section 4, where $(C_L, C_R, y, \bar{\mathbf{y}}, \mathbf{C}, \bar{\mathbf{C}}, g, D)$ are the public instances and $(\mathbf{a}, \hat{b}, sk, r)$ the witnesses. Moreover, $\mathcal{R}_{multi}$ expresses the range proof statements, which we use to prove that each amount in $\mathbf{a}$ and the balance $\hat{b}$ are within the range $[0, \text{MAX}]$, where MAX is an upper limit fixed in `Setup`.

4.4. Security requirements

We now show how the Multi-Transfer scheme has no effect on the security of Zether. In what follows, we consider $\mathcal{C}$ the challenger, $\mathcal{A}$ the adversary and $\mathcal{O}_{SC}$ the oracle smart contract, as entities involved in security experiments. The capabilities of any $\mathcal{A}$ in our experiments are listed below: (i) it has the view of all transactions to the oracle $\mathcal{O}_{SC}$ and state changes performed by the oracle; (ii) it can instruct $\mathcal{C}$ to submit transactions with certain inputs and then send the transactions to $\mathcal{O}_{SC}$; (iii) it can submit direct transactions to $\mathcal{O}_{SC}$; (iv) it does not have the secret keys of corrupted parties, but it can specify the corresponding public keys; (v) it can query the oracle $\mathcal{O}_{SC}$ to update the state of the accounts and also to withdraw from an account.

Overdraft-safety. The overdraft-safety security definition states that an adversary $\mathcal{A}$ cannot withdraw more money than it has. We consider the case of `MultiTrans`, in which $\mathcal{A}$ creates the transactions tx_{multi} and subsequently tries to withdraw from corrupted accounts. Let $\mathbf{y_i} = (y_1, \ldots, y_n)$ the set of public keys involved in the transactions, containing both not corrupted and corrupted keys (the latter indicated with $\mathbb{C} \subset \mathbb{S}$). Let $(C_{L,i}, C_{R,i})$ the state of each account involved, before tx_{multi} is processed. Let the ciphertexts of the transfer amounts be $\mathbf{C} = (C_1 ... C_n, D)$ and π the ZK-proof for tx_{multi}. First, $\mathcal{A}$ queries $\mathcal{O}_{SC}$ to initiate withdrawals from these accounts, thereby obtaining quantities $b_1, \ldots, b_n$. Then, if the tx_{multi} transaction is handled by `Verify` there must be ciphertexts (C_j, D) that encrypt an amount a under $y_j \in \mathbb{C}$ and ciphertexts (C_k, D) that encrypt an amount a under $y_k \in \mathbb{S} \setminus \mathbb{C}$, for some $j \neq k$. For the state of corrupted accounts, it must hold that $C_{L,i} / \prod C_j = g^{\hat{b}} \cdot (C_{R,i} / \prod D)^{sk}$ for some j and a non-negative quantity $\hat{b}$ (because of the soundness property of the ZK-proof). Now, when $\mathcal{A}$ makes the withdraw query on each y_i again, the y_j account state can be $(C_{L,i}, C_{R,i}) \circ ((\prod C_j)^{-1}, D^{-1})$ or $(C_{L,i}, C_{R,i})$, based on whether state changes have been applied. By Zether's design, this withdraw query will not take effect in the same epoch of the transfer query (due to the nonce of the transaction). Thus, in the next epoch the state changes of tx_{multi} will be applied, and the withdraw will return $b_j - a$. Therefore, we can see that the soundness of the ZK-proof together with the nonce mechanisms are used to enforce the overdraft-safety. This indicates that in the event of an adversary successfully compromising the soundness of the proof system, the adversary will gain a non-negligible advantage in the overdraft-safety experiment described above. We can better formulate this event from the definition of soundness 2.2 in the following:

Definition 4.1 (*Overdraft-Safety of Multi-Transfer Scheme*)**.** We say that Π_{MT} has overdraft-safety security if there exists a polynomial-time extractor $\mathcal{E}$ for which, for every PPT adversary $\mathcal{A}$, the advantage of breaking the soundness of the ZK-proof is negligible.

In particular, $\mathcal{A}$ is given as input the common random string σ of the NIZK scheme, and outputs the public parameters x and a pair of accepting transcripts whose challenges are distinct. The extractor $\mathcal{E}$ is given the public parameters and the transcripts, and outputs a witness w. This implies the adversary succeeds by convincing `Verify` of false statements.

Privacy. The privacy security definition states that no information is leaked to an adversary $\mathcal{A}$ about the transactions of honest users. A Multi-Transfer transaction tx_{multi} consists of the sender's public key y, the list of recipients' public keys $\bar{\mathbf{y}}$, two lists of ElGamal ciphertexts $\mathbf{C}$ and $\bar{\mathbf{C}}$, a blinding value D and a ZK-proof π. Now, we modify the Zether's privacy experiment in the case of Multi-Transfer. In such experiment, every adversary $\mathcal{A}$ sends two *publicly consistent* transfer queries to the challenger $\mathcal{C}$. Two queries are consistent if they have the same public parameters and are jointly consistent with the $\mathcal{A}$'s view. In the latter, both transfer queries have the same set of public keys and if one of the recipients is corrupted in this set, both queries have the same recipient and the same amount. If either of these consistencies is not met, the experiment aborts. Otherwise, $\mathcal{C}$ selects a uniformly random bit $b \leftarrow \{0, 1\}$ and executes the $(1 - b)$th query. Finally, $\mathcal{A}$ outputs a bit b' as a guess for b. The adversary succeeds in the privacy experiment if $b' = b$ with an advantage non-negligibly better then $1/2$. Considering that every ciphertext in $\mathbf{C}$ and $\bar{\mathbf{C}}$ is indistinguishable from the encryption of random messages and that y and every element in $\bar{\mathbf{y}}$ is indistinguishable from random element under DDH assumption, the advantage of $\mathcal{A}$ is defined by the advantage it would have in breaking the zero-knowledge property of the proof π. Hence, we give the security definition of privacy with respect to the zero-knowledge Definition 2.3 in the following:

Definition 4.2 (*Privacy of Multi-Transfer Scheme*)**.** We say that Π_{MT} has privacy security (or equivalently it is *private*) if there exists a PPT simulator $\mathcal{S}$ for which, for every PPT adversary $\mathcal{A}$, the advantage to distinguish between the simulated transcript and the real transcript of the honest proof is at most negligibly better then $1/2$.

In particular, $\mathcal{A}$ is given as input the common random string σ of the NIZK scheme, and outputs an instance-witness pair (x, w) such that $(\sigma, x, w) \in \mathcal{R}$, as well as the verifier's internal randomness. Then, $\mathcal{C}$ outputs a real or simulated transcript based on the tossing of a random bit $b \leftarrow \{0, 1\}$. The adversary $\mathcal{A}$ is given the transcript and outputs b' as a guess for b. The output of the experiment is 1 if the two bits are equal. Since the security of the NIZK scheme inherits that of the interactive version, in constructing the security proofs we reduce the adversary $\mathcal{A}$ into an adversary attacking the soundness and the zero-knowledge property of the interactive scheme, as discussed in Appendix C.

4.5. Further attacks

In the previous section, we have seen that our multi-transfer scheme is resistant to over-spending attacks w.r.t. the overdraft-safety property and to eavesdropping attacks w.r.t. the privacy property of our scheme. Here, we discuss some other well-known attacks in blockchain contexts.

Replay attacks and double-spending. Replay attacks occur when a malicious user attempts to execute a new transaction using a stolen ZK-proof. In the multi-transfer scheme, where ZK-proofs are handled in the NIZK version, an attacker could obtain a ZK-proof from honest users and have the new transaction processed ahead of them. This could result in users losing funds or even spending the same balance twice (double-spending attack). To prevent these attacks we can adopt *nonces* as in traditional account-based blockchains, which are still valid in the case of multi-transfer transactions. Indeed, the last nonce associated with an account can be provided with the transaction tx_{multi}. Then, the nonce value will be validated against such account and incremented when the transaction is processed. Double-spending attacks are mitigated not only by nonces but also by adopting pending transfers and

rollover mechanisms as in Zether. With these mechanisms, two distinct transactions tx_{multi} have effect only with respect to their balance state. In other words, a tx_{multi} does not alter the state of an account until the state changes from the previous tx_{multi} have been applied. Therefore, the corresponding ZK-proof is verified against the new state. If for some reason the nonce or the ZK-proof is invalid, none of the transfers within the tx_{multi} will be executed.

Scalability attacks. Different types of attacks that can be devised concern those that slow down or compromise the availability of blockchain services. For instance, a Denial-of-Service (DoS) attack can prevent legitimate user requests, even temporarily exhausting the resources of blockchain nodes [37]. In general, these kind of attacks are usually done when the cost of mounting the attack is very low. To mitigate such attacks, countermeasures such as increasing transaction fees and limiting transaction sizes have been proposed. In our case, we point out that the multi-transfer scheme cannot be useful for attacks of this type when, for example, the scheme is deployed in a smart contract. Indeed, a transaction with a high volume of transfers could exceed the gas limit or incur high fees, thus such constraints prevent or discourage scalability attacks.

5. Multi-transfer zero-knowledge proof system

In this section, we outline the zero-knowledge interactive proof system for our Multi-Transfer scheme. The system leverages the *aggregate Range Proof* (RP) from Bulletproofs theory [13] in combination with several Σ-Protocols, exploiting the adaptations of Σ-Bullets [12]. With the aggregate RP, we generate one proof for $m = n + 1$ range values, where n are the values of the transfer amounts and one value is the balance after the transfer. With several Σ-Protocols we prove the statements concerning the DLOG relations over the two lists of ciphertexts $\mathbf{C}$ and $\bar{\mathbf{C}}$. The adaptations in Σ-Bullets [12] suggest how to work with the ElGamal encryption as a substitute for Pedersen commitments in the RP system. The Inner Product Argument (IPA) protocol of Bulletproofs comes with logarithmic-sized proofs, this property along with the trustless is inherited in our ZK-proof. Moreover, a logarithmic-sized proof also helps in aggregation efficiency, given that the size of the relations over our proof grows with the number of transfers. We end up with an interactive ZK-proof that is generalized for many transfers, suitable for homomorphic cryptosystems like Zether and is SHVZK. The section is organized as follows: we review the Bulletproofs notations and theory; we show how we use the aggregate RP and the adaptations for the ElGamal encryption; finally, we outline the Σ-protocols for the rest of our ZK relations.

5.1. Bulletproofs review

In the Table 1, we summarize the notations we use in our proof system from the original Bulletproofs paper (for a complete description see section 2.3 of [13]):

Bulletproofs is a zero-knowledge argument system with which a prover can convince a verifier that a value v resides in a range from zero to 2^{n-1}, with n the range domain, without revealing the value v. In the final stage of the RP protocol, an IPA is employed on pre-defined vectors $\mathbf{l}$ and $\mathbf{r}$ to prove that their inner product $\langle \mathbf{l}, \mathbf{r} \rangle$ equals a specific $\hat{t}$. From this, it is possible to conclude that $v \in [0, 2^n - 1]$. Prior steps to the IPA protocol proceed as follows. From bit decomposition of v, the prover generates vectors $\mathbf{a}_L, \mathbf{a}_R$, for which it holds that $\langle \mathbf{2}^n, \mathbf{a}_L \rangle = v$ and $\mathbf{a}_L - \mathbf{1}^n = \mathbf{a}_R$, blinding vectors $\mathbf{s}_L, \mathbf{s}_R$, and commitments $A = commit(\mathbf{a}_L, \mathbf{a}_R)$ and $S = commit(\mathbf{s}_L, \mathbf{s}_R)$. The prover then sends A, S to the verifier. After the verifier sends y, z challenges, the prover defines the polynomial $t(X) = \sum_{i=0}^{d} t_i X^i$, where $d = 2$ is the degree, by taking the inner-product of the pre-defined vector polynomials $l(X), r(X)$. From $t(X)$, the prover generates commitments $T_i = commit(t_i)$ for $i \in [1, 2]$, and sends them to the verifier (note that the zero coefficient is not committed). Upon receiving x challenge from the verifier, the prover evaluates the polynomial $t(x) = \hat{t}$ at point x. Given the public parameters (g, h, V) for this argument, where $V = g^v h^\gamma$ commits to the range value v, the verifier can compute the zero coefficient by using V and the challenges. Finally, the prover sends $\hat{t}$ and $l(x) = \mathbf{l}, r(x) = \mathbf{r}$ evaluations along with other blinding factors (τ_x, μ). With these, the verifier checks the public V, the commitments A, S and that the inner-product $\langle \mathbf{l}, \mathbf{r} \rangle = \hat{t}$ is valid. The messages exchanged between prover and verifier in the Bulletproofs protocol are shown in the Table 2. However, in the last step, the prover sends the vectors $\mathbf{l}$ and $\mathbf{r}$, resulting in a linear communication cost. With these two vectors becoming witnesses and using the IPA protocol, the RP can get a logarithmic communication cost in n (bits of the range).

Table 1
Bulletproofs notations.

$\mathbb{Z}_p$	Ring of integers modulo prime number p.
$\mathbb{G}$	Cyclic group of prime order p.
g and h	Two distinct generators of the group $\mathbb{G}$.
$V = commit(v) = g^v \cdot h^\gamma$	Pedersen commitment for $v \in \mathbb{Z}_p$ using a random blinding factor γ.
$\mathbf{v} = (v_1, \dots, v_n) \in \mathbb{Z}_p^n$	Vector of dimension n with elements in $\mathbb{Z}_p$.
$\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = \sum_{i=1}^{n} v_{1,i} \cdot v_{2,i} \in \mathbb{Z}_p$	Inner-product between vectors of dimension n.
$\mathbf{v}_1 \circ \mathbf{v}_2 = (v_{1,1} \cdot v_{2,1}, \dots, v_{1,n} \cdot v_{2,n}) \in \mathbb{Z}_p^n$	Hadamard-product between vectors of size n.
$V = \mathbf{g}^{\mathbf{v}} = \prod_{i=1}^{n} g_i^{v_i}$	Pedersen vector commitment to $\mathbf{v} \in \mathbb{Z}_p^n$, with $\mathbf{g} = (g_1, \dots, g_n) \in \mathbb{G}^n$ vector of generators.
$\mathbf{s}_{[:k]} = (s_1, \dots, s_k) \in \mathbb{F}^k$ $\mathbf{s}_{[k:]} = (s_{k+1}, \dots, s_n) \in \mathbb{F}^{n-k}$	Vector slice operations.
$p(X) \in \mathbb{Z}_p^n[X]$	Vector polynomial $p(X) = \sum_{i=0}^{d} \mathbf{p_i} X^i$, with d the polynomial degree and $\mathbf{p_i} \in \mathbb{Z}_p^n$.

Table 2
Bulletproofs protocol messages.

Prover	Verifier
$A, S \rightarrow$	
	$\leftarrow y, z$
$T_1, T_2 \rightarrow$	
	$\leftarrow x$
$\tau_x, \mu, \hat{t}, \mathbf{l}, \mathbf{r} \rightarrow$	
	verify

5.2. Integrating aggregate RP in our proof

We use an *aggregate RP* to create one proof for m range values. In particular, we construct a range proof for the $a_1, \dots, a_j$ transfer amounts for some integer j and the balance value $\hat{b}$, having in total $m = j + 1$ range values. Based on the binary decomposition of $\hat{b}$ and each $a_1, \dots, a_j$, the prover begins by creating the two vectors $\mathbf{a_L}$ and $\mathbf{a_R}$ such that:

$$\langle \mathbf{2}^n, \mathbf{a}_{L[:n]} \rangle = \hat{b} \text{ and } \langle \mathbf{2}^n, \mathbf{a}_{L[(k-1)\cdot n : k \cdot n]} \rangle = a_{k-1}, \ \forall k \in [2, m]$$

and

$$\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^{m \cdot n} \in \{0, -1\}^{m \cdot n}$$

where n is the bit length of the range values.

Hence, the prover creates $A = \mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{a}_R} h^\alpha \in \mathbb{G}$ and $S = \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R} h^\rho \in \mathbb{G}$ commitments, with $\mathbf{g}$ and $\mathbf{h}$ vectors of generators, and the term $h \in \mathbb{G}$ used as a blinding generator with α and ρ random values. Moreover, the prover modifies the vectors of blinding terms $\mathbf{s}_L$ and $\mathbf{s}_R$ such that they stay in $\mathbb{Z}_p^{m \cdot n}$. The range proof proceeds as described in Section 5.1; below we outline the cascading changes derived from the *aggregate RP*.

The vector polynomials $l(X)$ and $r(X)$, which are used to derive $t(X)$, are modified to stay in $\mathbb{Z}_p^{m \cdot n}[X]$. As before, the prover commits to $t(X)$ generating $T_{1,2} = g^{t_{1,2}} h^{\tau_{1,2}}$ commitments, where $\tau_1, \tau_2 \in \mathbb{Z}_p$ are

uniformly random values. After the verifier sends its challenge x, the prover evaluates at x the polynomial $t(x) = \hat{t}$, and generates the blinding factors τ_x for $\hat{t}$ and μ for A, S commitments.

From this point, instead of verifying the public commitment V as described in Section 5.1, we adopt a similar strategy of Σ-Bullets [12]. This is because the verifier's equality involves an additive homomorphism, which is not applicable when ElGamal ciphertexts are not encrypted with the same key. The strategy suggests proving the opening of the polynomial commitment, which is the witness τ_x in the equality involving $T_{1,2}$ commitments. We can do this with a Σ-Protocol that satisfies the following zero-knowledge relation:

$$\mathcal{R}_{AggRP} : \{(g, h \in \mathbb{G},\ y, z, \psi(y,z), \hat{t} \in \mathbb{Z}_p;\ \hat{b} \in \mathbb{Z}_p, \mathbf{a} \in \mathbb{Z}_p^{m-1}, \tau_x \in \mathbb{Z}_p) : g^{\hat{t}-\hat{b}\cdot z^2 - \sum_{i=1}^{m-1} a_i \cdot z^{i+2} - \psi(y,z)} h^{\tau_x} = T_{1,2}\} \tag{1}$$

where y and z are the challenges of the verifier discussed in Section 5.1, and $\psi(y,z)$ is a function from the *aggregate RP* dependent on these challenges and can be calculated independently by the verifier.

The prover and verifier engage an argument of knowledge for relation (1); we denote this argument with Σ_{AggRP}. Therefore, the prover generates the uniformly random scalars k_{ba} and k_τ and computes $A_t = g^{-k_{ba}} h^{k_\tau}$. Afterwards, the prover transmits A_t, μ and $\hat{t}$ to the verifier. Once the verifier's challenge c is received, the prover calculates

$$s_{ba} = k_{ba} + c \cdot (\hat{b} \cdot z^2 + \sum_{i=1}^{m-1} a_i \cdot z^{2+i}) \in \mathbb{Z}_p$$

and

$$s_\tau = k_\tau + c \cdot \tau_x \in \mathbb{Z}_p,$$

which the prover then sends to the verifier.

These values are then used by the verifier in the proof check given by the expression:

$$g^{(\hat{t}-\psi(y,z))\cdot c - s_{ba}} \cdot h^{s_\tau} \stackrel{?}{=} A_t \cdot (T_1^x \cdot T_2^{x^2})^c.$$

The full aggregate range proof between the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ is shown in the interactive protocol below; the protocol is given by the conjunction *aggregate RP* $\wedge$ Σ_{AggRP}.

Protocol 1: *Aggregate RP* $\wedge$ Σ_{AggRP} interactive protocol

$\mathcal{P}$ computes
  $\mathbf{a}_L \in \{0,1\}^{m\cdot n}$ s.t.
    $\langle \mathbf{2}^n, \mathbf{a}_{L[:n]} \rangle = \hat{b}$ and
    for all $k \in [2, m]$:
      $\langle \mathbf{2}^n, \mathbf{a}_{L[(k-1)\cdot n : k\cdot n]} \rangle = a_{k-1}$
    end for
  $\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^{m\cdot n}$
  $\alpha, \rho \xleftarrow{\$} \mathbb{Z}_p$
  $A \leftarrow h^\alpha \mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{a}_R}$
  $\mathbf{s}_L, \mathbf{s}_R \xleftarrow{\$} \mathbb{Z}_p^{m\cdot n}$
  $S \leftarrow h^\rho \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R}$
end $\mathcal{P}$
$\mathcal{P} \to \mathcal{V} : A, S$
$\mathcal{V} \to \mathcal{P} : y, z \xleftarrow{\$} \mathbb{Z}_p^*$
$\mathcal{P}$ computes
  $l(X) := l_l + l_r \cdot X \in \mathbb{Z}_p^{m\cdot n}[X]$
  $r(X) := r_l + r_r \cdot X \in \mathbb{Z}_p^{m\cdot n}[X]$
  $l_l \leftarrow \mathbf{a}_L - z\mathbf{1}^{n\cdot m}$
  $r_l \leftarrow \mathbf{y}^{n\cdot m} \circ (\mathbf{a}_R + z\mathbf{1}^{n\cdot m}) + \sum_{k=1}^{m} z^{k+1} \cdot (\mathbf{0}^{n\cdot(k-1)} || \mathbf{2}^n || \mathbf{0}^{n\cdot(m-k)})$
  $r_r \leftarrow \mathbf{y}^{n\cdot m} \circ \mathbf{s}_R$
  $t(X) := \sum_{i=0}^{d} t_i X^i \in \mathbb{Z}_p[X]$, where $d = 2$ and $t(X) = \langle l(X), r(X) \rangle$
  $\tau_1, \tau_2 \xleftarrow{\$} \mathbb{Z}_p$
  $T_{1,2} = g^{t_{1,2}} h^{\tau_{1,2}}$
end $\mathcal{P}$
$\mathcal{P} \to \mathcal{V} : T_1, T_2$
$\mathcal{V} \to \mathcal{P} : x \xleftarrow{\$} \mathbb{Z}_p^*$
$\mathcal{P}$ computes
  $\mathbf{l} = l(x) \in \mathbb{Z}_p^{m\cdot n}$
  $\mathbf{r} = r(x) \in \mathbb{Z}_p^{m\cdot n}$
  $\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle \in \mathbb{Z}_p$
  $\tau_x = \tau_1 \cdot x + \tau_2 \cdot x^2 \in \mathbb{Z}_p$
  $\mu = \alpha + \rho \cdot x \in \mathbb{Z}_p$
  $k_{ba}, k_\tau \xleftarrow{\$} \mathbb{Z}_p$ ▷ begin Σ-Protocol
  $A_t = g^{-k_{ba}} h^{k_\tau}$
end $\mathcal{P}$
$\mathcal{P} \to \mathcal{V} : \hat{t}, \mu, A_t$
$\mathcal{V} \to \mathcal{P} : c \xleftarrow{\$} \mathbb{Z}_p$
$\mathcal{P}$ computes
  $s_{ba} = c \cdot (\hat{b} \cdot z^2 + \sum_{i=1}^{n} a_i \cdot z^{2+i}) + k_{ba}$
  $s_\tau = c \cdot \tau_x + k_\tau$
end $\mathcal{P}$
$\mathcal{P} \to \mathcal{V} : s_{ba}, s_\tau$
$\mathcal{V}$ requires
  $\psi(y,z) = (z - z^2) \cdot \langle \mathbf{1}^{m\cdot n}, \mathbf{y}^{m\cdot n} \rangle - \sum_{j=1}^{m} z^{2+j} \cdot \langle \mathbf{1}^n, \mathbf{2}^n \rangle$
  $g^{(\hat{t}-\psi(y,z))\cdot c - s_{ba}} \cdot h^{s_\tau} \stackrel{?}{=} A_t \cdot (T_1^x \cdot T_2^{x^2})^c$
end $\mathcal{V}$

Theorem 5.1. *Aggregate RP* $\wedge$ Σ_{AggRP} *is a public-coin (perfectly) special honest-verifier argument of knowledge of the relation* $\mathcal{R}_{AggRP}$ (1).

This proof appears in Appendix A.

To conclude the range proof, it remains to verify that $\langle \mathbf{l}, \mathbf{r} \rangle = \hat{t}$. Therefore, our aggregate range proof proceeds with the inner product argument protocol, the full description of which is referred to [13]. As specified in the paper, the following modified inputs are now given to the IPA between $\mathcal{P}$ and $\mathcal{V}$: $(\mathbf{g}, \mathbf{h}, C, \hat{t}; \mathbf{l}, \mathbf{r})$, where

$$\mathbf{h} = (h_1, h_2^{y^{-1}}, h_3^{y^{-2}}, \ldots, h_{m\cdot n}^{y^{-m\cdot n+1}})$$

and

$$C = AS^x \cdot \mathbf{g}^{-z} \cdot h^{-\mu} \cdot \mathbf{h}^{\mathbf{y}^{m\cdot n}\cdot z} \cdot \prod_{j=1}^{m} \mathbf{h}_{[(j-1)\cdot n : j\cdot n]}^{\mathbf{2}^n \cdot z^{1+j}}.$$

Moreover, we use the *multi-exponentiation* technique on the verifier for further optimization.

5.3. Σ-protocol for multiple transfers

In this section, we outline the remaining part of our ZK-proof for the multi-transfer transaction presented in Section 4. In particular, we design a Σ-Protocol satisfying the zero-knowledge relation:

$$\mathcal{R}_{MultiTrans} : \{(C_L, C_R, D, g, y \in \mathbb{G},\ \mathbf{C}, \bar{\mathbf{C}}, \bar{\mathbf{y}} \in \mathbb{G}^{m-1};\ \mathbf{a} \in \mathbb{Z}_p^{m-1},\ \hat{b}, sk, r \in \mathbb{Z}_p) : (C_i = g^{a_i} y^r \wedge \bar{C}_i = g^{a_i} \bar{y}_i^r \wedge D = g^r)_{i=1}^{m-1} \wedge C_L \cdot (\prod_{i=1}^{m-1} C_i)^{-1} = g^{\hat{b}} \cdot (C_R \cdot (\prod_{i=1}^{m-1} D)^{-1})^{sk} \wedge y = g^{sk}\} \quad (2)$$

The relation outlines the algebraic statements over the two lists $\mathbf{C}$ and $\bar{\mathbf{C}}$ of homomorphic encryptions and the statements relating to the knowledge of secrets. In summary, the prover proves (i) that each element in both the $\mathbf{C}$ and $\bar{\mathbf{C}}$ lists is a correct encryption corresponding to the ith element in the list $\mathbf{a}$ of amounts to transfer, (ii) that the remaining balance (C_L, C_R) cannot overdraft, i.e., corresponds to a non-negative value $\hat{b}$, after the deduction of the transfer amounts, (iii) the knowledge of the secret key sk associated with the public key y and the knowledge of the randomness r used in the encryption.

Hence, the prover and verifier engage in an interactive proof, satisfying the relation $\mathcal{R}_{MultiTrans}$ (2), shown in the following protocol.

Protocol 2: $\Sigma_{MultiTrans}$ interactive protocol

1: $\mathcal{P}$ computes:
2: $k_{sk}, k_r, k_{ba} \xleftarrow{\$} \mathbb{Z}_p$
3: $A_y = g^{k_{sk}} \in \mathbb{G}$
4: $A_D = g^{k_r} \in \mathbb{G}$
5: $A_{ba} = ((C_R / \prod_{i=1}^{m-1} D)^{z^2} \cdot \prod_{i=1}^{m-1} D^{z^{i+2}})^{k_{sk}} \cdot g^{k_{ba}} \in \mathbb{G}$
6: $A_{\bar{y}} = \prod_{i=1}^{m-1} (y \cdot \bar{y}_i^{-1})^{k_r} \in \mathbb{G}$
7: **end** $\mathcal{P}$
8: $\mathcal{P} \rightarrow \mathcal{V} : A_y, A_D, A_{ba}, A_{\bar{y}}$
9: $\mathcal{V} \rightarrow \mathcal{P} : c \xleftarrow{\$} \mathbb{Z}_p$
10: $\mathcal{P}$ computes:
11: $s_{sk} = c \cdot sk + k_{sk} \in \mathbb{Z}_p$
12: $s_r = c \cdot r + k_r \in \mathbb{Z}_p$
13: $s_{ba} = c \cdot (\hat{b} z^2 + \sum_{i=1}^{m-1} (a_i z^{i+2})) + k_{ba} \in \mathbb{Z}_p$
14: **end** $\mathcal{P}$
15: $\mathcal{P} \rightarrow \mathcal{V} : s_{ba}, s_{sk}, s_r$
16: $\mathcal{V}$ requires:
17: $g^{s_{sk}} \stackrel{?}{=} A_y y^c$
18: $g^{s_r} \stackrel{?}{=} A_D D^c$
19: $g^{s_{ba}} ((\frac{C_R}{\prod_{i=1}^{m-1} D})^{z^2} \cdot \prod_{i=1}^{m-1} D^{z^{2+i}})^{s_{sk}} \stackrel{?}{=} A_{ba} ((\frac{C_L}{\prod_{i=1}^{m-1} C_i})^{z^2} \cdot \prod_{i=1}^{m-1} C_i^{z^{2+i}})^c$
20: $\prod_{i=1}^{m-1} (y \cdot \bar{y}_i^{-1})^{s_r} \stackrel{?}{=} A_{\bar{y}} \cdot (\prod_{i=1}^{m-1} C_i / \bar{C}_i)^c$
21: **end** $\mathcal{V}$

Theorem 5.2. *$\Sigma_{MultiTrans}$ is a public-coin (perfectly) special honest-verifier argument of knowledge of the relation $\mathcal{R}_{MultiTrans}$ (2).*

This proof appears in Appendix B.

6. Related work and comparison

MimbleWimble [7] proposes improvements to the original UTXO-based Bitcoin platform, such as smaller transaction history and a degree of secrecy for transactions. Instead of directly inserting values into a transaction, Pedersen commitments are built upon multiple blinding factors. A range proof system is theorized in order to ensure that all the amounts that are exchanged during transactions are non-negative. However, one range proof is provided for each spending coin commitment, and aggregation of range proofs is left as an open problem. Moreover, while it is possible to spend many coins in one transaction, it is not clear how to make a multi-transfer transaction.

Zerocash [8] enhances the privacy of Bitcoin's UTXO model. In Zerocash the value of a coin is mixed together with the address of its owner using nested commitments. The transaction for spending the coins provides a zk-SNARK proof, built on an arithmetic circuit for a specific NP statement. Despite the constant-size proofs which confer succinctness, Zerocash is not completely trustless. Furthermore, a multi-transfer transaction in Zerocash requires an increase in the complexity of the circuit, resulting in decreased performances for the generation and verification of the ZK-proofs.

Lelantus [9] offers a private payment mechanism based on the UTXO model. Coin values need to be minted in *double-blinded Pedersen commitments* in order to be spent. The spend transaction includes Σ proofs to prove that the user knows all the private data related to the commitments. A *JoinSplit* transaction enables a user to merge, split or redeem coins and requires the generation of range proofs to prove that the output commitments are associated with non-negative values. Despite ZK-proof batching techniques and the possibility to spend multiple coins, a transaction must be provided with separate proofs (Σ and range proofs) for each coin; this results in significant overhead when funds to be spent increase.

Monero [10] is a private cryptocurrency based on the RingCT protocol [38], where the origin, destination and amounts of a UTXO transaction are kept hidden. In particular, the sender combines her/his unspent coins with a set of addresses (not related to each other) and their unspent outputs. Each of the output values in the transaction is concealed with a *Pedersen commitment*. Then, a *commitment to zero* is formed by subtracting the total input commitments from the total output commitments. This commitment is used by the sender within a ring structure to prove ownership of the coins she/he intends to spend. At the base, there is the MLSAG [38] ring signature scheme, used to make the actual signer indistinguishable in a set of public keys. Finally, range proofs are used to prove that the outputs of a transaction are in the range of admissible values. Monero suffers from the continuous growth of UTXO set size and weak anonymity due to the reuse of addresses.

Quisquis [11] aims at solving the drawbacks of Zerocash and Monero. In particular, a compact UTXO set is provided by replacing the public keys of the sender with those of the recipients at each transaction. Moreover, *updatable public keys* (UPKs) are used to update the public keys while the secret key is left unchanged. With this method, an address is only stored twice in the blockchain, when it is consumed as an input or it is generated as an output of a transaction. Confidential transfers are made with the use of commitments with which balances and transfer amounts are hidden. Then, those commitments are employed in constructing a zero-knowledge proof of knowledge, relying on DLOG assumptions and using groups in which the DDH problem is intractable. Moreover, Quisquis does not rely on a trusted setup and has great support for multi-transfer transactions. However, Quisquis lacks prevention of front-running situations, which could cause honest transactions to fail.

Zether [12] is the first privacy-preserving payment system based on the account model. Confidential transactions are accompanied with *trustless* thanks to the Σ-Bullets proof system. This allows proving that transfer amounts and balances are non-negatives and that the encryption is correct. Zether also provides a smart contract protocol where the account balances are kept encrypted and homomorphically updated at each epoch. The Zether smart contract also provides user-accessible functions for depositing, transferring, and withdrawing funds to and from accounts and guarantees replay attacks and front-running protection. Basic Zether does not have anonymity, i.e. the sender and receiver are linkable in a transaction, and the multi-transfer transaction is discussed only informally. Moreover, Zether does not natively

Table 3
Comparison of CT solutions. HE denotes homomorphic encryption. SC means whether the privacy solution supports smart contracts or not. Anonymity means whether the unlinkability between sender/recipient is guaranteed. Multi-Transfer means whether the protocol supports multiple transfers within a single transaction.

Name	Model	ZK techniques	SC	Trustless	Succinctness	Anon.	Multi-Trans.
MimbleWimble	UTXO	HE and Range proofs	No	No	No	Yes	No
Zerocash	UTXO	zk-SNARK	No	No	Yes	Yes	No
Lelantus	UTXO	Bulletproofs and Schnorr proofs	No	Yes	No	Yes	No
Monero	UTXO	RingCT	No	Yes	No	No	No
Quisquis	Hybrid	HE, UPKs and Bulletproofs	No	Yes	No	Yes	Yes
Zether	Account	HE and Σ-Bullets	Yes	Yes	No	No	No
Anon. Zether	Account	HE, Σ-Bullets and many-out-of-many	Yes	Yes	No	Yes	No
ZETH	Hybrid	zk-SNARK	Yes	No	Yes	Yes	No
BlockMaze	Account	zk-SNARK	Yes	No	Yes	Yes	No
Our work	Account	HE, Agg. Bulletproofs and Σ-protocols	Yes	Yes	No	No	Yes

support aggregation of proofs in transactions and batch verification for multiple transfers. An anonymous version of Zether is given in the work [28], which improves the Basic Zether with the *many-out-of-many* cryptographic primitive. This primitive is used to create an anonymity set, ensuring the unlinkability of the sender and receiver in a transaction. However, Anonymous Zether is restricted to the single sender/receiver payment model.

ZETH [16] offers a private payment mechanism by adapting the Zerocash system on top of Ethereum. The main feature is a mixing smart contract that enables the user to *mint* and *pour* an amount of *zethNotes*. A transaction between a sender and multiple recipients is based on the expense of an amount of previously minted notes and the creation of new notes and associated commitments, inserted in a Merkle tree. Double spending on spent notes is avoided with the generation of serial numbers. A zk-SNARK proof is provided by a sender to ensure that the transaction is well-balanced. Furthermore, each note is also encrypted before being sent to its recipient. Although the possibility to do multiple payments in a single transaction, there is no efficient way to check multiple Merkle paths as input notes increase.

BlockMaze [15] is a private payment system in the account model. Each address has both a zero-knowledge balance (ZK-balance) and a plaintext balance. A secure commitment is used for the ZK-balance and a *private computation circuit* for creating the transfer method. The transfer is performed in a two-step procedure to achieve *unlinkability* of sender/receiver addresses: the sender create a ZK-transaction to deposit funds and the receiver provides a ZK-proof to receive funds. The adopted zk-SNARKs inherit the trusted setup of Zerocash. Moreover, BlockMaze does not support the transfer of funds to multiple recipients within a single transaction.

A comparison between the CT solutions is shown in Table 3. Our solution benefits from trustless and multi-transfer support, but does not guarantee the unlinkability of sender/receiver address.

7. Implementation and evaluation

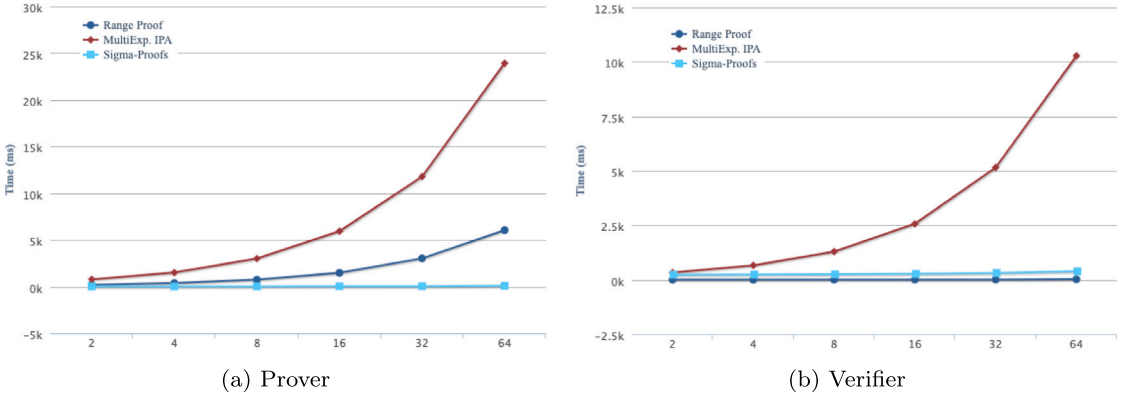
In this section, we present our implementation and concrete evaluation of ZeroMT. The source code written in Rust can be found on GitHub [39] and utilizes `merlin` [40] and `arkworks` [30] ecosystems. These libraries provide a Fiat–Shamir heuristic for building non-interactive proof systems through the STROBE-based transcript [41] and the arithmetic behind elliptic curve points and finite field elements. Arkworks also provides serialization mechanisms to convert scalar field elements and elliptic curve points into their byte representation. Our implementation is modular and allows us to identify which are the expensive components of the cryptographic scheme. It turns out that ZK-proofs are the most expensive components, hence we evaluate them in terms of proving time, verifying time and proof sizes. We use serialization in an uncompressed form which requires 64 bytes for an elliptic curve point and 32 bytes for a scalar field element. There is also an overhead of 8 bytes for serializing a vector object's memory reference from the Rust libraries. We use the Barreto–Naehrig elliptic curve known as `BN-254` since it is the choice of Zether's smart contract. In Table 4, we report our benchmarks for different prover and verifier executions and proof sizes, considering the $n = 32$-bit range domain and varying the number m of aggregate transfers, from 2 up to 64 values. Fig. 2 shows the prover and verifier times when the number of transfers increases. For range proofs, we implement aggregate RP and the *multi-exponentiation* technique on the IPA verifier. The benchmarks are executed on a 2.6 GHz 6 cores CPU machine running the Rust compiler and with 16 GB RAM. From the evaluations in Table 4, it can be seen that the IPA component significantly weighs on the total results of our ZK-proof. The overall size of our ZK-proof is of $928+128 \cdot log_2(m \cdot n)$ bytes, considering that there are $2\ \mathbb{G}^{log_2(m \cdot n)}$ elements at each MultiExp IPA round, which is in line with the asymptotic proof size of $\mathcal{O}(log(n \cdot m))$ from the theoretical result.

Table 4
Evaluation of multi-transfer ZK-proof. `Range Proof` refers to the implementation of Protocol 1 in Section 5.2. Σ-Proof refers to the implementation of Protocol 2 in Section 5.3. `Total` is the total cost considering the MultiExp IPA.

m	Range proof	MultiExp. IPA	Σ-Proofs	Total
Prover times (ms)				
2	189	772	6	967
4	372	1 509	10	1 891
8	737	3 002	14	3 753
16	1477	5 930	24	7 431
32	3023	11 778	43	14 844
64	6049	23 921	82	30 052
Verifier times (ms)				
2	5	331	12	348
4	5	663	18	686
8	6	1 296	27	1 329
16	8	2 571	47	2 626
32	14	5 157	88	5 259
64	24	10 275	161	10 460
Proof sizes (bytes)				
2	448	848	416	1 712
4	448	976	416	1 840
8	448	1 104	416	1 968
16	448	1 232	416	2 096
32	448	1 360	416	2 224
64	448	1 488	416	2 352

Comparison with concurrent works. Comparing our solution with other works is difficult due to the different balance models (UTXO, Hybrid or Account), ZK techniques, programming languages and transaction parameters. Here, we consider the concurrent works that are closest to our transaction scheme and balance model, such as Quisquis [11] and Anonymous Zether [28]. Unfortunately, basic Zether [12] does not provide any implementation and the measurements present in the paper are not comparable. Quisquis and Anonymous Zether rely on hybrid and account model respectively and share with us the batching techniques of Bulletproofs. Now, the comparison strictly depends on the transaction parameters.

Quisquis utilizes compressed group and field elements (33 and 32 bytes respectively) which reduce transaction and proof size. Quisquis



(a) Prover (b) Verifier

Fig. 2. Execution times when the number of transfers increases.

Table 5
Comparison with concurrent works. n denotes the bit-range domain (for a value $v \in [0, 2^n - 1]$). N denotes the number of participants in a transaction. m denotes the number of aggregate values.

	tx cost (ms)		tx cost (Big-$\mathcal{O}$)		tx cost (bytes)	
	Prover	Verifier	Proof size	tx size	Proof size	tx size
QuisQuis	471	71,6	$log(n)$	$N + log(n)$	13 408	26 060
Anon. Zether	2478	152	$log(N)$	N	3776	6148
Our work (ZeroMT)	7431	2626	$log(n \cdot m)$	N	2096	5104

considers a number of $N = 16$ participants in one transaction, of which we have 1 sender, 3 receivers and 12 randomly selected accounts for the anonymity set. Quisquis also considers a 64 bit-range domain, which inflates transaction costs. Anonymous Zether uses uncompressed group and field elements like in our work. The range domain is set to 32 bit-range, and the proof aggregates a fixed number of $m = 2$ range values. The number of participants considered in Anonymous Zether is also $N = 16$. In our work, to be better aligned in the comparison, we consider a number of participants of $N = 16$, of which we have 1 sender and 15 recipients, with $m = 16$ aggregate values under 32 bit-range domain. The comparison is shown in Table 5.

8. ZeroMT applications

In this section, we discuss the instantiation of our multi-transfer scheme in real-world scenarios, specifically the smart lock and IoT/Mec continuous payments case studies. Therefore, we highlight how ZeroMT enhances both privacy and scalability in these contexts.

Smart lock case study. A simple real-world case study of the smart lock and eBike rental service is presented in [29]. The protocol follows a client/server paradigm where, after a sequence of interactions performed off-chain between clients and servers, a final transaction is assembled by the client and sent to the blockchain (e.g. via smart contract). In this settings, a User (client) can initiate a rental service by requesting the opening of a smart lock. Then, upon concluding the service, the client settles the rental fees for the duration of the service with both the Service Providers and the Insurance (servers). The smart lock protocol is clearly executed off-chain before the payments towards multiple recipients can start. The blockchain serves as a means of ensuring the integrity of the client's data and facilitating balance transfers between the parties. However, due to the openness of the blockchain, there are some considerations for the privacy of clients and servers. Trivial solutions to safeguard privacy could consider excluding clients from blockchain payments or refraining from storing their identity within the smart contract. However, these approaches pose challenges in deterring dishonest behavior or automating dispute resolution. For servers instead, there is the risk of revealing balances and financial details of their services to competitors. The confidentiality and zero-knowledge properties of ZeroMT can be leveraged to enable private transactions. Indeed, transactions do not disclose their amounts and the balance values of the parties involved. Furthermore, it is evident that issuing separate transactions at the end of the off-chain protocol results in inefficient work from the main-chain. By exploiting the multi-transfer feature of ZeroMT, the client can aggregate multiple transfers in a single transaction. Therefore, the multi-transfer scheme can be integrated together with the smart lock protocol as follows. A `Setup` is executed to initialize and deploy the smart contract into the blockchain network. Any registered Users within the smart contract can execute the `Fund` method to add funds to their balance. At any time, Users execute the smart lock protocol to request the rental service. At the end of the smart lock session, Users run on their devices the `MultiTrans` method in order to finalize payments to multiple recipients, i.e., the Service Providers and the Insurance. The `MultiTrans` method generates the multi-transfer transaction that is sent to the `Verify` method of the smart contract. `Verify` executes the verification of the ZK-proof, and in case of success, a redistribution of balances of all participants takes place.

IoT/MEC multiple continuous payments. Another case study where ZeroMT could enhance scalability is presented in [42]. Here, the high-frequency of data exchange among Internet of Things (IoT) and Mobile Edge Computing (MEC) devices leads to continuous payments when the IoT/MEC system is integrated with the blockchain. Given also the adoption of ZK-proofs, this integration becomes more challenging. The scenario IoT/MEC and blockchain is represented following the data Buyer/Seller paradigm. Thus, an IoT device (Seller) collects data, which are then sent to the Edge node (Buyer). The IoT device sells its data at a predetermined cost and, upon receiving the data, the Edge node submits a transaction on the blockchain to make the requested payment. Considering the frequency of data exchanges, the Edge node will continuously issue payments. Therefore, the work in [42] focuses on minimizing the number of transactions that need to be verified by the blockchain. However, their solution is limited to a one-to-one Seller/Buyer relationship, even though it is quite plausible to envision an Edge node purchasing data from multiple IoT devices. In that case, to further reduce the number of transactions and the computational overhead from the ZK-proofs, the multi-transfer scheme can be adopted as follows. Rather than collecting data from a single IoT device, the Edge node makes data requests to multiple IoT devices. After gathering all the data, the Edge node assembles and submits a multi-transfer transaction to the blockchain to finalize the payments towards multiple Sellers/IoT devices.

9. Limitations and discussions

In this section, we discuss the limitations and potential mitigations of our ZeroMT multi-transfer scheme.

Downside of the Inner-Product argument. The IPA protocol, originally conceived in Bulletproofs, reveals an important limitation. On both sides of the prover and verifier, $log_2(n)$ rounds are executed, where n is the size of the vector sent each round equal to a power of two. While this characteristic is not an issue in the original design, it becomes more significant when applied. In our multi-transfer proof system, the dimension of the input vector is $m{\cdot}n$, where m is the number of transfers to each recipient plus one (the sender balance value), and n the range domain. According to the design of Bulletproofs, such $m \cdot n$ must be a power of two. As a consequence, this limitation is reflected in the value of m, meaning that our multi-transfer scheme cannot effectively support transfers towards an arbitrary number of recipients; rather, the number of recipients (plus one) is equal to the power of two. Given a scenario in which a party wants to make an arbitrary number of transfers a, a trivial solution could be to additionally transfer padding zero amounts p such that $\exists\ t \in \mathbb{N} : a + p = 2^t - 1$. This mitigation does not affect the multi-transfer proof system. The range proof and the IPA can still generate valid proofs since each padding amount p is non-negative and does not modify the sender remaining balance $\hat{b}$. When it comes to Σ-protocols, since the added padding values are equal to zero, their corresponding ciphertexts also have a value of zero, thus satisfying the verifier's overdraft-safety equation.

ZK-proof verification time. Despite the logarithmic-sized proofs, it turns out that the IPA subroutine significantly weighs on the whole ZK-proof verification time. Indeed, the verifier logic of the IPA involves a linear-time operation. Nonetheless the drawback of non-succinct verification, our ZK-proof is still optimal to amortize costs. However, we could significantly optimize the verification costs by following a similar strategy of [21], where an amortized succinctness is proposed introducing accumulators. With this method, the costly operations are batched and performed outside the verifier logic. In particular, the linear-time work of the verifier is deferred in a single step at the end of the proof. At each intermediate step, instead, the verifier does the succinct work. Applying this technique to the IPA, the verifier asymptotically results in a logarithmic cost barring the single linear time check. Additionally, the accumulator serves as a means of batching multiple proofs, and the linear time check can be performed once for an entire batch. Effectively, the cost of a non-succinct proof can be amortized over many proofs.

Unlinkability vs. traceability. Our multi-transfer scheme does not hide the link between sender and receiver in a transaction, i.e., the unlinkability of addresses. Although it seems a limitation, our purpose for privacy is confidentiality only. This because we believe that the traceability of transactions is an important property of the blockchain that must be preserved. Anyway, hiding the link between multiple parties of a multi-transfer is a challenging task. In that direction, the many-out-of-many [28] primitive could potentially be integrated into our ZK-proof, thereby forming an anonymity set within the multi-transfer scheme.

10. Conclusion and future work

ZeroMT is a new multi-transfer private payment mechanism for CT protocols. In summary, ZeroMT enriches account-based private payments with the following features: (i) *Multi-Transfer*: a single transaction can make private transfers to multiple receivers; (ii) *Confidential Transactions*: transactions do not reveal sender/receiver balances and the transfer amounts; (iii) *Zero-Knowledge*: transaction secrets cannot be disclosed in the verification process; (iv) *Non-Interactive*: single interaction from prover to the verifier of transaction statements; (v) *Trustless*: no additional trust is required from the ZK-proof or trusted third-party; (vi) *Aggregation*: an aggregate proof can be constructed for the validity of multiple transfers. ZeroMT relies on homomorphic cryptographic primitives and it is suitable for account-based blockchains and smart contract protocols. The concept of multi-transfer can be of interest for applications such as multi-party off-chain protocols and multiple continuous payments. Benefits are also shown in terms of scalability of CT protocols. For example, in platforms such as Ethereum that charge gas fees, executing the Zether transfer requires 7188k gas consumption for a single transfer. In contrast, with ZeroMT we are able to execute up to 15 aggregate transfers with a gas cost of around 7183k (the estimate is made with the EIP-1108 [43] and the Ethereum yellow paper, and refers to the costs of addition and multiplication of scalar and curve points, exponentiation and so on, when the `Verify` method is executed in a smart contract). Moreover, the costs are mainly for generating and verifying the ZK-proof and the question arises of how convenient is the aggregate proof of multiple transfers. Indeed, generating an aggregate proof for multiple transfers (comprising 2 to 64 transfers) takes approximately $37,58\%$ and $54,84\%$ less time compared to generating separate proofs for each transfer taken together. Similarly, verifying an aggregate proof takes $37,46\%$ to $58,14\%$ less time compared to verifying separate proofs. Moreover, the aggregate proof size is significantly smaller, ranging from $63,97\%$ to $97,77\%$ reduction compared to the combined size of all proofs. Future work is aimed at optimizing the IPA subroutine of the ZK-proof, as it may significantly reduce costs.

CRediT authorship contribution statement

Emanuele Scala: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Changyu Dong:** Supervision, Writing – review & editing. **Flavio Corradini:** Supervision. **Leonardo Mostarda:** Funding acquisition, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Appendix A. Proof of Theorem 5.1

The *Aggregate RP* has perfect completeness, special soundness and perfect special honest verifier zero-knowledge from Theorem 2 of [13]. We now show that our Σ_{AggRP} is an argument of knowledge of the relation (1).

Perfect completeness follows from equality 46 of Protocol 1. For simplicity we do not consider the challenges coming from Bulletproofs inherited in τ_x. If $A_t = g^{-k_{ba}} h^{k_\tau}$, $s_{ba} = k_{ba} + c(\hat{b}z^2 + \sum_{i=1}^{n} a_i z^{i+2})$ and $s_\tau = k_\tau + c\tau_x$ then:

$$
\begin{aligned}
& g^{c(\hat{t}-\psi(y,z))-s_{ba}} \cdot h^{s_\tau} \\
&= g^{-k_{ba}} \cdot h^{k_\tau} \cdot g^{c(\hat{t}-\psi(y,z))} \cdot g^{c(-\hat{b}z^2-\sum_{i=1}^{n} a_i z^{i+2})} \cdot h^{c\tau_x} \\
&= A_t \cdot (g^{(\hat{t}-\psi(y,z))-\hat{b}z^2-\sum_{i=1}^{n} a_i z^{i+2}} \cdot h^{\tau_x})^c \\
&= A_t \cdot T_{1,2}^c
\end{aligned}
$$

So the verifier accepts the transcript $(A_t, c, (s_{ba}, s_\tau))$.

For special soundness, suppose two accepting transcripts $(A_t, c, (s_{ba}, s_\tau))$ and $(A_t, c', (s'_{ba}, s'_\tau))$ with $c \neq c'$, we show that witnesses w_{ba} and w_τ can be extracted from these two transcripts. Let $(c - c')^{-1}$ denote the multiplicative inverse of $(c - c')$ mod $|\mathbb{Z}_p|$ and define:

$$w_{ba} = (s_{ba} - s'_{ba}) \cdot (c - c')^{-1}$$

$$w_\tau = (s_\tau - s'_\tau) \cdot (c - c')^{-1}$$

Since $(A_t, c, (s_{ba}, s_\tau))$ and $(A_t, c', (s'_{ba}, s'_\tau))$ are both accepting transcripts, it holds that:

$$g^{c \cdot t - s_{ba}} \cdot h^{s_\tau} = A_t T_{1,2}^c$$
$$g^{c' \cdot t - s'_{ba}} \cdot h^{s'_\tau} = A_t T_{1,2}^{c'}$$

where $t = \hat{t} - \psi(y, z)$. From the definition of A_t and $T_{1,2}$ we have:

$$\begin{aligned} A_t \cdot T_{1,2}^c &= g^{-k_{ba}} h^{k_\tau} \cdot (g^{t - \hat{b} \cdot z^2 - \sum_{i=1}^n a_i \cdot z^{2+i}} h^{\tau_x})^c \\ &= g^{c \cdot t - k_{ba} - c \cdot (\hat{b} \cdot z^2 + \sum_{i=1}^n a_i \cdot z^{2+i})} h^{k_\tau + c\tau_x} = g^{c \cdot t - s_{ba}} h^{s_\tau} \end{aligned}$$

Similarly, for the second equality:

$$A_t \cdot T_{1,2}^{c'} = g^{c' \cdot t - k_{ba} - c' \cdot (\hat{b} \cdot z^2 + \sum_{i=1}^n a_i \cdot z^{2+i})} h^{k_\tau + c'\tau_x} = g^{c' \cdot t - s'_{ba}} h^{s'_\tau}$$

Together, these equations imply that:

$$\begin{aligned} \frac{g^{c \cdot t - s_{ba}} h^{s_\tau}}{g^{c' \cdot t - s'_{ba}} h^{s'_\tau}} &= g^{t(c-c') - (s_{ba} - s'_{ba})} h^{s_\tau - s'_\tau} \\ &= g^{t(c-c') - (\hat{b} \cdot z^2 + \sum_{i=1}^n a_i \cdot z^{2+i})(c-c')} h^{\tau_x (c-c')} \end{aligned}$$

Solving two linear equations we deduce:

$$\hat{b} \cdot z^2 + \sum_{i=1}^n a_i \cdot z^{2+i} = (s_{ba} - s'_{ba})(c - c')^{-1} = w_{ba}$$

$$\tau_x = (s_\tau - s'_\tau)(c - c')^{-1} = w_\tau.$$

For perfect special honest verifier zero-knowledge, we show that there exists a PPT simulator S that can simulate verifying transcripts without the knowledge of the witness and can produce a distribution over transcripts identical to the distribution produced by the honest verifier and prover. The simulator is given the public instances $(g, h, \hat{t}, \delta)$ and uses the verifier's randomness x, y and z. The simulator computes a random challenge $c \xleftarrow{\$} \mathbb{Z}_p$ and random $s_{ba}, s_\tau \xleftarrow{\$} \mathbb{Z}_p$. The simulator then computes $T_{1,2} \xleftarrow{\$} \mathbb{G}$ and

$$A_t = g^{c(\hat{t} - \psi(y,z))} \cdot h^{s_\tau} \cdot (T_1^x \cdot T_2^{x^2})^{-c} \cdot g^{-s_{ba}}$$

and produces an accepting transcript $(A_t, c, (s_{ba}, s_\tau))$. Given that c, s_{ba} and s_τ are uniformly random elements generated by S, and so $A_t \in_R \mathbb{G}$ is a random group element, this means they are identically distributed as in the honest protocol.

Appendix B. Proof of Theorem 5.2

Perfect completeness follows from the fact that if the prover follows the protocol, the verifier always accepts. Hence, we show that the equalities 17, 18, 19, and 20 of Protocol 2 are correct. For 17 and 18, the protocol is complete if $A_y = g^{k_{sk}}$ and $s_{sk} = k_{sk} + csk$ are honestly generated by the prover, then

$$g^{s_{sk}} = g^{k_{sk}} \cdot g^{csk} = g^{k_{sk}} \cdot (g^{sk})^c = A_y y^c$$

similarly, if $A_D = g^{k_r}$ and $s_r = k_r + cr$ then

$$g^{s_r} = g^{k_r} \cdot g^{cr} = g^{k_r} \cdot (g^r)^c = A_D D^c$$

So the verifier accepts the transcripts (A_y, c, s_{sk}) and (A_D, c, s_r) respectively.

For equality 19, the protocol is complete if $A_{ba} = ((\frac{C_R}{\prod_{i=1}^n D})^{z^2} \cdot \prod_{i=1}^n D^{z^{i+2}})^{k_{sk}} \cdot g^{k_{ba}}$, $s_{ba} = k_{ba} + c(\hat{b}z^2 + \sum_{i=1}^n a_i z^{2+i})$ and $s_{sk} = k_{sk} + csk$ then:

$$g^{s_{ba}} \left[\left(\frac{C_R}{\prod_{i=1}^n D} \right)^{z^2} \cdot \prod_{i=1}^n D^{z^{i+2}} \right]^{s_{sk}} = g^{k_{ba}} \cdot \left[\left(\frac{C_R}{\prod_{i=1}^n D} \right)^{z^2} \cdot \prod_{i=1}^n D^{z^{i+2}} \right]^{k_{sk}}$$
$$\cdot \left[g^{\hat{b}z^2} \cdot \prod_{i=1}^n g^{a_i z^{i+2}} \right]^c \cdot \left[\left(\frac{C_R}{\prod_{i=1}^n D} \right)^{z^2} \cdot \prod_{i=1}^n D^{z^{i+2}} \right]^{c \cdot sk}$$

$$= A_{ba} \cdot \left[\left(g^{\hat{b}} \left(\frac{C_R}{\prod_{i=1}^n D} \right)^{sk} \right)^{z^2} \cdot \left(\prod_{i=1}^n g^{a_i} \cdot D^{sk} \right)^{z^{i+2}} \right]^c$$
$$= A_{ba} \cdot \left[\left(\frac{C_L}{\prod_{i=1}^n C_i} \right)^{z^2} \cdot \prod_{i=1}^n C_i^{z^{i+2}} \right]^c$$

So, we conclude that the verifier accepts the transcript $(A_{ba}, c, (s_{ba}, s_{sk}))$.

For equality 20, If $A_{\bar{y}} = \prod_{i=1}^n (y \cdot \bar{y}_i^{-1})^{k_r}$ and $s_r = k_r + cr$ then:

$$\begin{aligned} &\prod_{i=1}^n (y \cdot \bar{y}_i^{-1})^{s_r} \\ &= \prod_{i=1}^n (y \cdot \bar{y}_i^{-1})^{k_r} \cdot \prod_{i=1}^n (y \cdot \bar{y}_i^{-1})^{cr} = A_{\bar{y}} \cdot \prod_{i=1}^n \left(\frac{g^{a_i} y^r}{g^{a_i} \bar{y}_i^r} \right)^c \\ &= A_{\bar{y}} \cdot \prod_{i=1}^n \left(\frac{C_i}{\bar{C}_i} \right)^c \end{aligned}$$

So, the verifier accepts the transcript $(A_{\bar{y}}, c, s_r)$.

We now show that Protocol 2 is proof of knowledge, thus special soundness holds for the above equalities. For equality 17, suppose two accepting transcripts (A_y, c, s_{sk}) and (A_y, c', s'_{sk}) with $c \neq c'$, we show that the witness w_{sk} can be extracted in polynomial time from these two transcripts. Let $(c-c')^{-1}$ denote the multiplicative inverse of $(c-c')$ mod $|\mathbb{Z}_p|$ and define $w_{sk} = (s_{sk} - s'_{sk}) \cdot (c - c')^{-1}$. Since (A_y, c, s_{sk}) and (A_y, c', s'_{sk}) are both accepting transcripts, it holds that:

$$g^{s_{sk}} = A_y y^c$$
$$g^{s'_{sk}} = A_y y^{c'}$$

Say that $A_y = g^{k_{sk}}$ and $y = g^{sk}$

$$A_y \cdot y^c = g^{k_{sk}} \cdot g^{csk} = g^{k_{sk} + csk} = g^{s_{sk}}$$
$$A_y \cdot y^{c'} = g^{k_{sk}} \cdot g^{c'sk} = g^{k_{sk} + c'sk} = g^{s'_{sk}}$$

Combining together we have:

$$\frac{g^{s_{sk}}}{g^{s'_{sk}}} = g^{k_{sk} + csk - k_{sk} - c'sk} = g^{sk(c-c')} = g^{s_{sk} - s'_{sk}}$$

Hence $sk(c - c') \equiv s_{sk} - s'_{sk} \ mod \ |\mathbb{Z}_p|$ implies that

$$sk \equiv (s_{sk} - s'_{sk}) \cdot (c - c')^{-1} \ mod \ |\mathbb{Z}_p| = w_{sk}$$

that is $g^{sk} = g^{w_{sk}}$ meaning that w_{sk} is a witness.

With a similar procedure we can prove the same for equality 18. Considering two accepting transcripts (A_D, c, s_r) and (A_D, c', s'_r) with $c \neq c'$, we want to extract a witness $w_r = (s_r - s'_r) \cdot (c - c')^{-1}$. It holds that $g^{s_r} = A_D D^c$ and $g^{s'_r} = A_D D^{c'}$ both are accepted by the verifier. Since $A_D = g^{k_r}$ and $D = g^r$ we have

$$A_D \cdot D^c = g^{k_r} \cdot g^{cr} = g^{k_r + cr} = g^{s_r}$$

$$A_D \cdot D^{c'} = g^{k_r} \cdot g^{c'r} = g^{k_r + c'r} = g^{s'_r}$$

that together imply

$$\frac{g^{s_r}}{g^{s'_r}} = g^{r(c-c')} = g^{s_r - s'_r}$$

Hence we can conclude that $r \equiv (s_r - s'_r) \cdot (c - c')^{-1} \ mod \ |\mathbb{Z}_p| = w_r$, meaning that w_r is a witness.

Special soundness for equality 19 holds the same way. Suppose $(A_{ba}, c, (s_{ba}, s_{sk}))$ and $(A_{ba}, c', (s'_{ba}, s'_{sk}))$ both are accepting transcripts with $c \neq c'$. We show that witnesses w_{ba} and w_{sk} can be extracted in polynomial time. Let $(c - c')^{-1}$ be the multiplicative inverse of $(c - c')$ mod $|\mathbb{Z}_p|$ and define: $w_{ba} = (s_{ba} - s'_{ba})(c - c')^{-1}$ and $w_{sk} = (s_{sk} - s'_{sk})(c - c')^{-1}$. For simplicity, here we rename the z^2 terms as follows: $\frac{C_R}{\prod_{i=1}^n D} = C_{Rn}$ and $\frac{C_L}{\prod_{i=1}^n C_i} = C_{Ln}$. Considering the accepting transcripts it holds that:

$$g^{s_{ba}} \left(C_{Rn}^{z^2} \cdot \prod_{i=1}^n D^{z^{2+i}} \right)^{s_{sk}} = A_{ba} \left(C_{Ln}^{z^2} \cdot \prod_{i=1}^n C_i^{z^{2+i}} \right)^c$$

$$g^{s'_{ba}}\left(C_{Rn}^{z^2}\cdot\prod_{i=1}^{n}D^{z^{2+i}}\right)^{s'_{sk}}=A_{ba}\left(C_{Ln}^{z^2}\cdot\prod_{i=1}^{n}C_i^{z^{2+i}}\right)^{c'}$$

Given that $A_{ba}=(C_{Rn}^{z^2}\cdot\prod_{i=1}^{n}D^{z^{2+i}})^{k_{sk}}\cdot g^{k_{ba}}$, $C_{Ln}=g^{\hat{b}}C_{Rn}^{sk}$, $y=g^{sk}$ and $D=g^{r}$ we have:

$$A_{ba}\left(C_{Ln}^{z^2}\cdot\prod_{i=1}^{n}C_i^{z^{2+i}}\right)^{c}=$$
$$\left(C_{Rn}^{z^2}\cdot\prod_{i=1}^{n}D^{z^{2+i}}\right)^{k_{sk}}\cdot g^{k_{ba}}\cdot\left(C_{Ln}^{z^2}\cdot\prod_{i=1}^{n}C_i^{z^{2+i}}\right)^{c}=$$
$$\left(C_{Rn}^{z^2}\cdot\prod_{i=1}^{n}D^{z^{2+i}}\right)^{k_{sk}}\cdot g^{k_{ba}}\cdot\left(g^{\hat{b}z^2}\cdot C_{Rn}^{sk\cdot z^2}\cdot\prod_{i=1}^{n}g^{a_i z^{2+i}}\cdot\prod_{i=1}^{n}D^{sk\cdot z^{2+i}}\right)^{c}=$$
$$\left(C_{Rn}^{z^2}\cdot\prod_{i=1}^{n}D^{z^{2+i}}\right)^{k_{sk}}\cdot g^{k_{ba}}\cdot\left(g^{\hat{b}z^2}\cdot\prod_{i=1}^{n}g^{a_i z^{2+i}}\right)^{c}\cdot\left(C_{Rn}^{z^2}\cdot\prod_{i=1}^{n}D^{z^{2+i}}\right)^{sk\cdot c}=$$
$$\left(C_{Rn}^{z^2}\cdot\prod_{i=1}^{n}D^{z^{2+i}}\right)^{k_{sk}+csk}\cdot g^{k_{ba}+c(\hat{b}z^2+\sum_{i=1}^{n}a_i z^{2+i})}=$$
$$\left(C_{Rn}^{z^2}\cdot\prod_{i=1}^{n}D^{z^{2+i}}\right)^{s_{sk}}\cdot g^{s_{ba}}$$

With a similar procedure, we can obtain:

$$A_{ba}\left(C_{Ln}^{z^2}\cdot\prod_{i=1}^{n}C_i^{z^{2+i}}\right)^{c'}=\left(C_{Rn}^{z^2}\cdot\prod_{i=1}^{n}D^{z^{2+i}}\right)^{s'_{sk}}\cdot g^{s'_{ba}}$$

These equalities together imply:

$$\frac{\left(C_{Rn}^{z^2}\cdot\prod_{i=1}^{n}D^{z^{2+i}}\right)^{s_{sk}}\cdot g^{s_{ba}}}{\left(C_{Rn}^{z^2}\cdot\prod_{i=1}^{n}D^{z^{2+i}}\right)^{s'_{sk}}\cdot g^{s'_{ba}}}=\left(C_{Rn}^{z^2}\cdot\prod_{i=1}^{n}D^{z^{2+i}}\right)^{(c-c')\cdot sk}\cdot g^{(c-c')\cdot(\hat{b}z^2+\sum_{i=1}^{n}a_i z^{2+i})}$$

Hence, we deduce that:

$$sk\equiv(s_{sk}-s'_{sk})(c-c')^{-1}\ mod\ |\mathbb{Z}_p|=w_{sk}$$

$$\hat{b}z^2+\sum_{i=1}^{n}a_i z^{2+i}\equiv(s_{ba}-s'_{ba})(c-c')^{-1}\ mod\ |\mathbb{Z}_p|=w_{ba}$$

Finally, special soundness is given for equality 20. Suppose two accepting transcripts $(A_{\bar{y}},c,s_r)$ and $(A_{\bar{y}},c',s'_r)$ with $c\neq c'$, we build an extractor for witness $w_r=(s_r-s'_r)\cdot(c-c')^{-1}$. From the two accepted transcripts we have:

$$\prod_{i=1}^{n}(y\cdot\bar{y}_i^{-1})^{s_r}=A_{\bar{y}}\cdot(\prod_{i=1}^{n}C_i/\bar{C}_i)^{c}$$
$$\prod_{i=1}^{n}(y\cdot\bar{y}_i^{-1})^{s'_r}=A_{\bar{y}}\cdot(\prod_{i=1}^{n}C_i/\bar{C}_i)^{c'}$$

Given $A_{\bar{y}}=\prod_{i=1}^{n}(y\cdot\bar{y}_i^{-1})^{k_r}$ and $\prod_{i=1}^{n}\frac{C_i}{\bar{C}_i}=\prod_{i=1}^{n}\frac{g^{a_i}y^r}{g^{a_i}\bar{y}_i^r}$ we have:

$$A_{\bar{y}}\cdot(\prod_{i=1}^{n}C_i/\bar{C}_i)^{c}=$$
$$\prod_{i=1}^{n}(y\cdot\bar{y}_i^{-1})^{k_r}\cdot\prod_{i=1}^{n}(y^r\cdot\bar{y}_i^{-r})^{c}=\prod_{i=1}^{n}(y\cdot\bar{y}_i^{-1})^{k_r}\cdot\prod_{i=1}^{n}(y\cdot\bar{y}_i^{-1})^{cr}$$
$$=\prod_{i=1}^{n}(y\cdot\bar{y}_i^{-1})^{k_r+cr}=\prod_{i=1}^{n}(y\cdot\bar{y}_i^{-1})^{s_r}$$

Similarly, for the second equality we have:

$$A_{\bar{y}}\cdot(\prod_{i=1}^{n}C_i/\bar{C}_i)^{c'}=\prod_{i=1}^{n}(y\cdot\bar{y}_i^{-1})^{k_r+c'r}=\prod_{i=1}^{n}(y\cdot\bar{y}_i^{-1})^{s'_r}$$

that together imply:

$$\frac{\prod_{i=1}^{n}(y\cdot\bar{y}_i^{-1})^{s_r}}{\prod_{i=1}^{n}(y\cdot\bar{y}_i^{-1})^{s'_r}}=$$
$$\prod_{i=1}^{n}(y\cdot\bar{y}_i^{-1})^{k_r+cr-k_r-c'r}=\prod_{i=1}^{n}(y\cdot\bar{y}_i^{-1})^{(c-c')r}=\prod_{i=1}^{n}(y\cdot\bar{y}_i^{-1})^{s_r-s'_r}$$

Hence, $(c-c')r\equiv s_r-s'_r\ mod\ |\mathbb{Z}_p|$ implies that:

$$r\equiv(s_r-s'_r)(c-c')^{-1}\ mod\ |\mathbb{Z}_p|=w_r$$

meaning w_r is a witness.

We now show that Protocol 2 is perfect special honest verifier zero-knowledge. There exists a PPT simulator $\mathcal{S}$ which is given the public instances $(y,\bar{\mathbf{y}},C_L,C_R,\mathbf{C},\bar{\mathbf{C}},D,g)$ and the verifier's internal randomness z. Without the knowledge of the witness, the simulator produces a valid transcript with an identical, or indistinguishable, distribution to that produced in the honest protocol. Hence, the simulator computes the challenge $c\xleftarrow{\$}\mathbb{Z}_p$ and $s_{sk},s_r,s_{ba}\xleftarrow{\$}\mathbb{Z}_p$. For simplicity, we substitute $\frac{C_R}{\prod_{i=1}^{n}D}=C_{Rn}$ and $\frac{C_L}{\prod_{i=1}^{n}C_i}=C_{Ln}$, then the simulator computes:

$$A_y=g^{s_{sk}}\cdot y^{-c}\quad\in_R\mathbb{G}$$

$$A_D=g^{s_r}\cdot D^{-c}\quad\in_R\mathbb{G}$$

$$A_{ba}=g^{s_{ba}}\cdot(C_{Ln}^{z^2}\cdot\prod_{i=1}^{n}C_i^{z^{i+2}})^{-c}\cdot(C_{Rn}^{z^2}\cdot\prod_{i=1}^{n}D^{z^{i+2}})^{s_{sk}}\quad\in_R\mathbb{G}$$

$$A_{\bar{y}}=\prod_{i=1}^{n}(y\cdot\bar{y}_i^{-1})^{s_r}\cdot\prod_{i=1}^{n}(C_i/\bar{C}_i)^{-c}\quad\in_R\mathbb{G}$$

and sends to the verifier $A_y,A_D,A_{ba},A_{\bar{y}},c,s_{sk},s_r,s_{ba}$.

Given that g,D,y and $\bar{y}$ are random group elements and that c,s_{sk},s_r and s_{ba} are uniformly random challenges generated by $\mathcal{S}$, this implies that each A is a random group element $\in_R\mathbb{G}$. This means they are identically distributed as in the honest protocol.

Appendix C. Overdraft safety and privacy

In the light of Theorems 5.1 and 5.2 we can say that from the soundness of our ZK-proof in interactive form, the Multi-Transfer scheme has overdraft-safety security. In particular, from the existence of the extractor $\mathcal{E}$, the advantage of every adversary $\mathcal{A}$ in the following experiment is negligible:

$$Pr\left[\begin{array}{l}\sigma\leftarrow\mathtt{Setup}(1^{\lambda}),\ (x,tr,tr')\leftarrow\mathcal{A}(\sigma),\\ w\leftarrow\mathcal{E}(\sigma,tr,tr')\ :\\ \langle\mathcal{A}(\sigma,x,w),\mathcal{V}(\sigma,x)\rangle=1,(x,w)\notin\mathcal{R}.\end{array}\right]\leq\mathtt{negl}(\lambda)$$

where tr,tr' are two accepting transcripts with distinct challenges, $\langle\mathcal{A}(.),\mathcal{V}(.)\rangle$ is an interactive proof between adversary $\mathcal{A}$ and verifier $\mathcal{V}$ and $\mathtt{negl}$ is a negligible function in the security parameter λ.

We could also continue further reductions, until we define an adversary's advantage on all the underlying assumptions (e.g., DLOG). However, our modifications are minimal, and such a strategy would lead to a complex proof similar to [28]. Moreover, in our definition of soundness we use the "rewind technique" for 3-round protocols. As in [13], a more general multi-round protocol technique can be adopted, and subsequently conducted via the "forking lemma". However, in our case it is not strictly necessary, since we only integrate Σ-protocols.

Finally, still in the light of Theorems 5.1 and 5.2 we can say that from the zero-knowledge of our ZK-proof in interactive form, the Multi-Transfer scheme has the property of privacy. In particular, from the existence of the simulator $\mathcal{S}$, the advantage of every adversary $\mathcal{A}$ in

the following experiment is defined:

$$Pr\left[\begin{array}{l}\sigma \leftarrow \texttt{Setup}(1^{\lambda}),\ (x,w,\mu) \leftarrow \mathcal{A}(\sigma),\\ tr_0 \leftarrow \langle \mathcal{P}(\sigma,x,w), \mathcal{V}(\sigma,x)\rangle,\ tr_1 \leftarrow \mathcal{S}(\sigma,x,\mu),\\ b \xleftarrow{\$} \{0,1\};\ b' \in \{0,1\} \leftarrow \mathcal{A}(tr_b)\ :\\ \quad b' = b,\ (x,w) \in \mathcal{R}\end{array}\right] \leq \frac{1}{2} + \texttt{negl}(\lambda)$$

With, μ the verifier's internal randomness, $\langle \mathcal{P}(.), \mathcal{V}(.)\rangle$ the real transcript of the honest interactive proof between prover $\mathcal{P}$ and verifier $\mathcal{V}$ and `negl` a negligible function in the security parameter λ.

References

[1] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. Technical report, Manubot; 2019.
[2] Buterin V, et al. A next-generation smart contract and decentralized application platform.
[3] Chan W, Olmsted A. Ethereum transaction graph analysis. In: 2017 12th international conference for internet technology and secured transactions. ICITST, IEEE; 2017, p. 498–500.
[4] Fleder M, Kester MS, Pillai S. Bitcoin transaction graph analysis. 2015, arXiv preprint arXiv:1502.01657.
[5] Ron D, Shamir A. Quantitative analysis of the full bitcoin transaction graph. In: International conference on financial cryptography and data security. Springer; 2013, p. 6–24.
[6] Androulaki E, Karame GO, Roeschlin M, Scherer T, Capkun S. Evaluating user privacy in bitcoin. In: Financial cryptography and data security: 17th international conference, FC 2013, okinawa, Japan, April 1-5, 2013, revised selected papers 17. Springer; 2013, p. 34–51.
[7] Poelstra A. Mimblewimble. 2016.
[8] Sasson EB, Chiesa A, Garman C, Green M, Miers I, Tromer E, Virza M. Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE symposium on security and privacy. IEEE; 2014, p. 459–74.
[9] Jivanyan A. Lelantus: Towards confidentiality and anonymity of blockchain transactions from standard assumptions. IACR Cryptol ePrint Arch 2019;2019:373.
[10] Alonso KM, et al. Zero to monero. 2020.
[11] Fauzi P, Meiklejohn S, Mercer R, Orlandi C. Quisquis: A new design for anonymous cryptocurrencies. In: International conference on the theory and application of cryptology and information security. Springer; 2019, p. 649–78.
[12] Bünz B, Agrawal S, Zamani M, Boneh D. Zether: Towards privacy in a smart contract world. In: International conference on financial cryptography and data security. Springer; 2020, p. 423–43.
[13] Bünz B, Bootle J, Boneh D, Poelstra A, Wuille P, Maxwell G. Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE symposium on security and privacy. SP, IEEE; 2018, p. 315–34.
[14] Hao F. Schnorr non-interactive zero-knowledge proof. Technical report, 2017.
[15] Guan Z, Wan Z, Yang Y, Zhou Y, Huang B. Blockmaze: An efficient privacy-preserving account-model blockchain based on zk-SNARKs. IEEE Trans Dependable Secure Comput 2020.
[16] Rondelet A, Zajac M. Zeth: On integrating zerocash on ethereum. 2019, arXiv preprint arXiv:1904.00905.
[17] Blum M, Feldman P, Micali S. Non-interactive zero-knowledge and its applications. In: Providing sound foundations for cryptography: on the work of shafi goldwasser and silvio micali. 2019, p. 329–49.
[18] Ben-Sasson E, Chiesa A, Green M, Tromer E, Virza M. Secure sampling of public parameters for succinct zero knowledge proofs. In: 2015 IEEE symposium on security and privacy. IEEE; 2015, p. 287–304.
[19] Groth J. On the size of pairing-based non-interactive arguments. In: Annual international conference on the theory and applications of cryptographic techniques. Springer; 2016, p. 305–26.
[20] Bootle J, Cerulli A, Chaidos P, Groth J, Petit C. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In: Annual international conference on the theory and applications of cryptographic techniques. Springer; 2016, p. 327–57.
[21] Bowe S, Grigg J, Hopwood D. Recursive proof composition without a trusted setup. Cryptol ePrint Arch 2019.
[22] Bünz B, Chiesa A, Mishra P, Spooner N. Proof-carrying data from accumulation schemes. Cryptol ePrint Arch 2020.
[23] Xiong AL, Chen B, Zhang Z, Bünz B, Fisch B, Krell F, Camacho P. VERI-ZEXE: Decentralized private computation with universal setup. Cryptol ePrint Arch 2022.
[24] Daza V, Ràfols C, Zacharakis A. Updateable inner product argument with logarithmic verifier and applications. In: IACR international conference on public-key cryptography. Springer; 2020, p. 527–57.
[25] Bünz B, Maller M, Mishra P, Tyagi N, Vesely P. Proofs for inner pairing products and applications. In: International conference on the theory and application of cryptology and information security. Springer; 2021, p. 65–97.
[26] Lee J. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In: Theory of cryptography conference. Springer; 2021, p. 1–34.
[27] Scala E, Mostarda L. Range proofs with constant size and trustless setup. In: International conference on advanced information networking and applications. Springer; 2023, p. 301–10.
[28] Diamond BE. Many-out-of-many proofs and applications to anonymous zether. In: 2021 IEEE symposium on security and privacy. SP, IEEE; 2021, p. 1800–17.
[29] Corradini F, Mostarda L, Scala E. ZeroMT: Multi-transfer protocol for enabling privacy in off-chain payments. In: International conference on advanced information networking and applications. Springer; 2022, p. 611–23.
[30] arkworks-rs. arkworks.
[31] Scala E, Dong C, Corradini F, Mostarda L. Zero-knowledge multi-transfer based on range proofs and homomorphic encryption. In: Advanced information networking and applications: proceedings of the 37th international conference on advanced information networking and applications (AINA-2023), volume 2. Springer; 2023, p. 461–72.
[32] Katz J, Lindell Y. Introduction to modern cryptography. CRC Press; 2020.
[33] Thaler J, et al. Proofs, arguments, and zero-knowledge. Found Trends Priv Secur 2022;4(2–4):117–660.
[34] Fiat A, Shamir A. How to prove yourself: Practical solutions to identification and signature problems. In: Conference on the theory and application of cryptographic techniques. Springer; 1986, p. 186–94.
[35] Abdalla M, An JH, Bellare M, Namprempre C. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In: International conference on the theory and applications of cryptographic techniques. Springer; 2002, p. 418–33.
[36] Attema T, Fehr S, Klooß M. Fiat-shamir transformation of multi-round interactive proofs. In: Theory of cryptography: 20th international conference, TCC 2022, chicago, IL, USA, November 7–10, 2022, proceedings, part i. Springer; 2022, p. 113–42.
[37] Raikwar M, Gligoroski D. DoS attacks on blockchain ecosystem. In: European conference on parallel processing. Springer; 2021, p. 230–42.
[38] Noether S, Mackenzie A, et al. Ring confidential transactions. Ledger 2016;1:1–18.
[39] EmanueleSc. ZeroMT.
[40] zkcrypto. Merlin: composable proof transcripts for public-coin arguments of knowledge.
[41] Strobe protocol framework.
[42] Boo E, Kim J, Ko J. LiteZKP: Lightening zero-knowledge proof-based blockchains for IoT and edge platforms. IEEE Syst J 2021;16(1):112–23.
[43] Cardozo AS, Williamson Z. Eip-1108: Reduce alt bn128 precompile gas costs. Ethereum Improv Propos 2018;(1108).