Zero-knowledge Multi-Transfer based on range proofs and homomorphic encryption*

Emanuele Scala, Changyu Dong, Flavio Corradini, and Leonardo Mostarda

Abstract Zero-knowledge proofs are widely adopted in Confidential Transactions (CTs). In particular, with these proofs, it is possible to prove the validity of transactions without revealing sensitive information. This has become an attractive property in public blockchain where transactions must be publicly verifiable. However, several challenges must be addressed in order not to alter important properties of the blockchain, such as not introducing trusted third parties and/or circuit-dependent trusted setups. Moreover, there are limited proposals working on the standard account model and considering extended payment models where multiple payees are involved in one transaction. With this paper, we first present our concept of *Multi-Transfer* (MT) in CTs settings, i.e., a transfer that involves multiple payees in a single transaction with privacy guarantees for balances and transfer amounts. Inspired by the work of Zether, we design the MT zero-knowledge proof system, named *MTproof*, by combining the aggregate version of Bulletproofs and several Σ-Protocols to prove that an MT transaction is legit. We provide concrete evaluations of the *MTproof* in terms of proof size, prover and verifier execution time.

1 INTRODUCTION

With the rise of the first digital currency of Bitcoin [17], blockchain technology has become a growing active field of study in both academia and industry. Ethereum [5] has extended the functionality of the blockchain with smart contracts. The transactions enclosed in the blocks are executed according to the logic specified in smart

Emanuele Scala · Flavio Corradini · Leonardo Mostarda
Computer Science, University of Camerino, e-mail: {name.surname}@unicam.it

Changyu Dong
Guangzhou University, e-mail: {name.surname}@gzhu.edu.cn

* This work was supported by HD3FLAB project the Nationally funded by POR MARCHE FESR.

contracts. The state updates produced in a smart contract are propagated to all nodes of the network and immutably stored on the blockchain. This public announcement brings the transaction information publicly visible. Further, a known problem is that an adversary can link the user's real-world identity and disclose transaction information by analysing the public transaction graph [7, 13, 19]. In light of this, an increasing body of research is focusing on privacy applications such as *Confidential Transactions* (CTs), where guarantees on the anonymity of the entities involved in a financial transfer and the confidentiality of the transfer amounts are considered. *Zero-Knowledge Proofs* (ZKPs) are spreading widely in CTs, as they are capable of revealing nothing except the validity of the statements being proved relating to the transactions. Therefore, ZKPs are adopted in many privacy protocols, among the most cited are MimbleWimble [18], Zerocash [22], Lelantus [15] and Monero [1]. Such proposals are designed on the UTXO (unspent transaction outputs) model and move around a simple equation for a valid transaction: the total of the input amounts must equal the total of the output amounts. To keep the amounts hidden, this equation is then proved through the *commitment to zero* or a private computation linked to an *arithmetic circuit* (e.g., zk-SNARK). On another side, there are privacy protocols that move to the Ethereum account model: a model in which a transfer of funds between two addresses has the effect that the payer's account balance is debited and the payee's account balance is credited. In this setting, proposals of Zether [3] and Quisquis [11] design schemes in which the balances kept in an encrypted form are updated homomorphically. Moreover, Zether and Quisquis share the use of proof systems such as Bulletproofs [4] for *range proofs* and Σ-Protocols to prove algebraic statements under the DLOG assumption. Other solutions rely on proof systems for more general computations inspired by Zerocash, examples are Blockmaze [14], which has designed a specific *private computation circuit* for each transaction, and ZETH [20] that has developed the logic of UTXO on top of the account model using zk-SNARK to prove the transaction correctness.

Research question. Complex smart contracts can have more than two users, opening scenarios for a new model of transaction taking place amongst a set of participants. A similar concept is the "redistribution of wealth" introduced in Quisquis [11], where multiple account balances can be updated at the same time as the effect of one UTXO transaction. However, there are few proposals for CTs in the standard account model, and none consider such an extended payment model. Hence, we want to ask the question: *how can we design a zero-knowledge transfer in the account model, where there are multiple payees in one transaction?* Rather than having the traditional payment model, we introduce our concept of *Multi-Transfer* (MT) in CTs, i.e., a transfer that involves multiple payees with a single confidential transaction with hidden amounts and balances. We first formulate the statements over an MT zero-knowledge relation for which our proof is constructed, and then we design our zero-knowledge proof system, called *MTproof*, satisfying that relation. Inspired by the Σ-Bullets of Zether [3], our system is based on homomorphic primitives and combines two well-known proof systems: the *Aggregated Bulletproofs* [4] to construct one range proof of m aggregated values and several Σ-protocols [6] to prove statements on the encryption of multiple transfer amounts together with the balance

of the sender. Since the range proof is the most expensive component of the proof system even for a single value, it turns out that a single range proof for m aggregated values is more efficient.

Our contribution. In this paper, we initiate the formalization of the *Multi-Transfer* (MT) in the account model. We design the *MTproof* proof system for the corresponding MT relation, which is provided by the following properties: (i) Multi-Transfer - multiple payees receive funds from a single transaction; (ii) Confidential Transfers - hide the amounts in the transaction and sender/recipients balances values; (iii) Zero-Knowledge - verifier learns nothing about the secrets and witnesses of the transaction; (iv) Non-Interactive - can be transformed into a non-interactive version, i.e., no interactions from the verifier to the prover; (v) Trustless - no added trust derived from the proof systems or trusted executors; (vi) Aggregation - single aggregate proof for many statements related to the MT relation. As part of our contribution, we implement the *MTproof* proof system using `arkworks` [2] Rust ecosystem. We evaluate the *MTproof* concretely in terms of proof size, prover and verifier execution time.

2 Preliminaries

In what follows, we denote with pp the public parameters and with λ the security parameter of a scheme. We write $a \xleftarrow{\$} \mathcal{S}$ when a random variable a is uniformly sampled from a set $\mathcal{S}$.

Groups. Let $\mathcal{G}$ be a *group-generation algorithm* that on input 1^λ outputs the tuple $(\mathbb{G}, p, g)$, where $\mathbb{G}$ is a description of a *cyclic group*, p is a prime number and is the order of the group and $g \in \mathbb{G}$ is a generator of the group. We consider groups in which the *discrete logarithm problem* is computationally intractable. We refer to the DLOG assumption and its variant of the Decisional Diffie-Hellman (DDH) assumption for groups in which the discrete logarithm problem is hard.

Pedersen commitments A Pedersen commitments is defined over a finite cyclic group $\mathbb{G}$ of prime order. The message space is from the set of integers modulo p, $\mathbb{Z}_p$ where p is a safe prime. Let $g, h \in \mathbb{G}$ be two random public generators, $m \in \mathbb{Z}_p$ the message and $r \xleftarrow{\$} \mathbb{Z}_p^*$ the blinding factor, the commitment is defined as: com = Commit(m;r) = $(g^m h^r) \in \mathbb{G}$. We consider a *non-interactive* commitment scheme in which a valid commitment is proved by the knowledge of the secret relating to the commitment. A variant of this scheme is *Pedersen vector commitments* used to commit multiple messages at once. The input messages are gathered in a vector and a vector of generators is defined as well.

ElGamal homomorphic encryption. Considering a cyclic group $\mathbb{G}$ of prime order p and generator g of that group, the encryption scheme has a private key in the random integer $x \xleftarrow{\$} \{1, \ldots, p-1\}$ and a public key of the form $y = g^x$, then the public parameters of the scheme are (G, p, g, y). Let r be a randomness uniformly sampled in $r \xleftarrow{\$} \{1, \ldots, p-1\}$, the ciphertext for a message m is: $Enc(m) = (C_L, C_R) =$

$(m \cdot y^r, g^r)$. A more specific use case of the scheme can be considered as in Zether [3]. Here, the message m to encrypt is an integer value $a \in \mathbb{Z}_p$ which can be turned into a group element using the simple mapping $m = g^a$. Hence, the encryption function can be rewritten as $Enc(m) = (C_L, C_R) = (m \cdot y^r, g^r) = (g^a \cdot y^r, g^r)$. An *additive homomorphism* can be derived from the application of the group operator $\cdot$ between two ElGamal ciphertexts: $(C_L \cdot C'_L = g^{a+a'} \cdot y^{r+r'}, C_R \cdot C'_R = g^{r+r'})$.

Zero-Knowledge Proofs. Let $\mathcal{R}$ be a binary relation for an instance x and a witness w and $\mathcal{L}$ the corresponding language such that $\mathcal{L} = \{x \mid \exists w : (x,w) \in \mathcal{R}\}$. An interactive proof is a protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ in which $\mathcal{P}$ tries to convince $\mathcal{V}$ that an instance x is in the language $\mathcal{L}$ for the given relation $\mathcal{R}$. This can be done through an interactive exchange of messages between $\mathcal{P}$ and $\mathcal{V}$ representing the *transcript*, from which the verifier can accept or reject the conversation (namely the proof). The proof is said to be *zero-knowledge* if it essentially reveals nothing beyond the validity of the proof. An interactive proof is *honest-verifier perfect zero-knowledge* (HVZK) if it has *perfect completeness*, *special soundness* and *honest-verifier perfect zero-knowledge* properties. The HVZK protocol is called *public coin* if all the verifier's challenges sent to the prover are chosen uniformly at random and are independent of the prover's messages.

Σ*-Protocols*. HVZK public-coin interactive proofs consisting of three messages (a,c,z) where: a is the *announcement* computed by $\mathcal{P}$ and sent to $\mathcal{V}$, c is the *challenge* randomly sampled by $\mathcal{V}$ and sent to $\mathcal{P}$ and z is the *response* computed by $\mathcal{P}$ based on the challenge and sent back to $\mathcal{V}$. From that structure, one can prove that ciphertexts are well-formed, the knowledge of the opening of a commitment, the knowledge of a secret behind encryption, or other statements relating to the discrete logarithm. Moreover, any interactive zero-knowledge proof can be transformed in a *non-interactive* zero-knowledge proof (NIZK) by means of the Fiat-Shamir heuristic [12]. In such case, an honest prover tries to follow the protocol composed of the messages (a,c,z) where the challenge c is replaced by a *random oracle*.

Zero-knowledge relation. The relation is a valid collection of instances-witnesses together with the statements for which the zero-knowledge proof is constructed. We use the notation $Rel : \{(x_1,...,x_n \ ; \ w_1,...,w_m) : f(x_1,...,x_n,w_1,...,w_m)\}$ to specify that prover and verifier know the public instances $x_1,...,x_n$, and only the prover knows the witnesses $w_1,...,w_m$ such that $f(x_1,...,x_n,w_1,...,w_m)$ is true. f defines the statements for the given instances-witnesses and can be expressed in algebraic form.

2.1 Zero-knowledge Multi-Transfer relation

The concept of *Multi-Transfer* in the confidential setting has been previously presented in our work *ZeroMT* [8]. The aim is to enable a single payer to transfer currency to multiple payees within a single confidential transaction that has the effect of multiple concurrent transfers. The *ZeroMT* transaction layer builds upon a user program, defined as *MTU*, and a smart contract hosted on the account-model blockchain, the *MTSC* smart contract. Users can utilise the MTU to initiate the multi-transfer

transaction, the *MTX* transaction, towards n recipients. Each involved party i has an encrypted balance of the form $b[y_i] = (C_L = g^{b_i} \cdot y_i^r, C_R = g^r)$ obtained by means of an ElGamal public key $y_i = g^{sk_i} \in \mathbb{G}$, derived from a randomly sampled secret key $sk_i \xleftarrow{\$} \mathbb{Z}_p$. Given a list $\mathbf{a} = (a_1, \ldots, a_n)$ of plaintext amounts to be transferred, what the MTU does is encrypting those values with the public key y of the sender and the public keys $\bar{\mathbf{y}} = (\bar{y}_1, \ldots, \bar{y}_n)$ of the recipients. The result are respectively the lists $\mathbf{C} = (C_1, \ldots, C_n)$ and $\bar{\mathbf{C}} = (\bar{C}_1, \ldots, \bar{C}_n)$, where $\forall i \in [1,n]$, $\text{Enc}_y(a_i) = C_i = g^{a_i} y^r$ and $\text{Enc}_{\bar{y}_i}(a_i) = \bar{C}_i = g^{a_i} \bar{y}_i^r$. The two lists of ciphertexts are associated with the same randomness value $D = g^r$, where $r \xleftarrow{\$} \mathbb{Z}_p^*$ is a value uniformly sampled from the set of inverses in $\mathbb{Z}_p$. In order to initiate a transaction, the MTU forwards the $\mathbf{C}$ and $\bar{\mathbf{C}}$ lists to the MTSC along with a zero-knowledge proof π satisfying a *zero-knowledge multi-transfer relation*, which (informally) states that: (i) the sender knows the secret key sk for which the respective public key y encrypts the values in $\mathbf{C}$; (ii) the sender knows the randomness r used in the encryption process; (iii) the sender balance cannot be overdraft; (iv) the i-th ciphertexts in both $\mathbf{C}$ and $\bar{\mathbf{C}}$ are well-formed and encrypt the same amounts; (v) each of the transfer amounts in $\mathbf{a}$ is non-negative; (vi) the sender remaining balance b' is non-negative. For these statements, the MTU acts as a prover and the MTSC as a verifier. Both share the public instances $(y, \bar{\mathbf{y}}, C_L, C_R, \mathbf{C}, \bar{\mathbf{C}}, D, g)$, while the witnesses $(sk, \mathbf{a}, b', r)$ are known only by the prover. Further, we express the statement for the range proofs in which we prove that $\forall a_i \in (a_1, \ldots, a_n)$ each transfer amount a_i and the remaining sender's balance b' fall within the range of admissible values $[0, MAX]$, where MAX is the upper limit equal to $2^n - 1$ with n the bit length of the values. After the proof π is successfully verified by the MTSC, all the requested transfers are executed. The updates on the balances of the sender and each of the recipients are made through the additive homomorphism of the ElGamal scheme on the underlying group $\mathbb{G}$: $b[y] = b[y] \circ (C_{tot}^{-1}, D^{-1})$ where $b[y]$ is the reference of the sender's balance, and $\forall i \in [1,n]$, $b[\bar{y}_i] = b[\bar{y}_i] \circ (\bar{C}_i, D)$ is the i-th receiver's balance increased by the corresponding amount in the $\bar{\mathbf{C}}$ list.

3 MTproof: Multi-Transfer zero-knowledge proof system

In our context of the previous section, a zero-knowledge proof must guarantee that the MTX transaction is well-formed and legit, i.e., the balance transfer goes to the right payee and the payer has enough money to spend in his/her wallet. To this end, we design the interactive zero-knowledge proof system, called *MTproof*, satisfying the *multi-transfer relation*. The main idea is to combine the *Aggregated Inner Product Range Proof* and several $\Sigma-$Protocols as follows. We generate one aggregated range proof for $m = n+1$ values, such that the $m-1$ values correspond to each transfer amount a_i and one value b' to the remaining balance; we use $\Sigma-$Protocol to prove the statements for the two lists of n encrypted transfer amounts $(C_i, D)_{i=1}^n$ and $(\bar{C}_i, D)_{i=1}^n$ and for the sender's remaining balance. Using the above combination of proof systems, we generalise the Σ-Bullets protocol [3] of Zether to the case of n transfers

per epoch instead of one. Our proof system benefits from short and logarithmic-sized proofs as well as trustless property. We organize this section as follows: we present the key concepts of the Bulletproofs theory; we show how we modify Bulletproofs in our MTproof proof system; we present the Σ-protocols that compose the remaining part of MTproof.

Bulletproofs review. First, we summarise some notations of Bulletproofs [4], used in our proof system. $\mathbb{Z}_p$ is the ring of integers modulo p prime ($\mathbb{Z}_p^*$ is $\mathbb{Z}_p/\{0\}$). g and h are generators of a cyclic group $\mathbb{G}$ of prime order p. Capitalised are *commitments* and Greek letters are *blinding factors*, e.g., $A = g^a \cdot h^\alpha$ is a *Pedersen commitment* to the value a with blinding factor α. In bold are vectors, e.g., $\mathbf{a} \in \mathbb{Z}_p^n$ is a vector with elements in $\mathbb{Z}_p$ of dimension n. The *inner-product* of two vectors having size n is $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^{n} a_i \cdot b_i$. The *Hadamard product* of two vectors having size n is $\mathbf{a} \circ \mathbf{b} = (a_1 \cdot b_1, ..., a_n \cdot b_n)$. The following is useful for *Pedersen vector commitments*: let $\mathbf{g} = (g_1, ..., g_n) \in \mathbb{G}^n$ be a vector of generators then $A = \mathbf{g}^\mathbf{a} = \prod_{i=1}^{n} g_i^{a_i}$ is *binding* (but not *hiding*) commitment to the vector $\mathbf{a} \in \mathbb{Z}_p^n$. Slices of vectors are denoted with $\mathbf{a}_{[:k]} = (a_1, ..., a_k) \in \mathbb{F}^k$ and $\mathbf{a}_{[k:]} = (a_{k+1}, ..., a_n) \in \mathbb{F}^{n-k}$. A vector polynomial $p(X) \in \mathbb{Z}_p^n[X]$ is defined as $p(X) = \sum_{i=0}^{d} \mathbf{p_i} \cdot X^i$, where $\mathbf{p_i} \in \mathbb{Z}_p^n$ and d is the degree of the polynomial. For a complete description, refer to section 2.3 of [4]. With Bulletproofs, a prover can convince a verifier that a value v is in range, in particular, $0 \leq v < 2^n$. Given as public parameters the generators g and h and the Pedersen commitment $V = h^\gamma g^v$ of the value v using randomness γ, the system ends up proving that the inner-product of two committed vectors $\mathbf{l}$, $\mathbf{r}$ is a certain value $\hat{t}$, then the equality $\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle$ is valid if and only if $v \in [0, 2^n - 1]$. This is the final step of the range proof in which the prover and verifier engage in the *Inner-Product Argument* (IPA) protocol. We now present a high-level overview of the previous steps. The prover first creates two commitments, A commitment to the vectors $\mathbf{a}_L$ and $\mathbf{a}_R$ where $\langle \mathbf{a}_L, \mathbf{2}^n \rangle = v$ and $\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^n$, S commitment to blinding terms $\mathbf{s}_L$ and $\mathbf{s}_R$. The prover receives from the verifier two challenge points y and z and creates T_1 and T_2 commitments to the coefficients t_1 and t_2 of the polynomial $t(X)$. The prover honestly constructs the polynomial $t(X)$ from the inner product of two vector polynomials $l(X)$ and $r(X)$, which in turn have a special form derived from a linear combination of the vectors $\mathbf{a}_L$ and $\mathbf{a}_R$ respectively and the verifier's challenges. The prover does not commit to the zero-coefficient t_0 of $t(x)$ and sends to the verifier only $T_{1,2}$. Instead, the prover proves the opening of the commitments, sending the evaluation $\hat{t} = t(x)$ at a random point x from the verifier. The verifier can calculate the zero-coefficient himself from the commitment V of the value v and the challenges. After receiving the challenge x, together with $\hat{t}$ the prover also sends to the verifier a blinding value τ_x for $\hat{t}$, the two blinded vectors $\mathbf{l} = l(x)$ and $\mathbf{r} = r(x)$ and a blinding factor μ for the commitments A and S. Now the verifier can verify the Pedersen commitment V of the value v, that A and S are valid and that $\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle$ is correct. Transmitting $\mathbf{l}$ and $\mathbf{r}$ has a linear cost in n (bits of ranges). Using the IPA protocol with $\mathbf{l}$ and $\mathbf{r}$ becoming witnesses and some adjustments explained in section 4.2 of [4], we obtain a logarithmic proof size in n. Moreover, an *aggregated range proof* can be used to perform one proof for m values. This implies some modifications, in particular the prover computes $\mathbf{a}_L \in \mathbb{Z}_p^{m \cdot n}$ such that $\langle \mathbf{2}^n, \mathbf{a}_{L[(j-1)\cdot n : j \cdot n - 1]} \rangle = v_j$ for all $j \in [1, m]$. The prover modifies the constructions

of $l(X)$ and $r(X)$ such that they stay in $\mathbb{Z}_p^{m\cdot n}[X]$ and τ_x to include the randomness of each V_j commitment of the v_j value. Finally, verifier modifies the verification of $\hat{t}$ to include all the V_j commitments and the verification of A, S commitments respect to the new $\mathbf{r} \in \mathbb{Z}_p^{m\cdot n}$.

Using Bulletproofs in MTproof. We now outline the modifications of the Bulletproofs protocol required in *MPproof*, in order to prove that each transfer amount a_i of the vector $\mathbf{a} = (a_1, \ldots, a_n)$ is non-negative and the sender remaining balance b' after the transfer is also non-negative. More specifically, we generate an aggregated range proof valid for a vector of m values, where the first value is b' and the $m-1$ remaining values are those in $\mathbf{a}$. The range domain within which these values must be proved valid is $[0, 2^n - 1]$. The protocol is initiated by the prover creating the commitments $A = h^{\alpha} \cdot \mathbf{g}^{\mathbf{a}_L} \cdot \mathbf{h}^{\mathbf{a}_R} \in \mathbb{G}$ and $S = h^{\rho} \cdot \mathbf{g}^{\mathbf{s}_L} \cdot \mathbf{h}^{\mathbf{s}_R} \in \mathbb{G}$, where $\mathbf{g}, \mathbf{h}$ are vectors of generators, $h \in \mathbb{G}$ is the blinding generator and α, ρ the blinding random values. The commitment A to the vectors $\mathbf{a}_L$ and $\mathbf{a}_R$ is generated from the binary representation of b' and $\mathbf{a}$, respectively: $\langle \mathbf{a}_{L[:n]}, \mathbf{2}^n \rangle = b'$ and $\langle \mathbf{a}_{L[(j-1)\cdot n:j\cdot n]}, \mathbf{2}^n \rangle = a_{j-1}$ $\forall j \in [2, m]$. Then, by subtracting from $\mathbf{a}_L$ the vector of powers of 1 we obtain: $\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^{m\cdot n} \in \{0, -1\}^{m\cdot n}$. The commitment S is a commitment to the randomly sampled blinding terms in $\mathbf{s}_L, \mathbf{s}_R \in \mathbb{Z}_p^{m\cdot n}$. The prover then sends these commitments to the verifier, which responds with the random challenges y, z. The prover then constructs the commitments: $T_1 = g^{t_1} \cdot h^{\tau_1} \in \mathbb{G}$ and $T_2 = g^{t_2} \cdot h^{\tau_2} \in \mathbb{G}$. Both commitments consist of two uniformly random values $\tau_1, \tau_2 \in \mathbb{Z}_p$ and the coefficients t_1, t_2 of the polynomial $t(X) = t_0 + t_1 \cdot X + t_2 \cdot X^2 \in \mathbb{Z}_p[X]$. Such $t(X)$ polynomial is derived from the inner product of two polynomials $l(X), r(X) \in \mathbb{Z}_p^{m\cdot n}[X]$, both built from a linear combination of the challenges of the verifier and the vectors $\mathbf{a}_L, \mathbf{a}_R, \mathbf{s}_L, \mathbf{s}_R$. After receiving the commitments T_1 and T_2 from the prover, the verifier responds with the randomly sampled challenge x. The prover calculates the polynomial evaluation $\hat{t} = t(x)$, and two blinding factors, one for $\hat{t}$ and one for the commitments A and S, respectively: $\tau_x = \tau_1 \cdot x + \tau_2 \cdot x^2 \in \mathbb{Z}_p$ and $\mu = \alpha + \rho \cdot x \in \mathbb{Z}_p$. Simply replacing Pedersen commitment with ElGamal encryption is not sufficient at this point. In Bulletproofs, the equality to be verified, which includes commitments V, T_1 and T_2, exploits the additive homomorphic property of the Pedersen commitment, and this cannot be done with ElGamal encryptions under different keys. We follow the same intuition of Zether to combine Bulletproofs with a Σ-Protocol: instead of giving to the verifier the opening of polynomial commitments to $t(X)$, knowledge of the opening is proved. This corresponds to proving knowledge of the blinding value τ_x for the commitments $T_{1,2}$. To include this statement, our proof system now generates a proof for the following *zero-knowledge multi-transfer relation* (formally):

$$
\begin{aligned}
Rel_{ConfMultiTransfer} : \{\ (y, \bar{\mathbf{y}}, C_L, C_R, \mathbf{C}, \bar{\mathbf{C}}, D, g, z, \hat{t}, \delta(y,z);\ sk, \mathbf{a}, b', r, \tau_x) :\\
(C_i = g^{a_i} y^r \wedge \bar{C}_i = g^{a_i} \bar{y}_i^r \wedge D = g^r)_{i=1}^n \wedge\\
C_L / \prod_{i=1}^{n} C_i = g^{b'} (C_R / \prod_{i=1}^{n} D)^{sk} \wedge y = g^{sk} \wedge\\
g^{\hat{t} - \delta(y,z) - b' \cdot z^2 - \sum_{i=1}^{n} a_i \cdot z^{2+i}} h^{\tau_x} = T_{1,2} \}
\end{aligned}
\tag{1}
$$

Hence, the prover also calculates a blinding commitment $A_t = g^{-k_{ab}} \cdot h^{k_\tau}$, where k_{ab} and k_τ are two uniformly random scalars. At this point, the prover sends the values $\hat{t}, \mu$ and A_t to the verifier, and after receiving the random challenge c from the verifier, the prover generates and sends the scalars s_{ab} and s_τ such that: $s_{ab} = k_{ab} + c \cdot (b' \cdot z^2 + \sum_{i=1}^{m-1} a_i \cdot z^{2+i}) \in \mathbb{Z}_p$ and $s_\tau = k_\tau + c \cdot \tau_x \in \mathbb{Z}_p$. The verifier uses them in the final proof of knowledge check: $g^{c(\hat{t}-\delta(y,z))-s_{ab}} \cdot h^{s_\tau} \stackrel{?}{=} A_t T_1^{c \cdot x} T_2^{c \cdot x^2}$ where $\delta(y,z) = (z-z^2) \cdot \langle \mathbf{1}^{m \cdot n}, \mathbf{y}^{m \cdot n} \rangle - \sum_{j=1}^{m} z^{2+j} \cdot \langle \mathbf{1}^{m \cdot n}, \mathbf{2}^{m \cdot n} \rangle$. From this point, *MTproof* follows with the IPA protocol to prove that the inner product $\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle$ of the committed vectors $\mathbf{l}$ and $\mathbf{r}$ is valid. We use the logarithmic-sized proof optimization and the verifier *multi-exponentiation* technique from the IPA protocol of Bulletproofs. Hence, the prover and verifier engage in the IPA protocol on the inputs $(\mathbf{g}, \mathbf{h}', Ph^{-\mu}, \hat{t}; \mathbf{l}, \mathbf{r})$, where $\mathbf{h}' = (h_1, h_2^{y^{-1}}, h_3^{y^{-2}}, ..., h_{m \cdot n}^{y^{-m \cdot n+1}})$ and $P = AS^x \cdot \mathbf{g}^{-z} \cdot \mathbf{h}'^{z \cdot \mathbf{y}^{m \cdot n}} \cdot \prod_{j=1}^{m} \mathbf{h}'^{z^{1+j} \cdot \mathbf{2}^n}_{[(j-1) \cdot n : j \cdot n]}$.

Using Σ-Protocol in MTproof. We present the remaining part of *MTproof* relating to the Σ-protocols. In particular, we design four Σ-proofs, each for one of the equalities in conjunction in relation (1), except for the last one (the Bulletproofs proof of knowledge of the opening of the polynomial commitment). In the following, we parse the statements and we show the corresponding Σ-proof.

Σ-proof-`sk` for statement: the sender knows a secret key sk for which the respective public key y encrypts the values in $\mathbf{C}$ and the public key is well-formed

---

1: **input**: $(g, y \in \mathbb{G};\ sk \in \mathbb{Z}_p) : y = g^{sk}$
2: $\mathcal{P}$'s input : (g, sk)
3: $\mathcal{V}$'s input : (g, y)
4: **output**: $\{\mathcal{V}$ accepts or $\mathcal{V}$ rejects$\}$
5: $\mathcal{P} \rightarrow \mathcal{V} : A_y$ where $A_y = g^{k_{sk}} \in \mathbb{G}$ and $k_{sk} \stackrel{\$}{\leftarrow} \mathbb{Z}_p$
6: $\mathcal{V} \rightarrow \mathcal{P} : c \stackrel{\$}{\leftarrow} \mathbb{Z}_p$
7: $\mathcal{P} \rightarrow \mathcal{V} : s_{sk}$ where $s_{sk} = k_{sk} + c \cdot sk \in \mathbb{Z}_p$
8: $\mathcal{V} : g^{s_{sk}} \stackrel{?}{=} A_y y^c \in \mathbb{G}$ if yes, $\mathcal{V}$ accepts; otherwise rejects

---

Σ-proof-`r` for statement: the sender knows a randomness r used in the encryption

---

1: **input**: $(g, D \in \mathbb{G};\ r \in \mathbb{Z}_p) : D = g^r$
2: $\mathcal{P}$'s input : (g, r)
3: $\mathcal{V}$'s input : (g, D)
4: **output**: $\{\mathcal{V}$ accepts or $\mathcal{V}$ rejects$\}$
5: $\mathcal{P} \rightarrow \mathcal{V} : A_D$ where $A_D = g^{k_r} \in \mathbb{G}$ and $k_r \stackrel{\$}{\leftarrow} \mathbb{Z}_p$
6: $\mathcal{V} \rightarrow \mathcal{P} : c \stackrel{\$}{\leftarrow} \mathbb{Z}_p$
7: $\mathcal{P} \rightarrow \mathcal{V} : s_r$ where $s_r = k_r + c \cdot r \in \mathbb{Z}_p$
8: $\mathcal{V} : g^{s_r} \stackrel{?}{=} A_D D^c \in \mathbb{G}$ if yes, $\mathcal{V}$ accepts; otherwise rejects

---

Σ-proof-`ab` for statement: the sender balance cannot be overdraft, i.e., the sender remaining balance is equal to the subtraction between the sender balance and all of the $(m-1)$ transfer amounts in $\mathbf{C}$

---

input: $(g, D, C_L, C_R \in \mathbb{G},\ \mathbf{C} \in \mathbb{G}^{m-1};\ sk, b' \in \mathbb{Z}_p, \mathbf{a} \in \mathbb{Z}_p^{m-1})$:
$C_L / \prod_{i=1}^{m-1} C_i = g^{b'} (C_R / \prod_{i=1}^{m-1} D)^{sk}$
    $\mathcal{P}$'s input : $(g, D, C_R, sk, \mathbf{a}, b')$
    $\mathcal{V}$'s input : $(g, D, C_L, C_R, \mathbf{C})$
output: $\{\mathcal{V}$ accepts or $\mathcal{V}$ rejects$\}$
$\mathcal{V} \to \mathcal{P} : z \xleftarrow{\$} \mathbb{Z}_p^*$
$\mathcal{P}$ computes:
    $k_{sk}, k_{ab} \xleftarrow{\$} \mathbb{Z}_p$
    $A_{ab} = ((C_R / \prod_{i=1}^{m-1} D)^{z^2} \cdot \prod_{i=1}^{m-1} D^{z^{i+2}})^{k_{sk}} \cdot g^{k_{ab}} \in \mathbb{G}$
end $\mathcal{P}$
$\mathcal{P} \to \mathcal{V} : A_{ab}$
$\mathcal{V} \to \mathcal{P} : c \xleftarrow{\$} \mathbb{Z}_p$
$\mathcal{P}$ computes:
    $s_{sk} = k_{sk} + c \cdot sk \in \mathbb{Z}_p$
    $s_{ab} = k_{ab} + c \cdot (b' z^2 + \sum_{i=1}^{m-1} (a_i z^{i+2})) \in \mathbb{Z}_p$
end $\mathcal{P}$
$\mathcal{P} \to \mathcal{V} : s_{sk}, s_{ab}$
$\mathcal{V} : g^{s_{ab}} ((\frac{C_R}{\prod_{i=1}^{m-1} D})^{z^2} \cdot \prod_{i=1}^{m-1} D^{z^{2+i}})^{s_{sk}} \stackrel{?}{=} A_{ab} ((\frac{C_L}{\prod_{i=1}^{m-1} C_i})^{z^2} \cdot \prod_{i=1}^{m-1} C_i^{z^{2+i}})^c \in \mathbb{G}$
    if yes, $\mathcal{V}$ accepts; otherwise rejects

Σ-proof-$\mathtt{y}$ for statement: the i-th values in both $\mathbf{C}$ and $\bar{\mathbf{C}}$ are well-formed and correspond to the encryption of the i-th amount to be transferred

---

input: $(y \in \mathbb{G}, \bar{\mathbf{y}}, \mathbf{C}, \bar{\mathbf{C}} \in \mathbb{G}^{m-1};\ r \in \mathbb{Z}_p)$:
$(C_i = g^{a_i} y^r \wedge \bar{C}_i = g^{a_i} \bar{y}_i^r \wedge D = g^r)_{i=1}^{m-1}$
    $\mathcal{P}$'s input : $(y, \bar{\mathbf{y}}, r)$
    $\mathcal{V}$'s input : $(y, \bar{\mathbf{y}}, \mathbf{C}, \bar{\mathbf{C}})$
output: $\{\mathcal{V}$ accepts or $\mathcal{V}$ rejects$\}$
$\mathcal{P} \to \mathcal{V} : A_{\bar{y}}$ where $A_{\bar{y}} = \prod_{i=1}^{m-1} (y \cdot \bar{y}_i^{-1})^{k_r} \in \mathbb{G}$ and $k_r \xleftarrow{\$} \mathbb{Z}_p$
$\mathcal{V} \to \mathcal{P} : c \xleftarrow{\$} \mathbb{Z}_p$
$\mathcal{P} \to \mathcal{V} : s_r$ where $s_r = k_r + c \cdot r \in \mathbb{Z}_p$
$\mathcal{V} : \prod_{i=1}^{m-1} (y \cdot \bar{y}_i^{-1})^{s_r} \stackrel{?}{=} A_{\bar{y}} \cdot (\prod_{i=1}^{m-1} C_i / \bar{C}_i)^c$
    if yes, $\mathcal{V}$ accepts; otherwise rejects

---

4 MTproof implementation and evaluation

In this section, we evaluate *MTproof* using our code implementation in Rust and the `arkworks` [2] library suites. The source code of *MTproof* can be found on GitHub [10]. The elliptic curve, underlying all the operations on elements of $\mathbb{G}$, is the Barreto–Naehrig curve **BN-254**. The modular structure of *MTproof* allows us to conduct accurate and modular benchmarks to estimate each execution time for the prover

and verifier functions and the associated proofs' sizes. The results of the evaluations are shown in Table 1. The benchmarks are executed multiple times with different values for n and m, equal to powers of two. All the measurements are carried on a machine with an *Intel Core i7-10750H* (12 threads and 6 cores at 2.60GHz, with turbo frequencies at 5.00GHz) CPU and 16 GB of RAM, running the Rust compiler.

Table 1: *MTproof* evaluation results

n	m	Proving time (ms)	Verifying time (ms)	Proof size (bytes)
16	2	534	202	1,584
16	4	1,000	379	1,712
16	8	2,030	718	1,840
16	16	3,941	1,389	1,968
16	32	7,815	2,680	2,096
16	64	15,192	5,327	2,224
32	2	967	348	1,712
32	4	1,891	686	1,840
32	8	3,753	1,329	1,968
32	16	7,431	2,626	2,096
32	32	14,844	5,259	2,224
32	64	30,052	10,460	2,352
64	2	1,899	679	1,840
64	4	3,762	1,320	1,968
64	8	7,496	2,619	2,096
64	16	14,980	5,218	2,224
64	32	29,794	10,478	2,352
64	64	61,430	21,533	2,480

Comparison with concurrent work and results. We consider one concurrent work, Anonymous Zether in [9]. Anonymous Zether enhances the Basic Zether scheme by introducing the *many-out-of-many* primitive to build an anonymity set for the unlinkability of addresses. Proof size, proving time, and verifying time are provided with respect to the growth of the anonymity set. These evaluations are carried out by setting a fixed number of two aggregate 32-bit range values. Compared with *MTproof*, it is possible to notice the same logarithmic growth with the increase of m (aggregate values) and the increase of the anonymity set size, as shown in Fig. 1 for the proof size. Similar observation also applies to the proving and verifying times. Hence, for the considerations that follow, we consider our proof system evaluation as the lower limit $\Omega(log\ M)$, where $M = m * k$ with m the number of aggregate values and $n = k$ the bit range constant factor. The benchmark results in Table 1 highlight that as the number of aggregated values grows, hence m, the execution times and proof sizes become more and more convenient than those resulting from multiple and separate executions of the proving or verifying functions. Therefore, a consideration arises of how convenient it is to provide an aggregate proof for multiple values instead of providing proof for each of those values. For instance, considering a 64 bit range, the

time required to generate a proof for multiple aggregated values (from a number of 2 up to 64 values) is between 33,97% and 48,65% smaller than the time required to generate one proof for each of such values. In the same way, the time required to verify a proof for multiple aggregated values is between 35,20% and 49,66% smaller than the time required to verify the proofs individually. Furthermore, considering the same bit range, the size of an aggregated proof is between 64,35% and 97,86% smaller than all single proofs combined together.

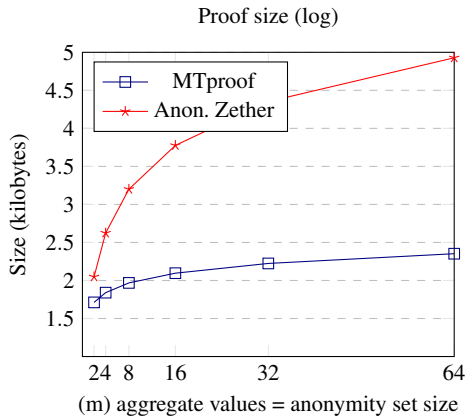

Fig. 1: Proof size comparison *MTproof* and *Anon. Zether*.

5 Conclusion and future work

MTproof is the zero-knowledge proof system aimed at realizing the Multi-Transfer, proposing aggregation, correctness and confidentiality in transactions. The system is based on Bulletproofs and Σ-protocol proof systems and has been implemented according to the tools of the research community. The evaluations highlight benefits from the aggregation, in terms of generation and verification times, and size of the proof. As future work, an optimization can be integrated into MTproof in its IPA component, which may lead to significant savings in execution time. Moreover, we will evaluate MTproof in real scenarios involving streams of sensor data [21, 16, 23].

Acknowledgment

We acknowledge the master student Francesco Pio Stelluti for his contribution to the codebase of MTproof.

References

1. K. M. Alonso et al. Zero to monero, 2020.
2. arkworks rs. arkworks.
3. B. Bünz, S. Agrawal, M. Zamani, and D. Boneh. Zether: Towards privacy in a smart contract world. In *International Conference on Financial Cryptography and Data Security*, pages 423–443. Springer, 2020.
4. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 315–334. IEEE, 2018.
5. V. Buterin et al. A next-generation smart contract and decentralized application platform.
6. D. Butler, D. Aspinall, and A. Gascón. On the formalisation of σ-protocols and commitment schemes. In *POST*, pages 175–196, 2019.
7. W. Chan and A. Olmsted. Ethereum transaction graph analysis. In *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 498–500. IEEE, 2017.
8. F. Corradini, L. Mostarda, and E. Scala. Zeromt: Multi-transfer protocol for enabling privacy in off-chain payments. In *International Conference on Advanced Information Networking and Applications*, pages 611–623. Springer, 2022.
9. B. E. Diamond. Many-out-of-many proofs and applications to anonymous zether. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1800–1817. IEEE, 2021.
10. EmanueleSc. Zeromt.
11. P. Fauzi, S. Meiklejohn, R. Mercer, and C. Orlandi. Quisquis: A new design for anonymous cryptocurrencies. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 649–678. Springer, 2019.
12. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*, pages 186–194. Springer, 1986.
13. M. Fleder, M. S. Kester, and S. Pillai. Bitcoin transaction graph analysis. *arXiv preprint arXiv:1502.01657*, 2015.
14. Z. Guan, Z. Wan, Y. Yang, Y. Zhou, and B. Huang. Blockmaze: An efficient privacy-preserving account-model blockchain based on zk-snarks. *IEEE Transactions on Dependable and Secure Computing*, 2020.
15. A. Jivanyan. Lelantus: Towards confidentiality and anonymity of blockchain transactions from standard assumptions. *IACR Cryptol. ePrint Arch.*, 2019:373, 2019.
16. N. Q. Mehmood, R. Culmone, and L. Mostarda. Modeling temporal aspects of sensor data for MongoDB NoSQL database. *Journal of Big Data*, 4(1), Mar. 2017.
17. S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, Manubot, 2019.
18. A. Poelstra. Mimblewimble. 2016.
19. D. Ron and A. Shamir. Quantitative analysis of the full bitcoin transaction graph. In *International Conference on Financial Cryptography and Data Security*, pages 6–24. Springer, 2013.
20. A. Rondelet and M. Zajac. Zeth: On integrating zerocash on ethereum. *arXiv preprint arXiv:1904.00905*, 2019.
21. G. Russello, L. Mostarda, and N. Dulay. A policy-based publish/subscribe middleware for sense-and-react applications. *Journal of Systems and Software*, 84(4):638–654, Apr. 2011.
22. E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE, 2014.
23. C. Vannucchi, M. Diamanti, G. Mazzante, D. Cacciagrano, R. Culmone, N. Gorogiannis, L. Mostarda, and F. Raimondi. Symbolic verification of event–condition–action rules in intelligent environments. *Journal of Reliable Intelligent Environments*, 3(2):117–130, Feb. 2017.