University of Camerino

School of Advanced Studies

Doctor of Philosophy in Computer Science and Mathematics - XXXVI Cycle

Trustless zero-knowledge proofs in public blockchain

Supervisor

Prof. Leonardo Mostarda

PhD Candidate

Emanuele Scala

Academic Year 2023-2024

ABSTRACT

The public blockchain lacks user privacy due to the possibility of linking transactions and disclosing related information. 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. ZK-proofs need to be *trustless*, i.e., zero-knowledge systems that do not introduce additional trust are required. Moreover, CTs do not fully exploit the potential of ZK-proofs, since each transaction comes with one or more proofs for a single transfer. This leads to higher blockchain space when the proofs are stored in blocks, and higher fees when the proofs are verified by blockchain maintainers. On the other hand, the account model is the most flexible for addressing security challenges.

As the first contribution of this thesis, we present *ZeroMT*, a novel *multi-transfer* private payment scheme for account-based blockchains. Our approach develops 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 our cryptographic scheme in the extended model provides privacy for users and preserves the security requirements of CTs solutions. We provide an implementation and evaluation of ZeroMT, showing the benefits of aggregating multiple transfers in terms of transaction cost savings. Despite the trustless nature of our ZK-proofs, a computationally expensive sub-component is highlighted, that is the *Inner-Product Argument* (IPA). In the final part of this thesis, we reconcile the IPA with the new *compressed* Σ*-Protocol* theory and propose the *compressed* Σ*-IPA*. Our new argument system achieves computational costs reduced by a factor of 2 compared to the original IPA.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to all those who have supported me throughout my doctoral journey.

First, I would like to thank my supervisor, Prof. Leonardo Mostarda, for his guidance and support. His expertise has been instrumental in shaping my research and helping me grow.

I am grateful to the evaluators of my dissertation, Prof. Daniele Bartoli and Prof. Roberto Tonelli, for their constructive comments.

I am also deeply grateful to Prof. Changyu Dong, who hosted me during my period abroad at Newcastle University. His valuable guidance, support, and contributions have significantly enriched this thesis and my overall research experience.

Special thanks to my family.

CONTENTS

List of Figures **xi**

List of Tables **xiii**

I Introduction **1**

1 Introduction & contribution **3**
1.1 Blockchain technology 3
1.2 Confidential transactions 5
1.3 Contribution 6
1.4 Publications 7

II Background **11**

2 Preliminaries **13**
2.1 Notations 13
2.2 Basic Group Theory 13
2.2.1 Prime numbers and modular arithmetic 14
2.2.2 Groups 15
2.2.3 Cyclic groups 17
2.3 Provable Security 17
2.3.1 Probability 18
2.3.2 Computational security 20
2.3.3 Security experiments 21
2.3.4 Discrete logarithm assumptions 22
2.4 Elliptic Curves 24
2.4.1 Fields and rings 24

2.4.2 Elliptic curve groups 25
2.4.3 Elliptic curves in practice 27
2.5 Cryptographic Primitives 28
2.5.1 Commitment schemes 28
2.5.2 Homomorphic cryptosystems 30
2.6 Proof Systems 31
2.6.1 Interactive proof systems 32
2.6.2 Interactive Zero-Knowledge 34
2.7 Interactive Protocols 36
2.7.1 Schnorr's protocol 36
2.7.2 Σ-Protocols 39
2.7.3 Σ-Protocols security: special-soundness 41
2.7.4 Σ-Protocols security: zero-knowledge 44
2.8 Bulletproofs and Compressed Σ-Protocols 45
2.8.1 Bulletproofs for range proofs 45
2.8.2 Inner-product argument system 46
2.8.3 Compressed Σ-protocols 48

III ZeroMT and Compressed Σ-IPA 51

3 ZeroMT: ZK-proofs based Multi-Transfer scheme for account model blockchain 53
3.1 Introduction 53
3.1.1 New directions of CT protocols: multi-transfer transactions 56
3.1.2 Our contribution 58
3.1.3 Related works 59
3.1.4 Motivation scenario 61
3.2 Multi-Transfer Scheme 64
3.2.1 Background on Zether 64
3.2.2 ZK-relation and multi-transfer scheme 67
3.2.3 Security analysis 70
3.2.4 Zero-knowledge multi-transfer proof system 73
3.2.5 Security reductions 86
3.3 Implementation and Evaluation 89
3.3.1 Code implementation and benchmarks 90
3.3.2 Comparison with concurrent works 91
3.3.3 Ethereum smart contract gas estimation 92
3.3.4 Aggregation savings 94
3.4 Applications and Limitations 95
3.4.1 ZeroMT applications 95
3.4.2 Limitations and discussions 96

4 Σ-IPA: Improved Zero-Knowledge Proofs **99**
4.1 Introduction 99
4.1.1 Our contribution 100
4.1.2 Related works 101
4.2 Σ-Inner-Product Argument Protocol 102
4.2.1 Σ-inner-product with constant rounds 102
4.2.2 Compressed Σ-inner-product with LOG rounds 105
4.2.3 Efficiency analysis 107
4.2.4 Further optimizations and discussions 107

IV Conclusions **109**

5 Conclusions & Future Works **111**

Bibliography **113**

A Inner-Product Argument system **121**
A.1 IPA first protocol 121
A.2 IPA second recursive protocol 122
A.3 IPA with multi-exponentiation 123

LIST OF FIGURES

3.1 general smart lock 62
3.2 multi-transfer transaction flow 64
3.3 execution times when the number of transfers increases 90

LIST OF TABLES

2.1 proof system notations and operations 14
2.2 bulletproofs protocol messages 46

3.1 comparison of CT solutions 61
3.2 evaluation of multi-transfer ZK-proof 91
3.3 comparison with concurrent works 92
3.4 required gas amount per operation (units) 93
3.5 gas consumption for ZeroMT verifier 93
3.6 aggregation savings (prover and verifier execution times, proof size) 94

4.1 proposals for improvement of the IPA 102

PART I

INTRODUCTION

CHAPTER 1

INTRODUCTION & CONTRIBUTION

1.1 Blockchain technology

With the rise of the first digital currency of Bitcoin [54], blockchain technology has become a growing active field of study in both academia and industry. A blockchain, also known as a *decentralized ledger*, is essentially an immutable way of storing data in a distributed and append-only fashion. All nodes in the blockchain network will have the same copy of data thanks to the *consensus mechanism*. Moreover, the blockchain enables interactions, exchanges or transfers of something between the network members by means of *transactions* stored in form of blocks and with a strong timestamp. Each block contains a limited number of transactions, the hash of the current block and the hash of the previous block, along with other information. By hashing a block of transactions and widely publishing the hash, the timestamp proves that the data must have existed at that time to be included in the hash. Each timestamp includes the previous timestamp in its hash, forming a chain where each subsequent timestamp reinforces the preceding ones. This ensures that an attacker cannot tamper the data within blocks without having to modify the entire history of the chain.

In summary, the main strenghts of a blockchain system from an high-level perspective are: *decentralization* and *robustness*. Decentralization is achieved by eliminating the need for intermediaries, i.e., some centralized systems or organizations that process user data and ensure validity and authorization mechanisms. Instead, blockchain transactions contain their own proof of validity and authorization, allowing information to be shared across mutually

untrusted nodes. Furthermore, the consensus mechanism guarantees that the transaction issuer has evidence that the majority of nodes agree on the exact time the transaction was received. Hence, blockchain nodes agree on the order in which transactions are processed and on a single transaction history. To accomplish this, it is also necessary to be aware of all transactions and therefore they must be publicly announced during the consensus phase. Summing up, blockchain decentralization is a key factor that replaces the need for centralized parties with a distributed database, secured by automated mechanisms that are generally less prone to human error.

About robustness, the blockchain has *high availability* which stems from its built-in redundancy and from the state machine replication approach (every node processes every transaction). In addition, the blockchain is extremely *fault tolerant*: nodes which go down can always recover lost transactions and nodes can be freely added or removed from the network.

The blockchain was originally conceived for financial applications, i.e., payment systems based on cryptocurrencies. With the rise of Ethereum [24], the blockchain has adapted to other contexts thanks to the support of *smart contracts*. These are special programs equipped on each node of the blockchain, through which a transaction protocol can be designed to facilitate, verify and digitally apply the terms of an agreement, between one or more parties. Finally, the execution of smart contracts requires charging *fees* to the requester and produces effects (e.g., state changes) that are stored in the blockchain.

However, the public nature of the blockchain, and therefore the capability to allow anyone to join the network, validate transactions, and contribute to the consensus process, brings with it the well-known challenge of **privacy**. This challenge is mainly evident in contexts where the confidentiality of transaction information is an important requirement. A possible approach to mitigate the privacy problem is to restrict the blockchain access to a set of previously identified and authorized peers, landing into the permissioned blockchain model [4, 29]. However, this blockchain model leads to centralized systems and it is not suitable for large scale networks.

Remaining in the context of public and permissionless blockchains, the first thing to consider when designing privacy is the choice between the two main balance representations: Bitcoin UTXO (unspent transaction outputs) or Ethereum account model. In the UTXO model, a user that has coins owns a bunch of transaction outputs. To spend coins, current owners issue transactions that take one or more old outputs as inputs and designate new outputs to recipients. Once consumed, old outputs cannot be spent again. The sum of the amounts for each unspent output belonging to a user is considered the balance of that user. On the other side, account model is the balance strategy tailored for smart contracts. This model features two

types of accounts, user and contract account, each with its own balance and address. Account information has its representation in a global state (world state), where balances are expressed as the quantities (or a sub-unit) of coins owned by the address. When a user, say Alice, wants to make a transfer of funds to another user, say Bob, she essentially initiates a transaction that will affect the world state such that Alice's account balance must be debited and the Bob's account balance must be credited. For the transaction to be valid, the payer must have enough money in his balance and the funds must come from his balance.

1.2 Confidential transactions

While consensus mechanisms ensure data integrity and availability, the public announcement of transactions makes transaction information visible to all. Such information can reveal the blockchain's current state, updates requested by transactions, and digital signatures that prove the origin of transactions. This can be a risk factor for those applications where confidential data must have privacy protection and anonymity for the parties must not be compromised. Further, well-know attacks that rely on transaction graph analysis can reveal users' real-world identity and transaction information [28, 41, 59, 5]. In light of this, the lack of privacy in public blockchain has received huge attention from researchers in the last years. Following a recent major advance in cryptography that makes Zero-Knowledge proofs (ZK-proofs) applicable, one of the leading privacy applications emerges from the literature: *Confidential Transactions* (CTs). Indeed, ZK-proofs are capable of revealing nothing except the validity of the statements being proved relating to the transactions. This paradigm allows transactions verification without compromising sensitive information.

A notable example of CT protocol is Zerocash [62] from which many other protocols follow [56, 47, 3]. Such proposals are designed on the Bitcoin UTXO 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). However, the privacy solutions surrounding the UTXO model require changes to the underlying blockchain.

On another side, privacy protocols that move to the Ethereum account model have a flexible way to add privacy requirements. Indeed, smart contracts, i.e., decentralized programs running on-chain, allow the development of cryptographic protocols without altering the underlying blockchain. A notable smart contract based CT protocol in the account model is Zether [20]. Zether relies on ZK-proofs to prove the validity of transactions and keeps

encrypted balances for users that are homomorphically updatable. Moreover, Zether outlines the important requirement of not introducing a source of trust in the CT protocol, shifting focus to the **trustless** ZK-proofs. On the contrary, previous proposals are mostly based on trusted ZK-proofs, i.e., those that introduce a trusted third-party to generate the *Common Reference String* (CRS) or a computationally expensive pre-processing phase (e.g., multi-party ceremonies).

However, the trade-off of gaining a trustless protocol results in worse scalability and higher fees in smart contracts. This is because trustless ZK-proofs have a non-constant proof size and linear time verification. Moreover, 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 [38], where multiple account balances can be updated at the same time as the effect of one transaction. We identify such a payment model with the name of *multi-transfer*.

Finally, there are proposals for CTs in the account model that consider the multi-transfer as a future extension and argue that this new payment model could be used to reduce the number of transactions submitted by users [60, 20, 45]. However, none of these solutions concretely develop such an extended payment model.

1.3 Contribution

We propose a method to reduce the costs in account-based privacy solutions for the public blockchain. In particular, we design a *ZK-proof based Multi-Transfer scheme*, namely *ZeroMT*, with which the transaction fees can be amortized by performing multiple transfers in a single transaction equipped with an *aggregate ZK-proof*. To this end, ZeroMT is built on a payment model extended to multiple payees which we summarize below.

Considering a smart contract scenario where the multi-transfer scheme is deployed, each user in the network can execute a user program (acting as a Prover) to generate a multi-transfer transaction. This transaction allows the user to transfer money privately to multiple payees (hidden amounts and zero-knowledge sender/receiver balances). An aggregate ZK-proof is also generated and provided with the transaction to prove the statements behind a *zero-knowledege multi-transfer relation*. Once the user submits the multi-transfer transaction, the balance admin (e.g. the smart contract acting as a Verifier) verifies the ZK-proof and updates the balances of all the users with the right amounts. 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 submitted by the users.

The *ZeroMT proof system* relies on Bulletproofs [21] (BP) and Σ-Protocols theory, which are know to be trustless and secure under standard cryptographic assumptions (e.g., DLOG) when instantiated under suitable cyclic groups. Moreover, the BP system adopts a subroutine for an inner-product relation, called *Inner-Product Argument* (IPA). Despite the logarithmic-sized proofs, it turns out that the IPA subroutine significantly weighs on the whole ZK-proof verification time. This because the verifier logic of the IPA involves expensive linear-time operations. Nonetheless the drawback of non-succinct verification, we follow the direction where trustless proof systems are going, landing into the *Compressed Σ-protocol* theory.

The final part of this dissertation is devoted to optimizing the computational costs of the IPA. To this end, we propose the compressed Σ-protocol version of the original IPA, namely the *compressed Σ-Inner-Product Argument* (Σ-IPA), which shows reduced computational cost.

Finally, the methodology adopted in this dissertation covers both theoretical and practical aspects. Indeed, the security of our cryptographic scheme and underlying ZK-proofs is proved through rigorous theorems and proofs, while practical performance evaluations are supported by the development of a Rust library (see the github repository [36]) and the use of established tools in the research community.

Organization. This work is organized as follows.

- Chapter 2 presents the mathematical background, such as the group theory, elliptic curve groups, computational security, the foundations of proof systems and their security definitions; it also gives the applied perspective by presenting the interactive protocols under study.
- Chapter 3 presents *ZeroMT*, from the high-level cryptographic scheme to the underlying interactive ZK-proof, while also providing security proofs, implementation details and concrete evaluations.
- Chapter 4 introduces to the IPA optimization, presenting the *compressed Σ-Inner-Product Argument* along with security proofs and efficiency analysis.
- Chapter 5 are the conclusions and future directions.

1.4 Publications

This section discusses the author's publications. There are also papers not discussed in this thesis, but which have an impact on the thesis in a transversal manner.

> [29] Corradini, F., Marcelletti, A., Morichetta, A., Polini, A., Re, B., Scala, E., & Tiezzi, F. (2021). Model-driven engineering for multi-party business processes on multiple blockchains. *Blockchain: Research and Applications, 2(3), 100018.*

With this journal, the author investigated privacy-enabling protocols and technologies for blockchains, while also delving into engineering and automation methodologies for systems such as Ethereum and Hyperledger Fabric. Additionally, the author addressed security and privacy aspects, which gave rise to interest in advanced cryptographic techniques for privacy in blockchain.

> [30] Corradini, F., Mostarda, L., & Scala, E. (2022, March). ZeroMT: multi-transfer protocol for enabling privacy in off-chain payments. *In International Conference on Advanced Information Networking and Applications (pp. 611-623). Cham: Springer International Publishing.*

With this conference, the author delved into research on zero-knowledge proofs and privacy protocols, focusing on the emerging context of confidential transactions. In particular, the author introduced the concept of "private multi-transfer", which allows parties in an off-chain protocol to gain privacy for their financial transactions on the public blockchain.

> [63] Scala, E., Dong, C., Corradini, F., & Mostarda, L. (2023, March). Zero-Knowledge Multi-transfer Based on Range Proofs and Homomorphic Encryption. *In International Conference on Advanced Information Networking and Applications (pp. 461-472). Cham: Springer International Publishing.*

With this conference, the author developed the "MTproof" zero-knowledge interactive proof system behind the multi-transfer, enabling the construction of ZK-proofs for transfers involving multiple payees.

> [65] Scala, E., & Mostarda, L. (2023, March). Range Proofs with Constant Size and Trustless Setup. *In International Conference on Advanced Information Networking and Applications (pp. 301-310). Cham: Springer International Publishing.*

With this conference, the author presented a potential optimization on range proofs, reducing proof size from logarithmic to constant and achieving efficient verification time.

> [66] Scala, E., & Mostarda, L. (2024, April). Efficient Inner-Product Argument from Compressed Σ-Protocols and Applications. *In International Conference on Advanced Information Networking and Applications (pp. 326-337). Cham: Springer Nature Switzerland.*

With this conference, the author proposed the compressed Σ-protocol version of the original IPA, namely the compressed Σ-IPA. The author provided security proofs and conducted an efficiency analysis of the proposed argument system.

> [64] Scala, E., Dong, C., Corradini, F., & Mostarda, L. (2023). ZeroMT: Towards Multi-Transfer transactions with privacy for account-based blockchain. *Journal of Information Security and Applications, 83:103794.*

With this journal, the author extended ZeroMT [30], defining the complete cryptographic scheme for the multi-transfer, providing rigorous security definitions, and proving them. Additionally, the author released a modular and open-source library for implementing and evaluating the multi-transfer scheme.

PART II

BACKGROUND

CHAPTER 2

PRELIMINARIES

This chapter is aimed at introducing the preliminary notions used in this dissertation.

2.1 Notations

We denote with $\lambda \in \mathbb{N}$ the security parameter, PPT means probabilistic polynomial-time, and with $s \xleftarrow{\$} \mathcal{S}$ we indicate a random variable s uniformly sampled from the set $\mathcal{S}$. We consider cyclic groups of large prime order p denoted with $\mathbb{G}$, and $|\mathbb{G}|$ is the order of the group. We use the *group-generation* function $\mathcal{G}$ on input the security parameter 1^λ (written in unary) to generate the tuple $(\mathbb{G}, p, g) \leftarrow \mathcal{G}(1^\lambda)$, where $\mathbb{G}$ describes a cyclic group, p is the group order and g is a generator of the group. We use the *multiplicative notation* for group operations and scalar multiplications. Rings of integers modulo prime p are denoted with $\mathbb{Z}_p$, and the invertible elements of $\mathbb{Z}_p$ are in $\mathbb{Z}_p^*$. We denote vectors in bold, e.g., $\mathbf{v} = (v_1, ..., v_n) \in \mathbb{Z}_p^n$ is a vector of scalars and $\mathbf{g} = (g_1, ..., g_n) \in \mathbb{G}^n$ a vector of generators. In uppercase letters we denote *commitments*, e.g., $V = commit(v)$ is a commitment for $v \in \mathbb{Z}_p$ and $commit(.)$ is the *commitment function* with an implicit *blinding factor*.

In the table 2.1, we summarize further notations and operations that we use in our proof systems.

2.2 Basic Group Theory

This section provides notions of prime numbers, modular arithmetic, groups and cyclic groups.

Table 2.1: Proof system notations and operations.

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.
$\mathbf{v} = (v_1, ..., v_n) \in \mathbb{Z}_p^n$	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.
$\mathbf{v}_1 \circ \mathbf{v}_2 = (v_{1,1} \cdot v_{2,1}, ..., v_{1,n} \cdot v_{2,n}) \in \mathbb{Z}_p^n$	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 $\mathbf{g} = (g_1, ..., g_n) \in \mathbb{G}^n$ vector of generators.
$\mathbf{s}_{[:k]} = (s_1, ..., s_k) \in \mathbb{F}^k$ $\mathbf{s}_{[k:]} = (s_{k+1}, ..., s_n) \in \mathbb{F}^{n-k}$	vector slice operations.
$\mathbf{s}_{lo} = (s_1, ..., s_k) \in \mathbb{F}^k$ $\mathbf{s}_{hi} = (s_{k+1}, ..., s_n) \in \mathbb{F}^{n-k}$	vector slice operations (simplified notation), when the size n of a vector $\mathbf{s}$ is a power of two. $\mathbf{s}_{lo}$ is of size $k = n/2$ and $\mathbf{s}_{hi}$ of size $n - k$, which represent the first and second halves of $\mathbf{s}$ respectively.
$p(X) \in \mathbb{Z}_p^n[X]$	vector polynomial $p(X) = \sum_{i=0}^{d} \mathbf{p_i} X^i$, with d the polynomial degree and $\mathbf{p_i} \in \mathbb{Z}_p^n$.

2.2.1 Prime numbers and modular arithmetic

We recall some basic notions of prime number and modular arithmetic from [48]. Let $\mathbb{Z}$ be the set of integers. For a and $b \in \mathbb{Z}$, if $a|b$ (a divides b) and $a|c$ then $a|(xb + yc)$ for any $x, y \in \mathbb{Z}$. If a is positive, $a|b$ and $a \neq 1 \wedge a \neq b$, we say that a is a *factor*, or *non-trivial divisor*, of b. If a positive integer p has no factors we say that p is a *prime number*. Any positive integer, not equal to 1, can be written uniquely as a product of primes. We write with $gcd(a, b)$ the greatest common divisor d between a and $b \in \mathbb{Z}$. Note that, for an integer a and a prime p, $gcd(a, p)$ is either equal to 1 or p. Two integers a and b are *relatively prime* if $gcd(a, b) = 1$. The greatest common divisor between a and $b \in \mathbb{Z}$ can be computed in *polynomial time* with the *Euclidean algorithm*. Another polynomial time algorithm of interest is the *extended Euclidean algorithm*, which can be used to compute the integers x, y such that $xa + yb = gcd(a, b)$ for two positive integers a, b.

Now we give some fundamentals of modular arithmetic. Let a, b and n be in $\mathbb{Z}$, with $n > 1$. ($a \bmod n$) denotes the remainder of a divided by n. The process of mapping a to ($a \bmod n$) is called *reduction modulo* n. Two integer a and b are *congruent modulo* n, written $a = b \bmod n$ (or $a \equiv b \bmod n$), if ($a \bmod n$) = ($b \bmod n$). By consequence, $a = b \bmod n$ if and only if $n|(a - b)$. Moreover, $a = (b \bmod n)$ implies $a = b \bmod n$, but the opposite is not true. The congruence modulo n is reflexive ($a = a \bmod n, \forall a$), symmetric ($a = b \bmod n \implies b = a \bmod n$), and transitive ($a = b \bmod n \wedge b = c \bmod n \implies a = c \bmod n$). Congruence modulo n respects the addition, e.g., $(a + b) = (a' + b') \bmod n$ if $a = a' \bmod n$

and $b = b' \bmod n$. The same is for subtraction and multiplication while, differently, the congruence does not respect division. However, a notion of division comes from invertibles. We say that a given integer b is *invertible* modulo n if there exists an integer c such that $bc = 1 \bmod n$, where c is called *multiplicative inverse* of b modulo n. It follows that, if c is a multiplicative inverse of $b \bmod n$ then so is $c \bmod n$, and if c' is another multiplicative inverse of b then $c = c' \bmod n$. When b is invertible, we denote with $b^{-1} \in \{1, ..., n-1\}$ the *unique* multiplicative inverse of b. Therefore, we can define division by b modulo n as follows: $(a/b \bmod n) \stackrel{\text{def}}{=} (ab^{-1} \bmod n)$. Division by b is only possible when b is invertible. Let $b, n \in \mathbb{Z}$, with $b \geqslant 1$ and the modulo $n > 1$, then b is invertible if and only if b and n are relatively prime. Hence, we can use the *Euclidean algorithm* to determine in polynomial time whether a given b has a multiplicative inverse modulo n.

2.2.2 Groups

A group $\mathbb{G}$ is a set associated with a binary operation, referred to as *group operation*. A group needs to respect the closure, existence of identity and inverses, and associativity conditions. A group is *abelian* if also the commutativity condition holds. A group with a finite number of elements is called *finite* group; we denote with $|\mathbb{G}|$ the *order* of the group (i.e., number of elements or size of the group). The identity element in a group is *unique* and each element $g \in \mathbb{G}$ has a *unique inverse*. Two distinct notations can be used for the *group operation*: *additive* or *multiplicative* notation. When using the additive notation, the group operation between two elements $g, h \in \mathbb{G}$ is denoted with $g + h$, the identity with 0, the inverse of g with $-g$, and sometimes we write $g - h$ instead of $g + (-h)$. When using the multiplicative notation the group operation between two elements $g, h \in \mathbb{G}$ is denoted with $g \cdot h$, the identity by 1, the inverse of g with g^{-1}, and sometimes we write g/h instead of $g \cdot h^{-1}$. The same group operation could be repeated a number of times, say k times, for a fixed group element. Therefore, it is useful to define a method such as the *group exponentiation* (not to be considered as a group operation). With the additive notation, a group exponentiation for a group element g is defined as

$$k \cdot g \stackrel{\text{def}}{=} \underbrace{g + g + \cdots + g}_{k \in \mathbb{N} \text{ times}}.$$

With the multiplicative notation, a group exponentiation for an element g is defined as

$$g^k \stackrel{\text{def}}{=} \underbrace{g \ldots g}_{k \in \mathbb{N} \text{ times}}.$$

Unless explicitly specified, in this dissertation we will use multiplicative notation. Several properties follows from the exponentiation, if $\mathbb{G}$ is a finite group with order $m = |\mathbb{G}| > 1$ we have that: for any group element g, $g^m = 1$; we are able to work "modulo the group order" in the exponent, i.e., for any $g \in \mathbb{G}$ and any integer x, $g^x = g^{(x \bmod m)}$; let $f_e : \mathbb{G} \to \mathbb{G}$ be a function defined as $f_e(g) = g^e$ for an integer $e > 0$, if $gcd(e, m) = 1$ then f_e is a bijection and e is invertible modulo m. Thus, we have $d = e^{-1} \bmod m$ and therefore f_d is the inverse of f_e.

Sometimes it may be useful to use *isomorphic* groups. An isomorphism is a different way of working on a group $\mathbb{G}$ by using another group $\mathbb{H}$ which is isomorphic with $\mathbb{G}$. Two groups $\mathbb{G}$ and $\mathbb{H}$ defined with group operations $\circ_{\mathbb{G}}$ and $\circ_{\mathbb{H}}$ respectively, are isomorphic if for a function $f : \mathbb{G} \to \mathbb{H}$: f is a bijection, and for all $g_1, g_2 \in \mathbb{G}$ we have $f(g_1 \circ_{\mathbb{G}} g_2) = f(g_1) \circ_{\mathbb{H}} f(g_2)$.

We write $\mathbb{G} \simeq \mathbb{H}$ when such function f exists, hence the groups are isomorphic. Note that if $\mathbb{G}$ is finite, $\mathbb{H}$ must be finite and of the same size. Moreover, if there exists an isomorphism f from $\mathbb{G}$ to $\mathbb{H}$ then f^{-1} is an isomorphism from $\mathbb{H}$ to $\mathbb{G}$.

Examples. The set $(\mathbb{Z}, +)$ is a group because the identity is 0 and each $x \in \mathbb{Z}$ has inverse $-x$. The set $(\mathbb{Z}, \cdot)$ is not a group since not all elements have an inverse. The set $(\mathbb{Z}_n, +)$ of integers modulo n is a group of order n. The set $(\mathbb{Z}_n^*, \cdot)$ of invertibles of $\mathbb{Z}_n$ is a group under multiplication modulo n. Now we see some examples from the properties of exponentiation. Consider the set $\mathbb{Z}_5 = \{0, 1, 2, 3, 4\}$ with $m = |\mathbb{Z}_5| = 5$ and under addition, i.e., $(\mathbb{Z}_5, +)$. Then $\forall g \in \mathbb{Z}_5$, $m \cdot g = 0$ where 0 is the identity: $5 \cdot 0 \bmod 5 = 0$, $5 \cdot 1 \bmod 5 = 0$, ..., $5 \cdot 4 \bmod 5 = 0$. Consider the set $\mathbb{Z}_5^* = \{1, 2, 3, 4\}$ with $m = |\mathbb{Z}_5^*| = 4$ and under multiplication, i.e., $(\mathbb{Z}_5^*, \cdot)$. Then $\forall g \in \mathbb{Z}_5^*$, $g^m = 1$ where 1 is the identity: $1^4 \bmod 5 = 1$, $2^4 \bmod 5 = 1$, ..., $4^4 \bmod 5 = 1$. Let's see that we are able to work modulo the group order in the exponent. Consider $\mathbb{Z}_5^*$ as before, then $m = 4$ and $n = 5$, we have $g^x = g^{(x \bmod m)}$ for any $g \in \mathbb{Z}_5^*$ and any integer x: $2^2 = 2^{(2 \bmod 4)} \implies 2^2 \bmod 5 = 2^{(2 \bmod 4)} \bmod 5 \implies 4 = 2^2 \bmod 5 = 4$, $2^3 = 2^{(3 \bmod 4)} \implies 8 \bmod 5 = 2^3 \bmod 5 \implies 3 = 3$.... Let's see why isomorphism can be useful with the following example. Let $g_1, g_2 \in \mathbb{G}$ and suppose both f and f^{-1} can be computed efficiently. Now, suppose $\circ_{\mathbb{H}}$ is more efficient than $\circ_{\mathbb{G}}$. We can do the computations: i) $h_1 = f(g_1)$, $h_2 = f(g_2)$; ii) $h = h_1 \circ_{\mathbb{H}} h_2$; iii) $g = f^{-1}(h) \simeq g_1 \circ_{\mathbb{G}} g_2$.

Subgroups. A set $(\mathbb{H}, \circ)$ is a *subgroup* of $(\mathbb{G}, \circ)$ if it satisfies the following conditions: $\mathbb{H}$ is a subset of $\mathbb{G}$ ($\mathbb{H} \subseteq \mathbb{G}$); the operator $\circ$ on $\mathbb{H}$ is the same as the operator on $\mathbb{G}$; $\mathbb{H}$ is a group, meaning it has the closure condition, each

element of $\mathbb{H}$ has an inverse in $\mathbb{H}$, and the identity element of $\mathbb{G}$ is also in $\mathbb{H}$.

The *Lagrange's theorem* states that for a finite group $\mathbb{G}$, the order of any subgroup $\mathbb{H}$ of $\mathbb{G}$ divides the order of the group $\mathbb{G}$. According to Lagrange's theorem, the number

$$h = \frac{|\mathbb{G}|}{|\mathbb{H}|}$$

is always an integer and it is called the *cofactor of the subgroup.*

2.2.3 Cyclic groups

Let $\mathbb{G}$ be a finite group of order m. Let $i \leqslant m$ be the *smallest positive integer* for which $g^i = 1$, for arbitrary $g \in \mathbb{G}$. Then the set, denoted with

$$\langle g \rangle = \{g^0, ..., g^{i-1}\},$$

repeats after i terms and contains at most i elements. We call the set $\langle g \rangle$ the *subgroup generated by* g. If the order of $\langle g \rangle$ is i, then i is called the *order of* g (i.e., the order of g is the smallest positive integer i with $g^i = 1$).

The indentity element of any group $\mathbb{G}$ is the only element of order 1, i.e., it generates $\langle 1 \rangle = \{1\}$. If there is an element $g \in \mathbb{G}$ of order m, where m is the order of $\mathbb{G}$, then $\langle g \rangle = \mathbb{G}$. In this case, we call $\mathbb{G}$ a *cyclic group* and say that g is a *generator* of $\mathbb{G}$ (a cyclic group may have multiple generators). If g is a generator, then every element $h \in \mathbb{G}$ is equal to g^x for some $x \in \{0, ..., m-1\}$.

Let $\mathbb{G}$ be a finite group of order m and $g \in \mathbb{G}$ an element of order i, the properties follow: for any integer x, we have $g^x = g^{(x \bmod i)}$; $g^x = g^y$ if and only if $x = y \bmod i$; different elements of the same group $\mathbb{G}$ may have different orders, however, we can place some restrictions on these possible orders, i.e., the order i of $g \in \mathbb{G}$ divides m $(i|m)$; if $\mathbb{G}$ is a *group of prime order* p, then $\mathbb{G}$ is *cyclic* and all elements of $\mathbb{G}$, except the identity, are generators of $\mathbb{G}$. In this last point, we considered the order m of $\mathbb{G}$ equal to a prime number p. By the fact that $i|m$, the only possible orders i of elements in $\mathbb{G}$ are 1 and p. Only the identity has order 1, and so all other elements have order p and generate $\mathbb{G}$. One last consideration is that all cyclic groups of the same order are isomorphic.

2.3 Provable Security

In this section, we present our framework for establishing the security of cryptographic schemes. Specifically, we introduce computational security, security experiments and the underlying cryptographic assumptions employed in our security proofs.

2.3.1 Probability

We consider the *discrete* probability theory. In this context, let $\mathbb{U}$ be a countable sample space, a *probability distribution* over $\mathbb{U}$ is a function $p : \mathbb{U} \to [0,1]$ such that $\sum_{x \in \mathbb{U}} p(x) = 1$. More generally, the function p assigns to every element of $\mathbb{U}$ a number between 0 and 1. This number is called the probability of that particular element in $\mathbb{U}$. A probability distribution is called *uniform distribution* if $p(x) = 1/|\mathbb{U}|$ for all $x \in \mathbb{U}$, where $|\mathbb{U}|$ is the size of $\mathbb{U}$ or the number of elements. In other words, all elements of $\mathbb{U}$ are equally likely to be sampled by the uniform distribution. A subset $\mathbb{E} \subseteq \mathbb{U}$ is called an *event*, and every event is associated with the probability $Pr[\mathbb{E}] = \sum_{x \in \mathbb{E}} p(x) \in [0,1]$.

As an example, let $\mathbb{U} = \{0,1\}^8$ with $|\mathbb{U}| = 256$ and the event $\mathbb{E}$ defined as the set $\mathbb{E} = \{\forall x \in \mathbb{U} : lsb_2(x) = 11\} \subseteq \mathbb{U}$ with $lsb_2(x)$ the two least significant bits of x. We have exactly 64 elements with $lsb_2(x) = 11$ inside $\mathbb{U}$ that has a size of 256. Considering the uniform distribution on $\mathbb{U}$, the probability for all elements respect to this distribution is equal to 1/256. We have that the probability of the event $\mathbb{E}$ occurs (in the sense that when we choose a random element in $\mathbb{U}$ we get a bit string that ends with 11), is equal to: $Pr[\mathbb{E}] = \frac{1}{4} = 64 * \frac{1}{256}$.

Two events $\mathbb{E}_1$ and $\mathbb{E}_2$ are *independent* if the occurrence of event $\mathbb{E}_1$ tells you nothing about whether event $\mathbb{E}_2$ occurs or not. In other words, knowing the outcome of one event does not affect the probability of the other event occurring. Thus, the probability that both events occur is equal to their multiplication:

$$Pr[\mathbb{E}_1 \wedge \mathbb{E}_2] = Pr[\mathbb{E}_1] * Pr[\mathbb{E}_2]$$

Definition 1 (Random variable)**.** *A random variable X is a function $X : \mathbb{U} \to \mathbb{V}$ for some non-empty set $\mathbb{V}$. Moreover, the random variable X induces a probability distribution over $\mathbb{V}$ such that:*

$$v \to Pr[X = v] := \sum_{x \in X^{-1}(v)} p(x) \in [0,1]$$

for any $v \in \mathbb{V}$ and events $X^{-1}(v) \subseteq \mathbb{U}$.

We denote with $r \xleftarrow{\$} \mathbb{U}$ a *uniform random variable* sampled over (some finite set) $\mathbb{U}$. Formally, r is said to be *uniformly distributed* over $\mathbb{U}$ if $Pr[r = x] = 1/|\mathbb{U}|$ for all $x \in \mathbb{U}$.

As an example, let $\mathbb{U} = \{0,1\}^2$ and r a uniform random variable over $\mathbb{U}$. Let define a new random variable $X = r_1 + r_2$, that is the sum of the

first and second bit of r (e.g., if $r = 00$ then $X = 0 + 0 = 0$). What is the probability that X is equal to 2? $Pr[X = 2] = \frac{1}{4}$. Because, the only way that $X = 2$ is if r happens to be 11. So, $Pr[X = 2] = Pr[r = 11]$. But the probability that r is equal to 11 is basically 1/4, because r is uniform over the set of all two bit strings.

A *randomized algorithm* is a non-deterministic function $\mathcal{A} : \mathbb{X} \rightarrow \mathbb{Y}$, which takes some input x from the message space $\mathbb{X}$, an implicit randomness r typically sampled uniformly from the set of all n bit strings, and produces a different output $y \in \mathbb{Y}$ every time we run $\mathcal{A}$, formally:

$$r \xleftarrow{\$} \{0,1\}^n, y \leftarrow \mathcal{A}(x; r)$$

So, the output y is a random variable defining a distribution over the set of all possible outputs of $\mathcal{A}$, given the message x. We denote $\mathcal{A}(x)$ the random variable defined in this way. Instead, if X is a random variable, D_X is some distribution defined by X, and $\mathcal{A}$ is a randomized algorithm, we denote $\mathcal{A}(X)$ the random variable with $Pr[\mathcal{A}(X) = y] = Pr[\mathcal{A}(x) = y : x \xleftarrow{R} D_X]$, where x is sampled according to the distribution D_X and the probability is also over the randomness r of $\mathcal{A}$.

Two random variables X and Y taking values in $\mathbb{V}$ are *independent* if $\forall a, b \in \mathbb{V}$:

$$Pr[X = a \wedge Y = b] = Pr[X = a] * Pr[Y = b]$$

As an example, let $\mathbb{U} = \{0,1\}^2$ and $r \xleftarrow{\$} \mathbb{U}$. We define two random variables X and Y as $X = lsb(r)$ and $Y = msb(r)$ (least and most significant bit respectively). We claim that X and Y are independent. Intuitively, choosing r uniformly from $\mathbb{U}$ is basically the same as flipping an unbiased coin twice. The first bit of r corresponds to the outcome of the first flip and the second bit corresponds to the outcome of the second flip. There are four possible outcomes all equally likely, which is why we get the uniform distribution over the two bit strings. Now, X and Y are independent because knowing the result of the first flip (the lsb of r) tells you nothing about the result of the second flip (the msb of r). Formally, we would have to prove that for all 0 and 1 pairs ($X = 0$ and $Y = 0$, $X = 1$ and $Y = 1$, and so on...), these probabilities are multiplied. Let's consider one of these pairs:

$$Pr[X = 0 \wedge Y = 0] = Pr[r = 00] = \text{ by } uniform \text{ distribution } \frac{1}{|\mathbb{U}|} = \frac{1}{4}$$

Since in the set $\{0,1\}^2$ we have two elements with $lsb = 0$ and two elements with $msb = 0$, this holds:

$$\frac{1}{4} = Pr[X = 0] * Pr[Y = 0] = \frac{1}{2} * \frac{1}{2}$$

Two random variables X and Y taking values in $\mathbb{V}$ are *mutually independent* if $\forall a, b \in \mathbb{V}$:

$$Pr[X = a|Y = b] = Pr[X = a] \text{ and } Pr[Y = b|X = a] = Pr[Y = b]$$

where $Pr[X = a|Y = b]$ and $Pr[Y = b|X = a]$ are the *conditional probabilities*.

2.3.2 Computational security

In modern cryptography, *computational security* serves as a method to mathematically prove the security of a cryptographic scheme. In computational security two relaxations are introduced: i) adversaries are considered *efficient algorithms*, meaning they operate for some feasible amount of time; since they have bounded computational power, we can make the resources required to break a scheme larger than those available to any realistic adversary. ii) Adversaries can potentially succeed (or security of the scheme can potentially fail) with some very small probability.

Computational security of cryptographic schemes can be defined with the *asymptotic approach*. This approach introduces a *security parameter* as input to the scheme and to the involved parties (honest parties and the adversary). The security parameter is chosen when the scheme is initialized and is assumed to be known to any adversary attacking the scheme. Hence, we view the running time of the adversary and its success probability as functions of the security parameter. In general, the security parameter is a mechanism that allows to "tune" the security of a scheme to the desired level: increasing the length of the security parameter decreases the success probability in breaking the scheme. Moreover, the efficiency of the scheme, such as the costs of party computations, communication, and the size of public input, is evaluated for any given security parameter.

We define the *small probabilities of success* as the success probabilities of any adversary smaller than any inverse polynomial in the security parameter; such probability is called *negligible success probability*. When defining the adversaries, we consider them as *probabilistic polynomial time* (PPT) algorithm, where probabilistic means that the algorithms are *randomized*. Polynomial time means that the running time of an algorithm is bounded by a polynomial that expresses the number of steps to compute the algorithm. Moreover, such polynomial denoting the running time of an algorithm is a function of the length of the inputs. We assume that the inputs of an algorithm are represented in binary, therefore, the algorithm's running time is measured in terms of the length of the binary representation of the inputs. We now give formal definitions for "polynomial-time algorithms" and "negligible function".

Definition 2 (Polynomial-time algorithm)**.** *An algorithm $\mathcal{A}$ runs in polynomial time if there exists a polynomial p such that, for every input $n \in \{0,1\}^*$, the computation of $\mathcal{A}(n)$ terminates within at most $p(|n|)$ steps.*

Definition 3 (Negligible function)**.** *A function f is negligible if for every positive polynomial p there is an N such that for all integers $n > N$ it holds that $f(n) < 1/p(n)$.*

Roughly speaking, a negligible function is one that is asymptotically smaller than any inverse polynomial function. Instead of $n > N$ we can also say for *all sufficient large values of* n. An equivalent formulation of the above is to require that for all constants c there exists an N such that for all $n > N$ it holds that $f(n) < n^{-c}$. We denote an arbitrary negligible function by **negl**.

Sometimes it can be useful to define the term **super-poly**, which is easily derived from the definition of negligible function.

Definition 4 (Super-poly function)**.** *A function f is called **super-poly** if $1/f$ is negligible.*

For completeness, we can also define the **poly-bounded** function, i.e., one f that is bounded by some fixed polynomial.

Definition 5 (Poly-bounded function)**.** *A function f is called **poly-bounded** if there exist constants c and d such that, for all integers $n \geqslant 0$, we have $f(n) \leqslant n^c + d$.*

When considering cryptographic schemes, $\lambda \in \{0,1\}^*$ denotes the bit-string of the security parameters, $|\lambda|$ the length of the string, and $\mathcal{A}$ refers to the adversary algorithm. We sometimes provide algorithms with the security parameter written in *unary* (i.e., as 1^λ, or a string of n ones) as input.

In summary, the general framework of any security definition will be as follows:

"*A scheme is secure if for every PPT adversary $\mathcal{A}$ carrying out some attack, the probability that $\mathcal{A}$ succeeds in breaking the scheme is negligible*".

2.3.3 Security experiments

We model the security of a scheme and its underlying cryptographic assumptions in terms of *security experiments* and *event probabilities*. We use the notation Pr[experiment : event] to denote the probability of some event after the specified experiment has occurred.

An experiment formally describes how algorithms, such as the adversary, generate or sample the elements involved in the event. We assume that the

sampling of elements happens with respect to the *uniform distribution* over finite sets. Note that, an adversary might succeed in a security experiment simply by guessing the sampled element. The distance between the success probability of an adversary and the success probability of the random guessing is referred to as the *advantage* of the adversary. We denote with $Adv^{EXP}_{\mathcal{A},\Pi}$ the advantage of any adversary $\mathcal{A}$ attacking the scheme Π during the experiment EXP.

In security definitions, we consider a scheme secure if the adversary's advantage can be bounded by some negligible function in λ. Computational security also comprises what are known as *security reductions*, through which we can reduce the security of a scheme to the hardness of computational problems. Let $\mathcal{A}$ be any PPT adversary performing a specific attack during the experiment EXP to the scheme Π. From the advantage of $\mathcal{A}$ in that experiment, we can derive the advantage of another PPT adversary $\mathcal{B}$ against a computationally hard problem $PROB$ and prove that $Adv^{EXP}_{\mathcal{A},\Pi} \leqslant Adv^{PROB}_{\mathcal{B}}$. This shows that we can break the security of the scheme with at least the same probability as solving an instance of the hard problem. Thus, if the problem is intractable, then necessarily $Adv^{EXP}_{\mathcal{A},\Pi}$ is negligible and Π satisfies the security we want to prove.

When introducing security experiments, we specify the capabilities of any adversary (e.g., whether $\mathcal{A}$ knows the secret key of the scheme or not) and the interactions between algorithms involved in the scheme. In particular, we can introduce an algorithm called *challenger* denoted with $\mathcal{C}$, which represents the honest party and interacts with the adversary. We often want to give some extra power to an adversary by giving access to an *oracle* algorithm, denoted $\mathcal{O}$, which responds with the desired outputs. This is useful when we don't want to give to the adversary the secret information needed for $\mathcal{O}$ to operate on its own. When $\mathcal{A}$ asks for specific outputs, we say that $\mathcal{A}$ *queries* $\mathcal{O}$ or that $\mathcal{A}$ has *oracle access* to $\mathcal{O}$.

2.3.4 Discrete logarithm assumptions

We now present some computational problems that are intractable in cyclic groups of prime order. We deal with such problems because we use them in the security proofs of our cryptographic schemes.

Let $\mathcal{G}$ a polynomial-time group-generation algorithm which, on input 1^λ (with λ the security parameter), outputs the tuple $(\mathbb{G}, p, g)$ where $\mathbb{G}$ is a description of a cyclic group, p is its order, and $g \in \mathbb{G}$ is a generator. The description of a cyclic group specifies how group elements are represented as (unique) bit-strings. We require that there are efficient algorithms (running in polynomial time in λ) for computing the group operation in $\mathbb{G}$, as well as for testing whether a given bit-string represents an element of $\mathbb{G}$. Efficient computation of the group operation implies efficient algorithms for exponen-

tiation and for sampling uniform group elements, e.g., when we choose a uniform $x \xleftarrow{\$} \mathbb{Z}_p$ and set $h = g^x \in \mathbb{G}$.

If $\mathbb{G}$ is a cyclic group of order p with generator g, then for every $h \in \mathbb{G}$ there is a unique $x \in \mathbb{Z}_p$ such that $g^x = h$. We call x the *discrete logarithm of h with respect to g* and we write $x = \log_g h$.

The *discrete-logarithm problem* (DLOG) in a cyclic group $\mathbb{G}$ with generator g is to compute $\log_g h$ for a uniform element $h \in \mathbb{G}$. In suitable groups, this problem is assumed to be computationally intractable, i.e., polynomial-time algorithms succeed with at most negligible probability in finding the discrete logarithm. The following definition is for the *discrete logarithm assumption*.

Definition 6 (DLOG assumption)**.** *We say that the discrete-logarithm problem is hard relative to $\mathbb{G}$ if for all probabilistic polynomial-time algorithm $\mathcal{A}$ there exists a negligible function* **negl** *such that*

$$Pr\left[\begin{array}{ll}(\mathbb{G}, p, g) \leftarrow \mathcal{G}(1^\lambda),\ h \xleftarrow{\$} \mathbb{G}; & \\ x \in \mathbb{Z}_p \leftarrow \mathcal{A}(\mathbb{G}, p, g, h) & : g^x = h\end{array}\right] \leqslant \mathbf{negl}(\lambda)$$

Roughly speaking, a group-generation algorithm is executed to obtain $(\mathbb{G}, p, g)$ and a uniform $h \xleftarrow{\$} \mathbb{G}$ is chosen. The PPT algorithm $\mathcal{A}$, which we will call the adversary in the security definitions, is given the tuple $(\mathbb{G}, p, g, h)$ and outputs $x \in \mathbb{Z}_p$. Thus, the fact that $g^x = h$ must happen with negligible probability in λ.

An alternative, but useful definition, is the *non-trivial discrete-logarithm relation* from [21].

Definition 7 (Non-trivial DLOG relation)**.** *For all probabilistic polynomial-time algorithm $\mathcal{A}$ and for all $k \geqslant 2$ there exists a negligible function* **negl** *such that*

$$Pr\left[\begin{array}{ll}\mathbb{G} \leftarrow \mathcal{G}(1^\lambda),\ h_1, ..., h_k \xleftarrow{\$} \mathbb{G}; & \\ x_1, ..., x_k \in \mathbb{Z}_p \leftarrow \mathcal{A}(\mathbb{G}, h_1, ..., h_k) & : \begin{array}{l}\exists x_1, ..., x_k \neq 0 \\ \wedge \prod_{i=1}^{k} h_i^{x_i} = 1\end{array}\end{array}\right] \leqslant \mathbf{negl}(\lambda)$$

We say that there is a *non-trivial DLOG relation* between uniformly random group elements $h_1, ..., h_k$ when $\prod_{i=1}^{k} h_i^{x_i} = 1$ and each $x_1, ..., x_k \in \mathbb{Z}_p$ is non-zero. Thus, the DLOG relation assumption states that an adversary $\mathcal{A}$ cannot find a non-trivial relation between randomly chosen group elements.

Related to the DLOG problem is the *Decisional Diffie-Hellman* (DDH) *problem*. The hardness of this problem is formalized in the following definition.

Definition 8 (DDH assumption)**.** *We say that the DDH problem is hard relative to* $\mathbb{G}$ *if for all probabilistic polynomial-time algorithm* $\mathcal{A}$ *there exists a negligible function* **negl** *such that*

$$Pr\left[\begin{array}{l}(\mathbb{G},p,g)\leftarrow\mathcal{G}(1^\lambda),\ x,y,z\xleftarrow{\$}\mathbb{Z}_p,\ b\xleftarrow{\$}\{0,1\},\\ pp_0\leftarrow(\mathbb{G},p,g,g^x,g^y,g^z),\ pp_1\leftarrow(\mathbb{G},p,g,g^x,g^y,g^{xy});\\ b'\in\{0,1\}\leftarrow\mathcal{A}(pp_b)\ :\\ \quad b'=b\end{array}\right]\leqslant\frac{1}{2}+\mathbf{negl}(\lambda)$$

Informally, the DDH problem corresponds to the inability of any adversary, which is given $h_1 = g^x$ and $h_2 = g^y$ for uniformly chosen $x, y \in \mathbb{Z}_p$, to distinguish whether a third group element $h_3 = g^{\log_g h_1 \cdot \log_g h_2} = g^{xy}$ or h_3 was chosen uniformly from $\mathbb{G}$. In the latter case, note that when z is uniform in $\mathbb{Z}_p$, then $h_3 = g^z$ is uniformly distributed in $\mathbb{G}$.

For considerations that follow, in this dissertation we will work with cyclic groups of prime order. The DLOG problem in groups of prime order is harder than in groups of non-prime order. Finding a generator in groups of prime order is trivial (every element of a prime-order group, except the identity, is a generator). Proofs of security for some cryptographic constructions require computing multiplicative inverses of certain exponents. When the group order is prime, any non-zero exponent will be invertible, making this computation possible. Working with prime-order groups applies in situations when the decisional DDH problem should be hard.

2.4 Elliptic Curves

In this section, we recall some basic definitions from the theory of fields, rings and elliptic curves. See [15, 19, 48] for a more detailed exposition.

2.4.1 Fields and rings

A *field* is a set $\mathbb{F}$ along with two binary operations $+, \cdot$ for which the following hold:

- $\mathbb{F}$ is an abelian group with respect to $+$. Here, 0 is the identity element and for $a \in \mathbb{F}$ the *additive inverse*, denote by $-a$, is the unique element satisfying $a + (-a) = 0$.
- $\mathbb{F} \setminus \{0\}$ is an abelian group with respect to $\cdot$. Here, 1 is the identity element and for $a \in \mathbb{F} \setminus \{0\}$ the *multiplicative inverse*, denote by a^{-1}, is the unique element satysfing $aa^{-1} = 1$.

- *Distributivity.* $\forall a, b, c \in \mathbb{F}$, we have $a \cdot (b + c) = ab + ac$.

A finite field $\mathbb{F}$ has a finite number of elements, and its order is the number of elements. In particular, the order of a finite field $\mathbb{F}$ is a *prime power* p^r for some p prime and $r \geqslant 1$. We denote with $\mathbb{F}_{p^r}$ the (unique) finite field of order p^r and call p the *characteristic* of $\mathbb{F}_{p^r}$. We can define the *group exponentiation* as in the case of groups. For $r > 1$, the set $\mathbb{F}_{p^r} \backslash \{0\}$, with respect to the multiplication is a cyclic group of order $p^r - 1$. We denote such multiplicative group with $\mathbb{F}^*_{p^r}$. If there is a large prime factor q of $p^r - 1$, then $\mathbb{F}^*_{p^r}$ has a cyclic subgroup of order q.

A *ring* is a set $\mathcal{R}$ with group laws $+, \cdot$ such that:

- $(\mathcal{R}, +)$ is an abelian group.
- $(\mathcal{R}, \cdot)$ is a monoid (there is no inverse for each element).
- *Distributivity.* $\forall a, b, c \in \mathcal{R}$, we have $a \cdot (b + c) = ab + ac$ and $(a + b) \cdot c = ac + bc$.
- If also $a \cdot b = b \cdot a$, then we have a *commutative ring.*

Note that, rings are like fields but do not respect division in the multiplicative case (in that case rings are semigroups or monoids). The set $(\mathbb{Z}_p, +, \cdot)$ of integers modulo p prime is a ring: $(\mathbb{Z}_p, +)$ is an abelian group (existence of additive identity and inverses) and $(\mathbb{Z}_p, \cdot)$ is a monoid. Moreover, the set $(\mathbb{Z}_p, +, \cdot)$ forms a field when considering $\mathbb{Z}_p \backslash \{0\}$ with respect to the multiplication, as each nonzero element in $\mathbb{Z}_p$ has a multiplicative inverse. We denote with $\mathbb{Z}^*_p$ the set of invertible elements of $\mathbb{Z}_p$.

2.4.2 Elliptic curve groups

An important class of groups in cryptography is given by the groups of *points on elliptic curves.* A helpful way to start discussing elliptic curves is to consider the set of 2-D points (x, y) satisfying the equation:

$$y^2 = x^3 + ax + b \tag{2.1}$$

where $a, b \in \mathbb{R}$ are constants and $-16(4a^3 + 27b^2) \neq 0$ (the discriminant denoted with Δ) to avoid singularity (invalid curves, i.e., curves with cusps or self-intersections).

The presented equation (2.1) is know as *Weierstrass normal form*, which define the set of points on an *elliptic curve over* $\mathbb{R}$ (it means that the coefficients a and b are elements of the field of real numbers $\mathbb{R}$). We denote an elliptic curve over the set $\mathbb{R}$ with $E(\mathbb{R})$. The set of points $E(\mathbb{R})$ along

with an *addition rule* and the *point at infinity* (denoted with $\mathcal{O}$) forms an abelian group, known as the **elliptic-curve group** of E. When discussed graphically, the point at infinity $\mathcal{O}$ is an "extra point" added on every vertical line intersecting the elliptic curve.

In cryptography, elliptic curves are used over finite fields, e.g., the fields $\mathbb{Z}_p$, $\mathbb{F}_{p^r}$ or $\mathbb{F}_{2^r}$. For simplicity, we consider elliptic curves defined over a finite field $\mathbb{F}_p$ with $p > 3$ prime. The following definition is revised from [15].

Definition 9 (Elliptic curve over finite field)**.** *Let $p > 3$ be a prime. An elliptic curve E defined over $\mathbb{F}_p$ is the set of points satisfying equation (2.1) such that*

$$E(\mathbb{F}_p) \stackrel{def}{=} \{(x, y) \in \mathbb{F}_p^2 \mid y^2 = x^3 + ax + b \ mod\ p,\ 4a^3 + 27b^2 \neq 0 \ mod\ p\} \cup \{\mathcal{O}\}$$

where $a, b \in \mathbb{F}_p$ are constants (chosen such that $\Delta \neq 0$) and $\mathcal{O}$ is the point at infinity.

Let $E(\mathbb{F}_p)$ an elliptic curve and let $P = (x, y)$, where $x, y \in \mathbb{F}_{p^r}$ with $r \geqslant 1$, a point on the curve E satisfying the equation (2.1). When $r = 1$ the point P is defined over the *base field* $\mathbb{F}_p$. When $r > 1$ the point is defined over an *extension field* of $\mathbb{F}_p$.

Also in the case of elliptic curves over finite fields, the elliptic-curve group is under the addition which can be defined algebraically. The identity element is the point at infinity $\mathcal{O}$, every point $P \in E(\mathbb{F}_p)$ has an additive inverse $-P$, and the addition is associative and commutative. So the sum of two point $P, Q \in E(\mathbb{F}_p)$ is $P + Q$ and the *exponentiation* corresponds to repeated addition, i.e., $k \cdot P \stackrel{\text{def}}{=} (k-1)P + P$ for any $k \in \mathbb{N}$. Instead of "exponentiation", we call this operation **scalar multiplication** for the additive notation. Moreover, a scalar multiplication can be computed using at most $2\log_2 k$ group operations using the *repeated squaring algorithm* [15].

Let $P \in E(\mathbb{F}_p)$ be a point of order q such that $q \cdot P = \mathcal{O}$. Then, any scalar multiplication of P gives a point in $E(\mathbb{F}_p)$ and results in a **cyclic group**. Such P is a generator, also called **base point**, of the cyclic group (it may generates a subgroup). Moreover, the order of P is also the order of the cyclic group (or subgroup) it generates.

Now, we highlight some basic facts derived from cryptographic assumptions. Let $E(\mathbb{F}_p)$ be a finite abelian elliptic-curve group which is also cyclic. We can ask about the complexity of problems like DLOG and DDH in the group $E(\mathbb{F}_p)$. Once enstablished that DLOG and DDH problems are hard in $E(\mathbb{F}_p)$, we can instantiate cryptographic constructions using this group. The resulting systems are called *elliptic curve cryptosystems.*

The DLOG problem in $E(\mathbb{F}_p)$ becomes as follows. Let $P \in E(\mathbb{F}_p)$ be a point of order q and $qP = \mathcal{O}$. We are given P and q, along with another point $Q = kP$, for some $k \in \mathbb{Z}_q$. We want to compute k, the discrete log of Q base P in $E(\mathbb{F}_p)$. Instead, the DDH problem becomes (informally) the problem of distinguishing tuples of the form (aP, bP, abP) from those of the form (aP, bP, cP). These problems are believed to be hard in elliptic-curve groups (or subgroups thereof) of large prime order.

2.4.3 Elliptic curves in practice

The best known algorithm that computes the DLOG in every cyclic group of order q requires at most $O(\sqrt{q})$ group operations. Similarly, in elliptic-curve groups we want the best algorithm to be not much faster than $\sqrt{q}$, where $q = |E(\mathbb{F}_p)|$. In case the algorithm is faster, the curves are considered cryptographically weak, e.g., *anomalous curves* where $|E(\mathbb{F}_p)| = p$ or those such that $|E(\mathbb{F}_p)|$ divides $p^r - 1$ small integer $r \geqslant 1$. In the latter groups, p^r must be sufficiently large so that the *general number field sieve* (GNFS) DLOG algorithm in $\mathbb{F}_{p^r}$ is infeasible. In practice, generating a curve on your own is not advised. Indeed, NIST recommends a collection of standardized curves such as the widely-used secp256r1, secp256k1, and Curve25519.

In the context of blockchain, the curve secp256k1 is well know as it is used for digital signatures in Bitcoin. It is present in the "standard for efficient cryptography" (acronym SEC2) [19] where the curve is defined over $\mathbb{F}_p$. The specification also defines the *elliptic curve domain parameters* given by the tuple (p, a, b, G, n, h) where: p prime specifies the field $\mathbb{F}_p$, $a, b \in \mathbb{F}_p$ are the constants in the curve equation $y^2 = x^3 + 7$ (here a is zero), G is the base point that generates the entire group $E(\mathbb{F}_p)$, n is the order of G and h the cofactor $h = |E(\mathbb{F}_p)|/n$. Because the prime p is close to 2^{256}, the number of points on the curve is also close to 2^{256}. Computing discrete log on these curves using a generic DLOG algorithm takes approximately 2^{128} group operations. In general, elliptic curves over $\mathbb{F}_p$ supply approximately t-bit security level, where $2t = \lceil \log_2 p \rceil$. This suggests that $(\log_2)$ the size of $E(\mathbb{F}_p)$ should be at least close to double t.

In our proof system prototype we use the *Barreto–Naehrig* elliptic curve (BN-curve) [12], defined over a 254-bit prime field and with estimated 128-bit security level. The BN-curve is the one precompiled in Ethereum blockchain [58] and can be found in the Ethereum source code under the name BN-256 [37]. The Zcash cryptocurrency also implemented the first Zerocash protocol [62] under the BN-curve. However, a variant of GNFS algorithm reduced the BN-curve security to around 110-bit security [49, 6]. For that reason, Zcash and Ethereum 2.0 switched to the *Barreto-Lynn-Scott* (BLS12-381) curve over a 381-bit field. We also encourage not to use the BN-curve in real-world applications or to use BN-curves over 384-bit prime fields.

Our instantiation of the BN-curve. The BN-curve is *pairing-friendly*, meaning that there exists a *bilinear map* from two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ to a target group $\mathbb{G}_T$. The pairing is denoted $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ and it allows for *pairing operations*. In our proof systems we do not use pairing operations, instead, we consider two mathematical structure combined: $\mathbb{Z}_p$ where *scalars* live and a cyclic group $\mathbb{G}$ of order p that is the elliptic-curve group. In particular, we only use the $\mathbb{G}_1$ group of the BN-curve, which is defined over a prime field $\mathbb{F}_p$ (see [7] for domain parameters). $\mathbb{G}_1$ is the set points satisfying the equation $y^2 = x^3 + 3$, forming a group under the addition rule and where the DLOG problem is hard. We use the *multiplicative notation* for group operations and scalar multiplications when specifying proof systems.

Moreover, elliptic-curve points can be represented in *affine coordinates* or *projective coordinates*. The adoption of one or the other representation leads to greater (or lesser) efficiency in elliptic-curve operations. Projective coordinates offer greater efficiency in point addition operations as they eliminate the need for computing inverses, a requirement in affine coordinates. However, we primarily use affine coordinates due to privacy concerns: expressing a point in projective coordinates may disclose information about its derivation, potentially compromising secret information [48]. Fortunately, transitioning between the two representations is straightforward, and we could implement both in the same system.

2.5 Cryptographic Primitives

This section provides the cryptographic primitives for encoding and encryption used in this dissertation. Such primitives are fundamental building blocks for proof systems.

2.5.1 Commitment schemes

Generally a commitment is used in the communication of values between two parties, a *committer* and a *verifier*. The committer commits to a specific value and the verifier can only accept or reject the resulting commitment after the original value has been revealed by the committer. During the whole process the committed value cannot be changed by the committer and the verifier has no ways to retrieve such value with only the commitment. A *commitment scheme* is usually a triple of PPT algorithms, the definition is given below.

Definition 10 (Commitment scheme)**.** *A commitment scheme is a triple of PPT algorithms* (`Setup`, `Commit`, `Verify`) *defined as follows:*

$pp \leftarrow \texttt{Setup}(1^\lambda)$ takes the security parameter λ and randomly generates the public parameters pp. We write these parameters implicit in the commit and verify algorithms: Commit(.) = Commit(.)$_{pp}$ and Verify(.) = Verify(.)$_{pp}$.

$com \leftarrow \texttt{Commit}(x; r)$ takes a value x from the message space and a randomness r taken uniformly, and outputs a commitment com.

$\{\texttt{accept}, \texttt{reject}\} \leftarrow \texttt{Verify}(x, com)$ takes the value x and the commitment com and outputs `accept` *or* `reject`*.*

The algorithm `Verify` can be used by the verifier based on the received value, the commitment and possibly together with extra data regarding the opening of the commitment. Then, the verifier states whether the claimed value of the committer is a valid opening for the associated commitment. We do not use the `Verify` algorithm since this reveals the original value, instead we consider a *non-interactive* commitment scheme in which a valid commitment is proved by the knowledge of the secret relating to the commitment.

The security properties of a commitment scheme are defined following the **binding** and **hiding** definitions.

Definition 11 (Binding commitment)**.** *A commitment scheme is (computationally) binding if, for all PPT adversaries $\mathcal{A}$, there exists a negligible function* $\mathbf{negl}(\lambda)$ *such that*

$$Pr\left[\begin{array}{l} pp \leftarrow Setup(1^\lambda),\ \{(x_b, r_b)\}_{b=0}^{1} \leftarrow \mathcal{A}(pp), \\ com_0 = Commit(x_0; r_0),\ com_1 = Commit(x_1; r_1)\ : \\ com_0 = com_1 \wedge (x_0, r_0) \neq (x_1, r_1) \end{array}\right] \leqslant \mathbf{negl}(\lambda)$$

where the probability is over Setup and $\mathcal{A}$. If $\mathbf{negl}(\lambda) = 0$*, we say that the scheme is perfectly binding.*

Definition 12 (Hiding commitment)**.** *A commitment scheme is (computationally) hiding if, for all PPT adversaries $\mathcal{A}$, there exists a negligible function* $\mathbf{negl}(\lambda)$ *such that*

$$Pr\left[\begin{array}{l} pp \leftarrow Setup(1^\lambda),\ (x_0, x_1) \leftarrow \mathcal{A}(pp), \\ b \xleftarrow{\$} \{0,1\},\ r \xleftarrow{\$} \mathbb{Z}_p,\ com = Commit(x_b; r), \\ b' \in \{0,1\} \leftarrow \mathcal{A}(pp, com)\ : \\ \quad b' = b \end{array}\right] \leqslant \frac{1}{2} + \mathbf{negl}(\lambda)$$

where the probability is over Setup, $\mathcal{A}$, b and r. If $\mathbf{negl}(\lambda) = 0$*, we say that the scheme is perfectly hiding.*

Roughly speaking, the binding property implies that it is hard to compute two openings of the same commitment, and the hiding property implies that it is hard to distinguish which message is committed.

We use the form of *Pedersen commitments* which can be defined over a finite cyclic group $\mathbb{G}$ of prime order. The definition is given below.

Definition 13 (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 large safe prime, and the randomly chosen blinding factor r belongs to the set $\mathbb{Z}_p^*$ of inverses in $\mathbb{Z}_p$. 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 commitment is defined as follows:*

$$Commit(m; r) := (g^m h^r) \in \mathbb{G}$$

Considering two Pedersen commitments such that com = Commit(m, r) and com$'$ = Commit(m', r'), an *additive homomorphism* is derived from the application of the group operator $\cdot$ between com and com$'$:

$$\begin{aligned} \text{com} \cdot \text{com}' &= \text{Commit}(m, r) \cdot \text{Commit}(m', r') \\ &= g^{m+m'} \cdot h^{r+r'} \\ &= \text{Commit}(m + m', r + r') \end{aligned}$$

where, $m, m' \in \mathbb{Z}_p$, $r, r' \xleftarrow{\$} \mathbb{Z}_p^*$, and $g, h, \text{com}, \text{com}' \in \mathbb{G}$.

We can also commit to multiple values at once using the *Pedersen vector commitment* variant. Here, messages and generators are gathered in vectors of size n and the commitment is computed as:

$$\begin{aligned} \text{Commit}(\mathbf{m}, r) &= \text{Commit}((m_1, \ldots, m_n), r) \\ &= g_1^{m_1} \cdot \ldots \cdot g_n^{m_n} \cdot h^r \\ &= \textstyle\prod_{i=1}^{n} g_i^{m_i} \cdot h^r \\ &= \mathbf{g}^{\mathbf{m}} \cdot h^r \end{aligned}$$

where $\mathbf{m} = (m_1, \ldots, m_n) \in \mathbb{Z}_p^n$, $\mathbf{g} = (g_1, \ldots, g_n) \in \mathbb{G}^n$, $h \in \mathbb{G}$, and $r \xleftarrow{\$} \mathbb{Z}_p^*$.

Pedersen commitments are computationally binding under DLOG assumption and perfectly hiding.

2.5.2 Homomorphic cryptosystems

ElGamal encryption [35] is a public key cryptosystem that is secure under the DDH assumption. Considering a cyclic group $\mathbb{G}$ of prime order p and

generator g 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 element 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 this dissertation, we use the specific instance of the ElGamal encryption of Zether [20]. In particular, the message to be encrypted is a value $a \in \mathbb{Z}_p$, which can be turned into a group element using the simple mapping $m = g^a$. Consequently, the encryption of a message can be rewritten as

$$Enc_y(a) = (C_L, C_R) = (g^a \cdot y^r, g^r).$$

Moreover, we leverage the *additive homomorphic* property of the Elgamal encryption. Given two distinct values a and a', encrypting them under the same public key y and respectively under the randomness r and r', we obtain the two ciphertexts:

$$(C_L = g^a \cdot y^r, C_R = g^r),$$

$$(C'_L = g^{a'} \cdot y^{r'}, C'_R = g^{r'}).$$

By applying the operation $(C_L, C_R) \cdot (C'_L, C'_R)$, it is possible to derive the ciphertext associated with $a + a'$. Indeed,

$$(C_L \cdot C'_L = g^{a+a'} \cdot y^{r+r'}, C_R \cdot C'_R = g^{r+r'}).$$

This permits to perform arithmetic operations between values without revealing the values, thus making the ElGamal cryptosystem a candidate in a variety of privacy-oriented scenarios. In chapter 3, we exploit this additive homomorphism for balance updates and arithmetic operations between amounts when they are encrypted under the same public key.

2.6 Proof Systems

In this section we lay the foundations of interactive proofs and interactive zero-knowledge proofs.

2.6.1 Interactive proof systems

We consider randomized and interactive procedures for the notion of *interactive proof systems*, rather than being non-interactive and deterministic. In our context a *proof* is a randomized process, in which the verifier interacts with the prover. This interaction can be envisioned as questions asked by the verifier, to which the prover must respond convincingly. Thus, prover and verifier are the interactive players of an interactive proof: the prover is the party that tries to convince the verifier about the validity of a proof; the verifier is the party that employs the verification procedure of that proof. We want to stress that if the verifier trusts the prover, then no proof is needed (the verifier is skeptical of everything the prover says).

We also have an asymmetry regarding the complexity of prover and verifier. The verifier is always required to be efficient, while the prover can have unbounded computational power. This asymmetry is evident in the definition of $\mathcal{NP}$-proofs, where the verification must be executed within polynomial time. Thus, as a first approximation, we view an interactive proof as conversation between computationally bounded verifier and unbounded prover, where the prover's goal is to prove that some assertion is valid. Specifically, the verifier is a *probabilistic polynomial-time* (PPT) *algorithm*, while no computational restrictions are given for the prover.

It is required that if the assertion holds then the verifier always accepts, i.e., when interacting with an honest prover. This situation is captured by the *completeness* property of an interactive proof, which states the ability of the prover to convince the verifier of true proofs. Note that this is the *perfect completeness* variant where probability that the verifier "always accepts" means equal to 1. A more general definition of completeness requires that the verifier accepts valid proofs with high probability, i.e., at least 2/3. On the other hand, if the assertion is false then the verifier must reject with high probability. Here, this condition is captured by the *soundness* property which states that the verifier cannot be convinced of false proofs.

We can formalize the interaction between prover and verifier by referring to the algorithms that prover and verifier employ. We can see an algorithm as a function mapping the view of prover and verifier of the interaction so far to a description of their next move. Such an algorithm prescribes the next message or final decision as a function of the common input, the party's internal coin tosses, and all messages it has received so far. Such formulation presumes that each party records the past coin tosses as well as all messages received so far, and determines the next move based on these. For example, if we have prover and verifier employing algorithm $\mathcal{P}$ and $\mathcal{V}$ respectively, their interaction is determined by the common input x, the randomness of both, say $r_{\mathcal{P}}$ and $r_{\mathcal{V}}$. Assuming that $\mathcal{V}$ takes the first move, and $\mathcal{P}$ the last move, the t-round *transcript* is $\alpha_1, \beta_1 \ldots \alpha_t, \beta_t$ where

$\alpha_i = \mathcal{V}(x, r_\mathcal{V}, \beta_1, \ldots, \beta_{i-1})$ and $\beta_i = \mathcal{P}(x, r_\mathcal{P}, \alpha_1, \ldots, \alpha_i)$. The final decision of $\mathcal{V}$ is defined as $\mathcal{V}(x, r_\mathcal{V}, \beta_1, \ldots, \beta_t)$. We say that an algorithm is PPT if its next move can be computed in a number of steps that is polynomial in the length of the common input x. Moreover, the algorithm only consider a number of messages, as well as the number of internal coin tosses, bounded by a polynomial in $|x|$. We now give the generic definition of *interactive proofs*.

Definition 14 (Interactive Proofs)**.** *An interactive proof for a set $\mathcal{S}$ is a two-party conversation, between a PPT verifier algorithm, denoted $\mathcal{V}$, and a computationally unbounded prover algorithm, denoted $\mathcal{P}$, satistfying the following two conditions:*

- Completeness*: For every $x \in \mathcal{S}$, the verifier $\mathcal{V}$ always accepts after interacting with the prover $\mathcal{P}$ on common input x:*

$$Pr[\langle \mathcal{P}, \mathcal{V} \rangle(x) = 1] = 1$$

- Soundness*: For every $x \notin \mathcal{S}$ and every algorithm $\mathcal{P}^*$, the verifier $\mathcal{V}$ rejects with high probability after interacting with $\mathcal{P}^*$ on common input x:*

$$Pr[\langle \mathcal{P}^*, \mathcal{V} \rangle(x) = 1] \leqslant \frac{1}{3}$$

Here, $\langle \cdot, \cdot \rangle$ denote interaction and output "1" is interpreted as "verifier accepts". We denote by $\mathcal{IP}$ the class of sets having interactive proof systems.

An important variant for the soundness definition is the *computational soundness*. Here, we allow the prover $\mathcal{P}^*$ to be *computationally bounded* and, therefore, a PPT algorithm. Interactive proof systems that relax the soundness property in this way are known as **argument systems**. Moreover, argument systems can result in a more efficient communication complexity.

The role of randomness in $\mathcal{IP}$ class is clearly important. $\mathcal{NP}$-proofs are obtained as a special case of $\mathcal{IP}$s by eliminating interaction and randomness, i.e., restricting the verifier to a deterministic algorithm. Thus, without randomness a verifier strategy yields a class of $\mathcal{IP}$ that has no advantage over $\mathcal{NP}$-proofs. Moreover, the prover is able to predict the interaction of a deterministic verifier: the prover supplies its own answers to the verifier's questions and the verifier concludes by just checking that these predictable answers are convincing. Looking at the definition (14), the *soundness error* stands out, which is essential to the power of $\mathcal{IP}$s. While it is natural to allow the use of randomization, it is natural to allow also a probability of error, which means that these verification procedures may rule by statistical evidence. Clearly, this probability of error is explicitly bounded. The following reinforces the need for soundness error. Suppose that a set $\mathcal{S}$ has an

interactive proof $(\mathcal{P}, \mathcal{V})$ with no soundness error; that is, for every $x \notin \mathcal{S}$ and every potential $\mathcal{P}^*$, the verifier $\mathcal{V}$ rejects with probability 1 after interacting with $\mathcal{P}^*$ on common input x. Then, $\mathcal{S} \in \mathcal{NP}$ [42].

We can also generalize the soundness error introducing the *negligible soundness error*: a proof system is said to have negligible soundness error if for every $x \notin \mathcal{S}$ and every $\mathcal{P}^*$,

$$Pr[\langle \mathcal{P}^*, \mathcal{V} \rangle(x) = 1] < \mu(|x|),$$

for some negligible function $\mu(.)$.

We now briefly explain some renowned outcomes within the class of $\mathcal{IP}$ sets. The strongest evidence supporting the belief that interactive proofs are more powerful than $\mathcal{NP}$-proofs comes from the celebrated result $\mathcal{IP} = \mathcal{PSPACE}$ [68]. This result presents an interactive proof for the $\mathcal{PSPACE}$-complete set of unsatisfiable QBF formulae. Towards this result, there are notable works showing that $co\mathcal{NP} \subseteq \mathcal{IP}$, whereas, it is widely believed that some sets in $co\mathcal{NP}$ do not have $\mathcal{NP}$-proofs (i.e., $\mathcal{NP} \neq co\mathcal{NP}$). To name a few, we have the interactive proof for the language of *non-isomorphic graph* [43] and the interactive proof for the $co\mathcal{NP}$-complete set of unsatisfiable CNF formulae [52].

2.6.2 Interactive Zero-Knowledge

In section 2.6.1, we saw a general definition of interactive proof, i.e., definition (14). To be more concrete, the set $\mathcal{S}$ of valid claims and the specific assertion x are also referred to as *language* and *statement* respectively. Thus, following the definition, a prover claims that a statement x is in the language L. For $\mathcal{NP}$ languages, let $\mathcal{R}$ be a witness relation recognizable in polynomial-time and associated with $L \in \mathcal{NP}$, formally

$$L = \{x : \exists w \ s.t. \ |w| = poly(|x|) \wedge (x, w) \in \mathcal{R}\} \ ,$$

now $x \in L$ implies that the statement x admits an efficiently verifiable *witness* w. Then an interactive proof for the language L consists of a prover that on input the statement x provides a proof for which the verifier accepts that $(x, w) \in \mathcal{R}$, i.e., $\mathcal{V}(x; w) = 1$. This allows the prover to convince a verifier of the existence of the witness w. In many applications, the prover $\mathcal{P}$ wants to convince the verifier $\mathcal{V}$ without disclosing any information other than the validity of the statements being proved. In particular, an interactive proof should not reveal any additional information about the secret witness w; that is, a verifier obtaining a convincing proof only gains knowledge about the validity of the statement. Interactive proofs with this property are said to be *zero-knowledge* and are also called *zero-knowledge proofs* (ZK-proofs). The zero-knowledge property formulation follows the so-colled **simulation**

paradigm. Here, we consider the verifier as a potential adversary trying to gain "extra" knowledge from the honest prover. The main observation is that the adversary "gains nothing" if whatever it can obtain by deviating from the prescribed behavior can be obtained within the same computational effort by an honest behavior. Specifically, an honest behavior is any computation that is based only on the statement itself. The interpretation of "gain nothing" means that any feasible adversarial behavior can be "simulated" by honest behavior. This suggests the following strategy in defining zero-knowledge: we can observe if there exists a PPT algorithm, called *simulator*, that takes as input only the statement x and outputs a conversation (transcript) with a distribution indistinguishable from that of the conversation generated with an honest prover. The existence of a simulator shows that a dishonest verifier $\mathcal{V}^*$ can generate transcripts without interacting with an honest prover $\mathcal{P}$ and, hence, that honest conversations do not reveal any information that $\mathcal{V}^*$ could not have obtained on its own. We are now ready to define the zero-knowledge property for interactive proofs.

Definition 15 (Zero-Knowledge)**.** *An interactive proof between $\mathcal{P}$ and $\mathcal{V}$ over a language L is said to be zero-knowledge if, for every (potentially dishonest) PPT verifier $\mathcal{V}^*$, there exists a PPT simulator $\mathcal{S}^*$ that, on input the public statements, outputs transcripts such that the distribution over transcripts generated by $\mathcal{S}^*$ is identical to the distribution over the transcripts generated by $\mathcal{P}$ and $\mathcal{V}^*$. Formally:*

$$(\mathcal{P}, \mathcal{V}^*)(x) = \mathcal{S}^*(x), \quad \textit{for every } x \in L$$

where $(\mathcal{P}, \mathcal{V}^*)(x)$ is the output distribution of $\mathcal{V}^*$ after interacting with $\mathcal{P}$ on common input x, and $\mathcal{S}^*(x)$ is the output distribution of the simulator $\mathcal{S}^*$ on input x.

When the two distribution are identical as defined in (15), the interactive proof is said to be *perfectly zero-knowledge*, when instead they are *computationally indistinguishable*, the interactive proof is said to be *computationally zero-knowledge*. In the latter case, it is convenient to make explicit the statistical distance between the distributions of the definition (15):

$$\Delta((\mathcal{P}, \mathcal{V}^*)(x), \mathcal{S}^*(x)) \leqslant \mu(|x|), \quad \text{for every } x \in L,$$

where μ is a negligible function.

We will see an alternative notion of zero-knowledge: *honest-verifier zero-knowledge* (HVZK). The difference is that this notion requires a simulator for the honest verifier, i.e., a simulator whose distributions over the transcripts are statistically close to those of the transcripts of honest executions of the interactive proofs. Although it is a weaker notion of zero-knowledge,

generic transformations exist for certain systems, such as *public-coin* interactive proofs, to convert them into zero-knowledge interactive proofs [32, 31]. Moreover, public-coin interactive proofs can be made non-interactive through the so-called Fiat-Shamir transform [39]. Since the HVZK property is sufficient for non-interactive zero-knowledge proofs, it is often enough to prove that an interactive proof is HVZK.

2.7 Interactive Protocols

In most cases, the terms "interactive proofs" are used interchangeably with "interactive protocols". This often happens in theoretical literature, particularly when discussing concepts like the $\mathcal{IP}$ complexity class. However, in applied cryptography literature, "interactive protocols" are the most prevalent terms. Therefore, we permit this interchange of terms, but with "interactive protocols" we emphasize interactive proofs in the applied context. In this section, we outline an important family of interactive protocols, the **Σ-protocols**. These interactive protocols derive from the generalization of their progenitor, the *Schnorr's protocol* [67]. We organize the section as follows: in section 2.7.1 the Schnorr's protocol is presented to highlight some technical terms in the context of interactive protocols; in section 2.7.2 we present the generalization of the Schnorr's protocol, namely the Σ-protocols, along with definitions for *zk-relations*, the general Σ-protocol and for *perfect completeness*; in section 2.7.3 and 2.7.4 we provide the important security definitions of *special-soundness* and *special honest-verifier zero-knowledge* for Σ-protocols.

2.7.1 Schnorr's protocol

The Schnorr's protocol is designed to address the *identification problem*, with its security derived from the assumption that the DLOG problem is hard. Let $\mathbb{G}$ be a cyclic group of prime order p with generator $g \in \mathbb{G}$. Suppose the prover $\mathcal{P}$ has a secret key $\alpha \in \mathbb{Z}_p$, and the corresponding public key is $u = g^\alpha \in \mathbb{G}$. The Schnorr's protocol allows $\mathcal{P}$ to convince $\mathcal{V}$ that he knows the discrete logarithm α of u to the base g (discrete $\log_g u$), without actually sending this value to $\mathcal{V}$. In other words $\mathcal{P}$ wants to convince $\mathcal{V}$ that he knows α to prove his identity (prover's statement). Let $\mathbb{C} \subset \mathbb{Z}_p$, the Schnorr's protocol is a 3-round interactive protocol $\Pi_{sch} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$, that proceeds as follows:

- At the `Setup`, the *group-generation algorithm* $\mathcal{G}$ is executed: $(\mathbb{G}, p, g) \leftarrow \mathcal{G}(1^\lambda)$, with λ the *security parameter*. Then, the secret

key $\alpha \xleftarrow{\$} \mathbb{Z}_p$ is randomly chosen and the public key $u = g^\alpha \in \mathbb{G}$ is computed.

- Next, prover $\mathcal{P}(u, g; \alpha)$ and verifier $\mathcal{V}(u, g)$ engage in a 3-round protocol.
- In the first round, $\mathcal{P}$ computes $\alpha_t \xleftarrow{\$} \mathbb{Z}_p$ random element and $u_t \leftarrow g^{\alpha_t} \in \mathbb{G}$. Then, $\mathcal{P}$ sends u_t to $\mathcal{V}$.
- In the second round, $\mathcal{V}$ computes the *random challenge* $c \xleftarrow{\$} \mathbb{C}$ and sends it to $\mathcal{P}$.
- In the third round, $\mathcal{P}$ computes $\alpha_z \leftarrow \alpha_t + \alpha c \in \mathbb{Z}_p$, and sends this final message α_z to $\mathcal{V}$.
- Finally, $\mathcal{V}$ checks that $g^{\alpha_z} \stackrel{?}{=} u_t \cdot u^c$. If the equality holds, $\mathcal{V}$ outputs `accept`, otherwise `reject`.

An interaction between $\mathcal{P}$ and $\mathcal{V}$ generates a conversation $(u_t, c, \alpha_z) \in \mathbb{G} \times \mathbb{C} \times \mathbb{Z}_p$. We call this *accepting conversation* for u if $\mathcal{V}$'s check passes. It is easy to be convinced that the protocol is correct, i.e., that an interaction between $\mathcal{P}$ and $\mathcal{V}$ generates an accepting conversation:

$$g^{\alpha_z} \stackrel{?}{=} u_t \cdot u^c \implies g^{\alpha_t + \alpha c} = g^{\alpha_t} \cdot (g^\alpha)^c = u_t \cdot u^c$$

Moreover, the protocol is *sound* and *HVZK*. We now prove these properties following a proof strategy similar to [15]. For the *soundness*, we have to show that there exists an algorithm that efficiently recovers the secret key α from the public key u. This algorithm is called *extractor*, and by showing the existence of the extractor, we prove that Schnorr's protocol is *proof of knowledge* of a discrete logarithm. Assuming $DLOG$ assumption for $\mathbb{G}$ is hard, and assuming the challenge space size $N := |\mathbb{C}|$ is super-poly, we can compute the $DLOG$ of a random $u \in \mathbb{G}$ using the *rewind technique*, as discussed below.

Let u as defined in Π_{sch}. Let $\mathcal{P}$ generate the first flow u_t of the protocol. We supply $\mathcal{P}$ a random challenge c and hope $\mathcal{P}$ generates a valid response α_z. If this happens, we rewinds $\mathcal{P}$ to the point just after which it generates u_t and supply $\mathcal{P}$ with another random challenge c', hoping that $\mathcal{P}$ generates another valid response α'_z. If this happens, we obtain two accepting transcripts (u_t, c, α_z) and (u_t, c', α'_z) for a given u and with matching first message u_t. Moreover, with overwhelming probability, we have $c \neq c'$ from the assumption that $\mathbb{C}$ is super-poly. Now, we can easily compute (discrete) $\log_g u$ by formulating the extractor. Since we have two accepting transcripts, we have two equations:

$$g^{\alpha_z} = u_t \cdot u^c \text{ and } g^{\alpha'_z} = u_t \cdot u^{c'}$$

Putting the two equality together, we have:

$$\frac{g^{\alpha_z}}{g^{\alpha'_z}} = \frac{u_t \cdot u^c}{u_t \cdot u^{c'}} = g^{\alpha_z - \alpha'_z} = u^{c-c'}$$

Since $\Delta c = c - c' \neq 0$ and the group order p is prime, the inverse $1/\Delta c$ exists in $\mathbb{Z}_p$. Hence, we can do:

$$g^{(\alpha_z - \alpha'_z)\cdot(c-c')^{-1}} = u$$

$$\alpha \equiv (\alpha_z - \alpha'_z) \cdot (c - c')^{-1} \bmod |\mathbb{Z}_p|$$

meaning that we can efficiently compute (discrete) $\log_g u$ as $\Delta\alpha/\Delta c$, where $\Delta\alpha := \alpha_z - \alpha'_z$ and $\Delta c := c - c'$. This proves that the Schnorr's protocol respects the soundness of an interactive proof.

We now show that Schnorr's protocol Π_{sch} is HVZK by constructing the simulator as required in the zero-knowledge definition (15), but in the case of *honest verifier*. In general, we say that Π_{sch} is HVZK if there exists a probabilistic algorithm S, called simulator, such that the output distribution of S on input the public statement is identical to the distribution of a transcript between $\mathcal{P}$ and $\mathcal{V}$. Consequently, any possible adversary gains no information from $\mathcal{P}$, as they can simulate conversations on their own using S, without knowing the witness. The fact that we observe the honest verifier case means that this works for conversations between $\mathcal{P}$ and the actual honest verifier $\mathcal{V}$.

Now for showing that Π_{sch} is HVZK, we build S that generates messages of a transcript between real $\mathcal{P}$ and $\mathcal{V}$. On input u, S computes:

$$\alpha_z \xleftarrow{\$} \mathbb{Z}_p,\ c \xleftarrow{\$} \mathbb{C},\ u_t \leftarrow g^{\alpha_z} \cdot u^{-c}$$

and outputs the transcript (u_t, c, α_z).

We argue that S on input u has the right distribution. Indeed, in real interaction between $\mathcal{P}$ and $\mathcal{V}$, c and α_z are independent (c uniformly distributed over $\mathbb{C}$ and α_z uniformly distributed over $\mathbb{Z}_q$); moreover, given c and α_z the value u_t is uniquely distributed by the equation $g^{\alpha_z} = u_t \cdot u^c$. It is clear that this is the same as the output distribution of S. One last consideration is that Π_{sch} would be insecure if the challenge space were small. Indeed, any adversary could use S to prepare an accepting transcript (u_t, c, α_z), send u_t to $\mathcal{V}$ and then hope that the challenge chosen by $\mathcal{V}$ match the prepared challenge c. If so, any adversary could respond with α_z and so make $\mathcal{V}$ accept. In this situation any adversary has success of about $1/|\mathbb{C}|$,

therefore the challenge space must be super-poly to ensure security.

The Schnorr's protocol is therefore an interactive proof $(\mathcal{P}, \mathcal{V})$ that satisfies the following (DLOG) relation:

$$\mathcal{R} = \{(u, g \in \mathbb{G};\ \alpha \in \mathbb{Z}_p) :\ g^\alpha = u\} \tag{2.2}$$

where the prover's statement is $u = g^\alpha$ for a secret witness α. Note that relation (2.2) does not specify the group-generation algorithm $\mathcal{G}$, and also considers the description of the group $\mathbb{G}$ along with the generator $g \in \mathbb{G}$ and its order. From now on, we allow these relations to be defined on such *system parameters*, which are assumed to be publicly known to each party and generated once at setup time.

2.7.2 Σ-Protocols

In this section, we generalize the *Schnorr's protocol* of section 2.7.1, which can be seen as a Σ-Protocol for the problem of proving the knowledge of a secret discrete logarithm. Σ-Protocols are a class of *interactive protocols* well established also in the context of zero-knowledge proofs. They can be used to build advanced and secure identification and signature schemes, as well as to prove statements over encrypted messages to a skeptical verifier. Such statements can be proved in zero-knowledge, meaning we can prove that a message satisfies certain properties, without revealing anything to the verifier except the validity of the statements.

Now we contextualize some concepts introduced in section 2.3.2 in the context of Σ-protocols. In general, an interactive protocol can run among multiple parties, whereas Σ-protocols focus on *two-party protocols*. An interactive protocol may run many times (from start to end) and each such run is called *protocol instance*. Parties start a protocol instance in some initial *configuration*, composed by inputs and system parameters. Then, parties send and receive messages and run some computations to updated their local configurations. Such computations are called *interactive protocol algorithms* and modeled as efficient probabilistic algorithms. The configuration of a protocol instance is encoded as a bit-string of arbitrary length, e.g., $x \in \{0,1\}^*$. Interactive protocol algorithms are often preceded by a *setup* algorithm, which typically takes the security parameter $\lambda \in \{0,1\}^*$ and is executed once for all protocol instances to generate system parameters (e.g., the group description and interactive proof setup parameters). The *configuration size* is always bounded by some fixed polynomial in λ, i.e., the length of the inputs is bounded by some $p(|\lambda|)$. Efficient interactive protocol algorithms require that the length of any input, and also of any produced output, is poly-bounded. Hence, we consider not only that computations run

in poly-time but also that the configuration size remains poly-bounded.

Before formally defining the Σ-Protocols, we give a useful definition for **zero-knowledge relations**:

Definition 16 (Zero-knowledge relation)**.** *A **relation** is a binary relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$, where $\mathcal{X}$, $\mathcal{W}$, and $\mathcal{R}$ are finite sets. Elements $\mathbf{x} = \{x_1, ..., x_n\}$ of $\mathcal{X}$ are called **instances** and $\mathbf{w} = \{w_1, ..., w_n\}$ of $\mathcal{W}$ **witnesses**. A relation $\mathcal{R}$ formally specifies some **statements** as a function $f(\mathbf{x}, \mathbf{w})$ which are satisfied if and only if $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}$. We use the notation*

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

*to specify a relation for an interactive proof between prover $\mathcal{P}$ and verifier $\mathcal{V}$, where elements in $\mathbf{x}$ are public and are known to both $\mathcal{P}$ and $\mathcal{V}$, while those in $\mathbf{w}$ are only known to $\mathcal{P}$. We say that the relation is **zero-knowledge** if $\mathcal{P}$ convinces $\mathcal{V}$ that the statements are true, without revealing information about $\mathbf{w}$.*

We now give a general definition for **Σ-protocol**, revised from [15]:

Definition 17 (Σ-protocol)**.** *Let $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$ be a binary relation. A **Σ-protocol** for $\mathcal{R}$ is an interactive protocol $\Pi = (\mathcal{P}, \mathcal{V})$ where:*

- *$\mathcal{P}$ is an interactive protocol algorithm called **prover**, which takes as input an instance-witness pair $(x, w) \in \mathcal{R}$.*
- *$\mathcal{V}$ is an interactive protocol algorithm called **verifier**, which takes as input an instance $x \in \mathcal{X}$ and outputs `accept` or `reject`.*
- *The interactive protocol between $\mathcal{P}$ and $\mathcal{V}$ is structured so that it always works as follows:*
  - *$\mathcal{P}$ starts the protocol by computing a message a, called **announcement**, and sends a to $\mathcal{V}$;*
  - *Upon receiving $\mathcal{P}$'s announcement a, $\mathcal{V}$ chooses a **challenge** c at random from a finite **challenge space** $\mathcal{C}$, and sends c to $\mathcal{P}$.*
  - *Upon receiving $\mathcal{V}$'s challenge c, $\mathcal{P}$ computes a **response** z, and sends z to $\mathcal{V}$.*
  - *Upon receiving $\mathcal{P}$'s response z, $\mathcal{V}$ outputs `accept` or `reject`. The $\mathcal{V}$'s output must be computed strictly as a function of the instance x and the conversation (a, c, z). In particular, all $\mathcal{V}$ computations are completely deterministic except the random choice of the challenge.*

We require that for all $(x, w) \in \mathcal{R}$, when $P(x, w)$ and $V(x)$ interact and follow the prescribed protocol, $V(x)$ always outputs `accept`.

The *generalized structure* of any interactive Σ-protocol is given in protocol 1.

Protocol 1 Generalized Σ-protocol

1: Let $\mathcal{P}(x, w)$ be a prover on input $(x, w) \in \mathcal{R}$.
2: Let $\mathcal{V}(x)$ be a verifier on input the instance x.

3: $\mathcal{P}$: computes *announcement* a
4: $\mathcal{P} \rightarrow \mathcal{V} : a$
5: $\mathcal{V}$: generates *challenge* $c \xleftarrow{\$} \mathcal{C}$
6: $\mathcal{V} \rightarrow \mathcal{P} : c$
7: $\mathcal{P}$: computes *response* z
8: $\mathcal{P} \rightarrow \mathcal{V} : z$
9: $\mathcal{V}$: outputs `accept` or `reject`

What is shown in protocol 1 is a generalized 3-round Σ-protocol. When multiple protocol instances are executed, it results in a **multi-round Σ-protocol**. The simple structure of Σ-protocols allows them to be easily integrated into large interactive proofs.

Following the definition (17), if the verifier $\mathcal{V}$ outputs `accept` we call the *conversation* (a, c, z) an accepting transcript for x. Interactions between the verifier and an *honest prover* produce accepting transcripts; this suggests how to verify the **correctness** of a conversation between prover and verifier. We can generalize the above concept with the definition of **perfect completeness** for any Σ-protocol.

Definition 18 (Perfect completeness)**.** *Let $(\mathcal{P}, \mathcal{V})$ be a Σ-protocol for relation $\mathcal{R}$. If the prover $\mathcal{P}$ follows the protocol then the verifier $\mathcal{V}$ will accept with probability 1.*

Of course, non-accepting transcripts can occur if the verifier interacts with a "dishonest" prover who does not follow the protocol. To prevent this, well-defined security properties for **soundness** are given in section 2.7.3. Furthermore, it is often required that the size of the challenge space is **super-poly**, or commonly said that Σ-protocols must have a **large challenge space**.

2.7.3 Σ-Protocols security: special-soundness

In this section we define the fundamental security properties for the soundness of Σ-Protocols, valid for both standard 3-round and multi-round protocols.

Definition 19 (k-Special-Soundness)**.** *Let* $\Pi = (\mathcal{P}, \mathcal{V})$ *be a* Σ*-protocol for relation* $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$*. We say that* Π *is k-special-sound if there exists a polynomial-time deterministic algorithm* $\mathcal{E}$*, called witness* ***extractor****, which is given as input an instance* $x \in \mathcal{X}$ *and* k *accepting transcripts* (a, c_1, z_1)*,* ...,(a, c_k, z_k) *with a the common first* $\mathcal{P}$*'s message,* $c_1, ..., c_k$ *pairwise distinct* $\mathcal{V}$*'s challenges,* $z_1, ..., z_k$ *the final* $\mathcal{P}$*'s messages, and always outputs a witness* $w \in \mathcal{W}$ *satisfying* $(x, w) \in \mathcal{R}$*, i.e.,* w *is a witness for* x*. When* $k = 2$*, it is simply said that* Π *is special-sound.*

Suppose $\Pi = (\mathcal{P}, \mathcal{V})$ is a *special-sound* Σ-protocol for relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$ that has a large challenge space, we say that Π acts as a **proof of knowledge**. Indeed, consider an arbitrary $\mathcal{P}^*$ (even a potentially "dishonest" one) that makes $\mathcal{V}$ accept a statement for x with non-negligible probability. Then $\mathcal{P}^*$ must "know" a witness for x in the following sense: an extractor $\mathcal{E}$ is allowed to control $\mathcal{P}^*$ in such a way that after $\mathcal{P}^*$ sends the first message a, the extractor $\mathcal{E}$ rewinds $\mathcal{P}^*$ with a new challenge c' to get two accepting transcripts (a, c, z) and (a, c', z') for x, with $c \neq c'$, and then $\mathcal{E}$ extracts the witness w satisfying $(x, w) \in \mathcal{R}$. Hence, when we say that $\mathcal{P}^*$ must "know" a witness for x, we mean that the witness can be extracted from $\mathcal{P}^*$ using rewinding.

The notion of special-soundness is weaker than that known as *knowledge-soundness*. Also in the knowledge-soundness a (possible dishonest) prover $\mathcal{P}^*$ must know a witness w such that $(x, w) \in \mathcal{R}$, but this time the extractor only has **oracle access** to $\mathcal{P}^*$ (it should extract a witness by interacting with a prover in a black-box manner), while the special-soundness also gets the transcripts directly. For 3-round protocols, i.e., the standard Σ-protocols, it is known that the special-soundness implies knowledge-soundness. For multi-round Σ-protocols this implication is not straightforward. Indeed, in [9] it is shown that special-soundness tightly implies knowledge soundness for multi-round protocols. Essentially, a multi-round Σ-protocol is a $(2\mu + 1)$-round interactive protocol where the verifier sends μ challenges, instead of one challenge. The special-soundness definition for multi-round Σ-protocol is a generalization of definition (19) and is given below (for a more details see [17]).

Definition 20 ($(k_1, ..., k_\mu)$-Special-Soundness)**.** *Let* $\Pi = (\mathcal{P}, \mathcal{V})$ *be a* $(2\mu+1)$*-round* Σ*-protocol with* μ *verifier's challenges and for relation* $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$*. We say that* Π *is* $(k_1, ..., k_\mu)$*-special-sound if there exists a polynomial-time deterministic algorithm* $\mathcal{E}$*, called witness* ***extractor****, which is given as input an instance* $x \in \mathcal{X}$ *and a* $(k_1, ..., k_\mu)$*-tree of accepting transcripts and always outputs a witness* $w \in \mathcal{W}$ *satisfying* $(x, w) \in \mathcal{R}$*, i.e.,* w *is a witness for* x*.*

For a $(2\mu + 1)$-round Σ-protocol with μ verifier's challenges, a $(k_1, ..., k_\mu)$-tree of transcripts has the definition below.

Definition 21 ($(k_1, ..., k_\mu)$-tree)**.** *A $(k_1, ..., k_\mu)$-tree of transcripts is a set of $K = \prod_{i=1}^{\mu} k_i$ transcripts that are in a tree structure. The structure has nodes corresponding to the prover's messages (which can be empty) and edges corresponding to the verifier's challenges. Every node at depth i has exactly k_i children corresponding to the k_i pairwise distinct challenges. The tree has precisely K paths, each one is an accepting trascript, from the leaves to the root.*

An alternative notion of knowledge soundness is given in [51], which introduced the notion of **witness extended emulation**. Informally, an interactive protocol is said to have witness extended emulation if for any (possibly dishonest) prover $\mathcal{P}^*$ there exists an efficient (*expected polynomial-time*) algorithm that, on input a statement and with rewindable oracle access to $\mathcal{P}^*$, not only outputs a witness w but also a transcript that is required to be indistinguishable from a conversation between $\mathcal{P}^*$ and an honest verifer. Moreover, the probability that the transcript is accepting and that the witness is not valid is negligible (or in other words, if the transcript is accepting then this algorithm outputs a witness with overwhelming probability).

Such an algorithm is called a *witness extended emulator* and interactive protocols that have witness extended emulation are also proofs of knowledge. In [17], it is shown that that multi-round spacial-soundness implies *witness-extended emulation.* In this dissertation, we focus on $(k_1, ..., k_\mu)$-special-soundness protocols with some μ (challenges) and some set of k_i's (transcripts). Then, from [17] it follows that our protocols are *proof of knowledge.*

The following definition is useful when composing interactive proofs. The definition is revised from [9], and informally states that two interactive proofs Π_1 and Π_2, for relations $\mathcal{R}_1$ and $\mathcal{R}_2$ respectively, are composable if the final move of Π_1 is replaced with the protocol Π_2. Moreover, the verifier of Π_1 accepts if and only if the prover's final message of Π_1 is a witness for the statement relative to relation $\mathcal{R}_2$.

Definition 22 (Composable interactive proofs)**.** *Let Π_1 for relation $\mathcal{R}_1$ and Π_2 for relation $\mathcal{R}_2$ be two interactive proofs with $2\mu_1 + 1$ and $2\mu_2 + 1$ rounds respectively. Then, Π_1 and Π_2 are composable if, for an efficient computation ψ, the transcript $(\alpha_1, c_1, \alpha_2, ..., c_{\mu_1}, \alpha_{\mu_1+1})$ of Π_1 for statement x_1 is accepting if and only if the prover's final message α_{μ_1+1} is a witness for statement $x_2 = \psi(\alpha_1, c_1, \alpha_2, ..., c_{\mu_1})$ and $(x_2; \alpha_{\mu_1+1}) \in \mathcal{R}_2$. If the verifier of Π_2 accepts the proof for $\mathcal{R}_2$, then the composition $\Pi = \Pi_2 \diamond \Pi_1$ is accepted.*

A lemma (3.1 of [9]) follows for the special-soundness of composable interactive proofs. The lemma states that if Π_1 is $\mathbf{k}_1$-special-sound and Π_2 is $\mathbf{k}_2$-special-sound, where $\mathbf{k}_1 = (k_1, ..., k_{\mu_1})$ and $\mathbf{k}_2 = (k_1, ..., k_{\mu_2})$, then $\Pi = \Pi_2 \diamond \Pi_1$ is $(\mathbf{k}_1, \mathbf{k}_2)$-special-sound.

2.7.4 Σ-Protocols security: zero-knowledge

In section 2.6.2, we saw the general definition of (interactive) zero-knowledge proofs. Here, we provide the alternative definition that we use in our interactive proofs, suitable for Σ-protocols: *(special) honest-verifier zero-knowledge* (SHVZK).

Definition 23 ((Special) Honest-Verifier Zero-Knowledge)**.** *An interactive proof between $\mathcal{P}$ and $\mathcal{V}$ over a language L is said to be (special) honest-verifier zero-knowledge if there exists a PPT simulator $\mathcal{S}$ that, on input the public statements, outputs transcripts such that the distribution over transcripts generated by $\mathcal{S}$ is identical to the distribution over the transcripts generated by $\mathcal{P}$ and $\mathcal{V}$. Formally:*

$$(\mathcal{P}, \mathcal{V})(x) = \mathcal{S}(x), \quad \textit{for every } x \in L$$

where $(\mathcal{P}, \mathcal{V})(x)$ is the output distribution of $\mathcal{V}$ after interacting with $\mathcal{P}$ on common input x, and $\mathcal{S}(x)$ is the output distribution of the simulator $\mathcal{S}$ on input x.

When the two distribution are identical, the interactive proof is said to be *perfectly* HVZK, when instead they are *computationally indistinguishable*, the interactive proof is said to be *computationally* HVZK. In the latter case, we can make explicit the statistical distance between the distributions by means of the negligible function (as we did for definition (15)). Further, if the simulator proceeds by first sampling the verifier's messages uniformly at random, the interactive proof is said to be *special* HVZK (SHVZK). In other words, for SHVZK interactive proofs, there must be a PPT simulator that takes as input the public statements and the verifier's internal randomness, and outputs transcripts such that the distribution over generated transcripts is identical to the distribution over the transcripts produced by an honest execution of prover and verifier.

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. In summary, Σ-protocols are commonly expected to have the properties of perfect completeness, special soundness and SHVZK. Moreover, using the Fiat-Shamir heuristic [40], 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 [2, 11].

2.8 Bulletproofs and Compressed Σ-Protocols

In this section, we provide the key concepts of the main argument systems used in this dissertation. In particular, section 2.8.1 describes the *range proof* protocol of Bulletproofs [21], section 2.8.2 outlines the Inner-Product Argument system [17] used as a subroutine of range proofs [21], and section 2.8.3 introduces the Compressed Σ-Protocol theory [10]. The notation for these proof systems is summarized in section 2.1.

2.8.1 Bulletproofs for range proofs

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 Range Proof (RP) protocol, an Inner-Product-Argument (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 $A = \mathit{commit}(\mathbf{a}_L, \mathbf{a}_R)$ and $S = \mathit{commit}(\mathbf{s}_L, \mathbf{s}_R)$. The prover then sends A, S to the verifier.

- Upon receiving y,z challenges from the verifier, 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 $T_i = \mathit{commit}(t_i)$ 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.

- The prover sends $\hat{t}$ and $\mathbf{l} = l(x), \mathbf{r} = r(x)$ evaluations along with other blinding factors τ_x, μ.

- Finally, the verifier checks that the public V is valid, that commitments A,S are correct, and that the inner-product $\langle \mathbf{l}, \mathbf{r} \rangle = \hat{t}$.

The messages exchanged between prover and verifier in the RP protocol are shown in the table 2.2.

Table 2.2: 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

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).

Aggregate Range Proof. An aggregate RP 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 $\mathbf{r} \in \mathbb{Z}_p^{m \cdot n}$.

2.8.2 Inner-product argument system

The *Inner-Product Argument* (IPA) of Bulletproofs is an improved version of the argument system first introduced by Bootle et al. [17]. It is used within the range proof system to efficiently prove that an inner-product between two scalar vectors equals a specific value. Hence, the IPA is an argument of knowledge where a prover $\mathcal{P}$ convinces a verifier $\mathcal{V}$ that he knows the openings for a *binding* vector commitment that satisfy a given *inner-product relation*.

The IPA is a proof system for the following inner-product relation, with $\mathbf{g}, \mathbf{h}, T, c$ public parameters and $\mathbf{a}, \mathbf{b}$ witnesses:

$$\mathcal{R}_{BP} = \{(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, T \in \mathbb{G}, c \in \mathbb{Z}_p;\ \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n) : T = \mathbf{g}^{\mathbf{a}} \mathbf{h}^{\mathbf{b}} \wedge c = \langle \mathbf{a}, \mathbf{b} \rangle\} \quad (2.3)$$

where $\mathbf{g}$ and $\mathbf{h}$ are vectors of (independent) generators, $\mathbf{a}$ and $\mathbf{b}$ are vectors of scalar elements, T is a group element and is a vector commitment of both $\mathbf{a}$ and $\mathbf{b}$, and c is a scalar corresponding to the result of the inner-product between $\mathbf{a}$ and $\mathbf{b}$. The relation (2.3) highlights that the goal by the prover is to convince the verifier that he knows the two vectors $\mathbf{a}$ and $\mathbf{b}$ for the statement $T = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} \wedge c = \langle \mathbf{a}, \mathbf{b} \rangle$. A naive solution that satisfies this relation is for the prover to send the vectors $\mathbf{a}, \mathbf{b}$ to the verifier, but this would cost $2n$ elements in communication.

An optimized version of the IPA reduces the communication to $2log_2(n)$ elements. In this, the value c is given as a part of the vector commitment T by means of an additional group element u. The new IPA results in a proof system for the following inner-product relation:

$$\mathcal{R}_{\mathcal{IPA}} = \{(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n,\ u, T \in \mathbb{G}\ ;\ \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n)\ :\ T = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} \cdot u^{\langle \mathbf{a}, \mathbf{b} \rangle}\} \tag{2.4}$$

where, given the public elements $\mathbf{g}, \mathbf{h}, T$ and u, the prover proves to the verifier that he knows the two vectors $\mathbf{a}, \mathbf{b}$ such that $T = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} \cdot u^{\langle \mathbf{a}, \mathbf{b} \rangle}$. The group element u is appropriately chosen to avoid any known discrete logarithm relation with the vectors $\mathbf{g}, \mathbf{h}$.

The IPA protocol between prover $\mathcal{P}$ and verifier $\mathcal{V}$ is divided into two protocols performed in sequence. The first protocol satisfies the relation 2.3, then the second protocol is used to satisfy the relation 2.4. Next, we provide a brief overview of the two protocols.

First protocol. $\mathcal{P}$ is given as input $(\mathbf{g}, \mathbf{h}, T, c, \mathbf{a}, \mathbf{b})$, whereas $\mathcal{V}$ is given $(\mathbf{g}, \mathbf{h}, T, c)$. The protocol starts with $\mathcal{V}$ computing and sending to $\mathcal{P}$ a uniformly random challenge y. Then, both $\mathcal{P}$ and $\mathcal{V}$ compute the commitment $T' = T \cdot u^{y \cdot c} \in \mathbb{G}$, with $T = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}$ and $c = \langle \mathbf{a}, \mathbf{b} \rangle$. Finally, $\mathcal{P}$ and $\mathcal{V}$ run the *second protocol* on input $(\mathbf{g}, \mathbf{h}, u^y, T'; \mathbf{a}, \mathbf{b})$.

The second protocol follows a logarithmic number of recursive rounds. This is achieved using the *folding strategy*, i.e., a "folding and recursion" approach. With this, the communication complexity of the IPA is logarithmic in the size of the input length and therefore the proof size is also logarithmic. Below we give the intuition of the approach by illustrating a single round of recursion.

Second protocol. $\mathcal{P}$ is given as input $(\mathbf{g}, \mathbf{h}, u, T; \mathbf{a}, \mathbf{b})$, whereas $\mathcal{V}$ is given $(\mathbf{g}, \mathbf{h}, u, T)$, considering the input substitutions $u^y \rightarrow u$ and $T' \rightarrow T$ of the first protocol. The second protocol proceeds as follows:

- $\mathcal{P}$ computes $L = \mathbf{g}_{[n':]}^{\mathbf{a}_{[:n']}} \cdot \mathbf{h}_{[:n']}^{\mathbf{b}_{[n':]}} \cdot u^{\langle \mathbf{a}_{[:n']}, \mathbf{b}_{[n':]} \rangle} \in \mathbb{G}$ and $R = \mathbf{g}_{[n':]}^{\mathbf{a}_{[n':]}} \cdot \mathbf{h}_{[:n']}^{\mathbf{b}_{[:n']}} \cdot$

$u^{\langle \mathbf{a}_{[n':]}, \mathbf{b}_{[:n']} \rangle} \in \mathbb{G}$, where n' is half the length of the respective vectors. Then, $\mathcal{P}$ sends L, R to $\mathcal{V}$.

- $\mathcal{V}$ chooses a random challenge $x \xleftarrow{\$} \mathbb{Z}_p^*$ and sends x to $\mathcal{P}$.
- $\mathcal{P}$ computes $\mathbf{a}' = x \cdot \mathbf{a}_{[:n']} + x^{-1} \cdot \mathbf{a}_{[n':]} \in \mathbb{Z}_p^{n'}$ and $\mathbf{b}' = x \cdot \mathbf{b}_{[n':]} + x^{-1} \cdot \mathbf{b}_{[:n']} \in \mathbb{Z}_p^{n'}$ and sends $\mathbf{a}', \mathbf{b}'$ to $\mathcal{V}$ (here is actually where the folding is applied, in fact the size of the witnesses are halved).
- Finally, the verifier computes $T' = L^{x^2} \cdot T \cdot R^{x^{-2}} \in \mathbb{G}$ and outputs accept if $T' \stackrel{?}{=} \mathbf{g}_{[:n']}^{x^{-1}\mathbf{a}'} \cdot \mathbf{g}_{[n':]}^{x\mathbf{a}'} \cdot \mathbf{h}_{[:n']}^{x\mathbf{b}'} \cdot \mathbf{h}_{[n':]}^{x^{-1}\mathbf{b}'} \cdot u^{\langle \mathbf{a}', \mathbf{b}' \rangle}$.

Improved proof size by recursion. Instead of the prover sending the vectors $\mathbf{a}', \mathbf{b}'$ to the verifier, they can recursively call the (second protocol) IPA on T' and $\mathbf{a}', \mathbf{b}'$, with respect to the new generators that both compute: $\mathbf{g}' = \mathbf{g}_{[:n']}^{x^{-1}} \circ \mathbf{g}_{[n':]}^{x} \in \mathbb{G}^{n'}$ and $\mathbf{h}' = \mathbf{h}_{[:n']}^{x} \circ \mathbf{h}_{[n':]}^{x^{-1}} \in \mathbb{G}^{n'}$. When the base case is reached, the vectors $\mathbf{a}', \mathbf{b}', \mathbf{g}', \mathbf{h}'$ will each have a size of one. The resulting protocol has $\log_2(n)$ total rounds and $2\log_2(n) + 2$ total communication, with respect to the input size n.

The full specification of the first protocol and the (recursive) second protocol appears in appendix A.1 and A.2. The verifier's work is dominated by $4n$ group exponentiations. This can be improved with a single *multi-exponentiation* that delays all exponentiations at the very last round.

Improved verifier through multi-exponentiation. In the last round the verifier checks that $T = g^a h^b u^{a \cdot b}$, with g, h, a, b the shrunken elements from previous rounds. The exponentations can be postponed in the verifier's last round by computing the final g, h as: $g = \prod_{i=1}^{n} g_i^{s_i} \in \mathbb{G}$ and $h = \prod_{i=1}^{n} h_i^{1/s_i} \in \mathbb{G}$, where $\mathbf{s} = (s_1, \ldots, s_n) \in \mathbb{Z}_p^n$. The scalars $s_1, \ldots, s_n \in \mathbb{Z}_p$ only depends on the $\log_2(n)$ verifier's challenges and are calculated as follows:

$$\forall i \in [1, n] : s_i = \prod_{j=1}^{log_2(n)} x_j^{b(i,j)},$$

where

$$b(i,j) = \begin{cases} 1 & \text{if bit } j-1 \text{ of } i-1 \text{ is } 1 \\ -1 & \text{otherwise} \end{cases}$$

.

2.8.3 Compressed Σ-protocols

The *Compressed Σ-protocols* [10] arise from the intuition of reconciling the Bulletproofs (BPs) compression mechanism with the Σ-protocol theory. This

enables the design of ZK-proofs within an established theory and under DLOG assumptions, with the same communication complexity of BPs. The theory of compressed Σ-protocols starts from the basic pivot: let L an *arbitrary* and *public linear form*, given a Pedersen vector commitment T to a vector $\mathbf{x}$ and the public scalar $y = L(\mathbf{x})$, the prover $\mathcal{P}$ provides a proof of knowledge to a verifier $\mathcal{V}$ that he knows $\mathbf{x}$ whose commitment satisfies L. Hence, a simple (but with inefficient communication) Σ-protocol is provided for that pivot, and then by applying an appropriate BPs *compression* a protocol with logarithmic communication is obtained. Moreover, the theory relies on known analyses for knowledge extractors and the resulting protocol is an honest-verifier zero-knowledge proof of knowledge.

The Σ-protocol, called Π_0, for the basic pivot is a 3-move protocol between $\mathcal{P}$ and $\mathcal{V}$ that proceeds as described below.

- In the first move, $\mathcal{P}$ computes a commitment A to a random vector $\mathbf{r}$. Then, $\mathcal{P}$ sends A and the values $y = L(\mathbf{x})$ and $y' = L(\mathbf{r})$ to $\mathcal{V}$, where $\mathbf{x}$ is the secret vector of which $\mathcal{P}$ wants to prove knowledge.
- In the second move, $\mathcal{V}$ sends a random challenge c.
- In the third move, $\mathcal{P}$ opens the commitment AT^c to a vector $\mathbf{z}$ and finally $\mathcal{V}$ checks that $L(\mathbf{z}) = cy + y'$.

The protocol is HVZK and is dominated by a linear communication complexity. Then the protocol communcation can be reduced to logarithmic using BPs compression technique, obtaining an *argument of knowledge* (because soundness degrades from unconditional to computational).

Concretely, let $\mathbb{G}$ an abelian group of prime order p, $T = \mathbf{g}^{\mathbf{x}}h^{\gamma} \in \mathbb{G}$ a Pedersen vector commitment of the secret vector $\mathbf{x} \in \mathbb{Z}_p^n$ of length n with $\gamma \xleftarrow{\$} \mathbb{Z}_p$ the secret randomness for the blinding factor, and L a linear form such that $L : \mathbb{Z}_p^n \to \mathbb{Z}_p$. Then, Π_0 is a proof of knowledge for the relation:

$$\mathcal{R}_{\Sigma} = \{(\mathbf{g} \in \mathbb{G}^n, h, T \in \mathbb{G}, y \in \mathbb{Z}_p;\ \mathbf{x} \in \mathbb{Z}_p^n, \gamma \in \mathbb{Z}_p) : T = \mathbf{g}^{\mathbf{x}}h^{\gamma} \wedge y = L(\mathbf{x})\} \tag{2.5}$$

To define the compressed version of Π_0 the general idea is that instead of the prover sending the vector $\mathbf{z}$ in the final move, the prover convinces the verifier that he knows $\mathbf{z}$. This is done by first incorporating the linear form L into the commitment and replacing the final move with two Σ-protocols $\Pi_{1,2}$ satisfying a new relation (see relation $\mathcal{R}_2$ in [10]). Essentially, protocol Π_2 is actually where the BPs *folding strategy* is applied. Finally, the compressed Σ-protocol Π_c for relation (2.5) is obtained by composition, i.e., $\Pi_c := \Pi_0 \diamond \Pi_1 \diamond \Pi_2$ from definition (22). The resulting protocol Π_c is $(2\mu + 3)$-move,

perfectly complete, **k**-special-sound and SHVZK, where $\mu = \log_2(n+1) - 1$ with n the size of the secret vector.

PART III

ZEROMT AND COMPRESSED Σ-IPA

CHAPTER 3

ZEROMT: ZK-PROOFS BASED MULTI-TRANSFER SCHEME FOR ACCOUNT MODEL BLOCKCHAIN

3.1 Introduction

Bitcoin [54] 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 [24] 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 [28, 41, 59, 5]. 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 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 [56, 62, 47, 3]. 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 [38, 20]. 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 [21, 46]. 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 [45, 60].

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) [14]. 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 [13], 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 [60]. 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 [62]. Groth16 [44] 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. [17] and its improvement proposed in Bulletproofs [21]. 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 this dissertation 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. [18] 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* [71, 33, 23, 50]. However, such schemes may incur costly pairing operations for the verifier [65].

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 [47], given the possibility of deriving other information such as the number of coins exchanged. Moreover, UTXO-based protocols such as Ze-

rocash [62] and Monero [3] suffer from continuous growth in the size of the UTXO set [38, 34], 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 [38] 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 [20, 34], and the hybrid balance models can lead to increased gas costs for storage when integrated into practice in smart contract platforms [20]. 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 [20], 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 [20]. 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.

3.1.1 New directions of CT protocols: multi-transfer transactions

A new direction of CT protocols 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 [38] 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 [20, 60, 30]. 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 [60]. 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.

- 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.

3.1.2 Our contribution

We propose *ZeroMT*, a new *Zero-knowledge Multi-Transfer* 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.

In summary, for the benefits of the account model discussed, our starting point is Zether [20], 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 the fundamental security properties (e.g., overdraft-safety and privacy) still 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 [8]. Finally, we provide concrete evaluations in terms of ZK-proof performance and transaction costs.

3.1.3 Related works

In this section we provide the review and comparison of relevant literature on CT protocols, with particular emphasis on the concept of multi-transfer.

MimbleWimble [56] 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 [62] 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 [47] 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 [3] is a private cryptocurrency based on the RingCT protocol [55], 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 [55] 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 [38] 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 ZK-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 [20] 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 support aggregation of proofs in transactions and batch verification for multiple transfers. An anonymous version of Zether is given in the work [34], 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 [60] 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 [45] 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 3.1. Our solution benefits from trustless and multi-transfer support, but does not guarantee the unlinkability of sender/receiver address.

Table 3.1: 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

3.1.4 Motivation scenario

In this section, we motivate our *ZeroMT* proposal by taking a practical scenario, i.e., the smart lock case study [26, 25]. In this case study there are untrustworthy small devices and servers performing off-chain interactions, before some data is stored in the blockchain. In particular, a smart lock is installed on different electric bicycles. A Client can start the ebike rental service by requesting the opening of the smart lock and its closing at the end of the service. The Client pays the rental time to the Service Provider which shares the profit and fees with the Insurance. In the following, we generalize the smart lock scenario through a Client/Server paradigm, in order

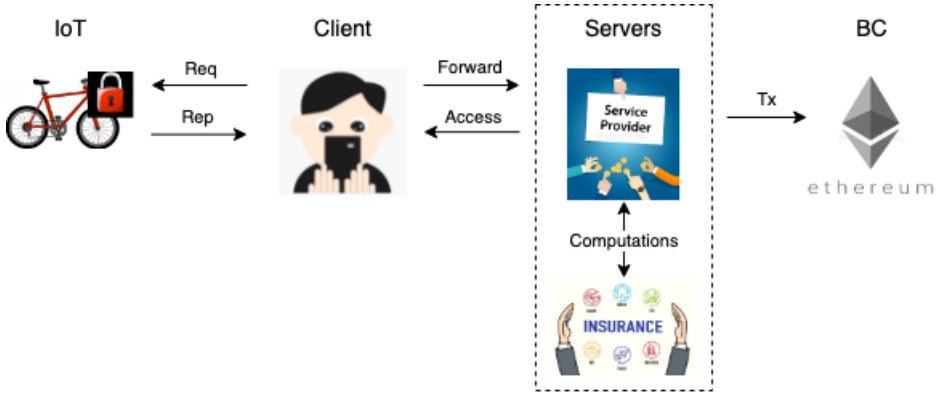


Figure 3.1: General smart lock

to highlight the need for multi-transfer transactions in privacy settings. The Client starts a new session making a request to the IoT device, then forwards the received data to the Servers. The IoT device generates session data for the Client. The Servers make their own computations on Client's session data, then assemble a transaction with the data to make the session publicly verifiable. Finally, Servers generate an access key to enable the IoT device. The generalized scenario is depicted in Figure 3.1. The blockchain is used for the integrity of the Client's session data and also for balance transfers between parties. A first attempt for the Client's privacy is not to involve him in payments via the blockchain, nor to store his identity in the blockchain. However, this limit the possibility of having automatic payments, as well as mechanisms to discourage dishonest behaviours through automatic penalties, e.g., when a dispute occurs. There are also privacy concerns for Servers: balances and transaction amounts are all in clear in the blockchain side and everyone can access them. In business contexts, it is often preferred not to disclose balances and the amounts exchanged in transactions to competitors. Moreover, in off-chain scenarios there are often interactions between multiple entities with different roles, and thus off-chain sessions end with transfers involving multiple payees. Traditional payment models only allow separate transfers for each recipient, requiring inefficient work from the main chain to execute and validate individual transactions. This can also have an impact on privacy, giving an attacker the possibility to analyze a wider set of transactions coming from the same source.

ZeroMT aims at solving these issues through a transaction scheme that has advantage in two aspects: i) for *privacy*, it allows *confidential transactions* without revealing sensitive information to the blockchain network and *zero-knowledge balances* with homomorphic updates by balance administrators (e.g., smart contracts); ii) for *scalability*, it leverages on *aggregation* techniques to allow *multi-transfer transactions* with multiple

payees and to reduce the number of transactions validated in the main chain.

***ZeroMT* simplistic payment mechanism**. The more complex case of a *multi-transfer* is when we want to assemble a single transaction for transfers between a group of senders and a group of recipients. This can be seen as a *many-to-many* transfer of funds relation. We reduce the problem with the simpler case where one payer creates a transaction for multiple recipients, say a *one-to-many* transfer relation. Moreover, we claim that this payment transaction must be authorised by the payer, the one to whom the balance will be charged (the Client in our smart lock scenario). Suppose we have an ebike user who has finished the rental service. After a sequence of context-dependent interactions with the Service Provider, the user's personal device has collected the following information, before assembling the *multi-transfer transaction*: (i) a list of the public addresses of the recipients, in our case the public key of the Service Provider (SP) and that of the Insurance (INS), respectively denoted with $\bar{y}_{sp}$ and $\bar{y}_{ins}$; (ii) a list of transfer amounts to be credited to recipients' balances, in our case those of the SP and INS, denoted with a_{sp} and a_{ins}.

To make the multi-transfer confidential, we need to hide the amounts in the transaction and the balances of all off-chain users. We can employ a *homomorphic encryption* method to keep amounts hidden and balances encrypted within the balance admin. This also allows to perform homomorphic updates of those balances when the transaction is executed. In order to transfer the right amount to the right recipient, the Client encrypts each amount under both its public key and the recipient's public key. Then the Client generates two lists of ciphertexts, $\mathbf{C} = (C_1, ..., C_n)$ and $\bar{\mathbf{C}} = (\bar{C}_1, ..., \bar{C}_n)$, where $Enc_y(a_i) = C_i$ is the ciphertext under the Client's public key y of each i-th amount in the list of transfer amounts, and $Enc_{\bar{y}_i}(a_i) = \bar{C}_i$ is the ciphertext under the i-th public key in the list of recipient addresses of the i-th amount in the list of transfer amounts. Together with these ciphertexts, the Client also provides with the multi-transfer transaction a ZK-proof, denoted with π, to prove statements behind the *zero-knowledge multi-transfer relation* (we will define this zk-relation in section 3.2.2).

Upon receiving the multi-transfer transaction sent by the Client, the balance admin runs a verification procedure to validate the ZK-proof π. If the proof π is successfully verified (the procedure outputs 1), all the requested transfers are executed and the encrypted balances updated. The general flow of the multi-transfer transaction is depicted in Figure 3.2. We remark that the ZK-proof π is an *aggregate* proof, i.e., it allows batch verification of multiple transfers through a single proof. In the next sections, we will formally define the multi-transfer transaction scheme and show benefits of the aggregation.

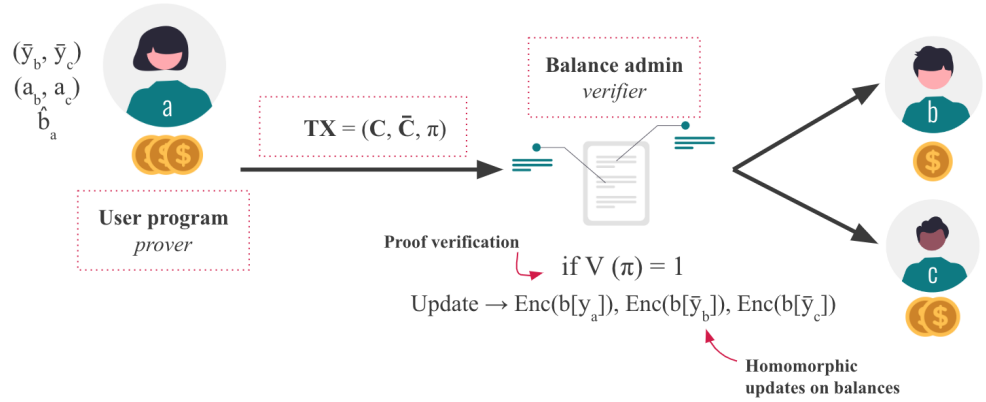


Figure 3.2: Multi-transfer transaction flow. The payer (a) runs a user program (acting as a prover) to create a multi-transfer transaction (TX). This transaction allows the payer to transfer money privately to payees (b) and (c). The ZK-proof π is also generated to prove the statements behind the multi-transfer 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.

3.2 Multi-Transfer Scheme

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 3.2.4) 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.

3.2.1 Background on Zether

In this section, we summarize the basics of Zether [20]. 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 tx_{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.

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 3.4). This results in cost savings for transactions and provides atomicity to the batch of aggregate transfers.

3.2.2 ZK-relation and multi-transfer scheme

The intuition behind the multi-transfer was presented in section 3.1.4. Instead, in this section we formalize the concept of multi-transfer and its cryptographic scheme, namely the **multi-transfer scheme**.

In the multi-transfer settings, 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 $(C_1 ... C_n, D)$ and $(\bar{C}_1 ... \bar{C}_n, D)$ encrypts an equal amount a_i at the same index $i \in [1, n]$ and is well-formed;
(v) that each individual amount $(a_1, ..., a_n)$ and the balance $\hat{b}$, after deducting these amounts, are not negative.

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

In section 3.2.4, we treat zk-relations $\mathcal{R}$ formally as a set of instances, witnesses and algebraic statements as in definition (16), to specify how our ZK-proof is constructed.

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 parame-

ters 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 i-th 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.

Protocol 2 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. $\pi = \mathbf{Prove}_{\text{nizk}}(\mathcal{R}_{multi}[C_L, C_R, y, \bar{\mathbf{y}}, \mathbf{C}, \bar{\mathbf{C}}, g, D; w])$

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 $\mathcal{R}_{multi}$, 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`.

3.2.3 Security analysis

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 $\mathcal{O}_{\mathcal{SC}}$ 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 $\mathcal{O}_{\mathcal{SC}}$ 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 $\mathcal{O}_{\mathcal{SC}}$; (iii) it can submit direct transactions to $\mathcal{O}_{\mathcal{SC}}$; (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 $\mathcal{O}_{\mathcal{SC}}$ 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 $\mathbf{y_i} = (y_1, ..., y_n)$ 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 $\mathcal{O}_{\mathcal{SC}}$ to initiate withdrawals from these accounts,

thereby obtaining quantities $b_1, ..., b_n$. 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 (19) of soundness in the following:

Definition 24 (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 (23) in the following:

Definition 25 (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 section 3.2.5 (Corollaries 1 and 2).

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. Now, 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 $\mathrm{tx}_{\mathrm{multi}}$. 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 [57]. 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.

3.2.4 Zero-knowledge multi-transfer proof system

In this section, we design the zero-knowledge interactive proof system for our Multi-Transfer scheme. The system leverages the *aggregate Range Proof* (RP) from Bulletproofs theory [21] in combination with several Σ-Protocols, exploiting the adaptations of Σ-Bullets [20]. 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 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 dedicated to the following objectives: we recall some underpinnings of non-aggregate RP, such as Bulletproofs constraints and the derivation of complex formulas; we show how we use the aggregate RP and the Σ-Bullets adaptations in our ZK-proof; we design the Σ-protocols for the rest of our ZK-relations; we prove two Theorems that establish the security of our

interactive ZK-proof.

Non-aggregate RP and Bulletproofs constraints. A non-aggregate RP is a proof system for proving that a value $v \in \mathbb{Z}_p$ is in range, e.g., $v \in [0, 2^{n-1}]$ with n the bit-range domain. The non-aggregate RP of Bulletproofs uses *Pedersen commitments* as building blocks and formulates the statement $V = g^v h^\gamma \wedge v \in [0, 2^{n-1}]$ using generators $g, h \in \mathbb{G}$ and a randomness $\gamma \in \mathbb{Z}_p$. In order to prove that v is in range, the Bulletproofs strategy is to first decompose v into bits and create a vector $\mathbf{a}_L \in \{0,1\}^n$ such that $\langle \mathbf{a}_L, \mathbf{2}^n \rangle = v$. We provide a simple toy example for better clarification. Considering $v = 4$ and $n = 5$, we have $\mathbf{a}_L = (0,0,1,0,0)$ and $\mathbf{2}^n = (2^0, 2^1, 2^2, 2^3, 2^4)$, and it is straightforward to see that

$$v = \sum \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ & & \times & & \\ 2^0 & 2^1 & 2^2 & 2^3 & 2^4 \end{pmatrix} = 4.$$

Hence, $\langle \mathbf{a}_L, \mathbf{2}^n \rangle = v$ can be considered a proof system constraint to prove that $\mathbf{a}_L$ is composed of the bits of v, but it is not sufficient. Indeed, Bulletproofs adds two more constraints to prove that each bit of $\mathbf{a}_L$ is in $\{0,1\}$. In particular, an additional vector $\mathbf{a}_R$ is used such that

$$\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^n \text{ and } \mathbf{a}_L \circ \mathbf{a}_R = \mathbf{0}^n.$$

As an example, consider $\mathbf{a}_L = (0,1,1,0,0,1,1,1,...,1)$, then $\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^n = (-1,0,0,-1,-1,0,0,0,...,0)$ if and only if

$$\mathbf{a}_L \circ \mathbf{a}_R = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & ... & 1 \\ & & & & \times & & & & & \\ -1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & ... & 0 \end{pmatrix} = \mathbf{0}^n.$$

Now, the Bulletproofs strategy is to combine all the above constraints into a single *inner-product constraint* in order to use the *inner-product argument* in the final stage of the RP. To do this, the constraints are first combined through a random linear combination chosen by the verifier. Hence, given some random challenge $y \xleftarrow{\$} \mathbb{Z}_p$, the constraints are rewritten as follows:

$$\langle \mathbf{a}_L, \mathbf{2}^n \rangle = v \text{ and } \langle \mathbf{a}_L - \mathbf{1}^n - \mathbf{a}_R, \mathbf{y}^n \rangle = 0 \text{ and } \langle \mathbf{a}_L, \mathbf{a}_R \circ \mathbf{y}^n \rangle = 0.$$

Using an additional random challenge $z \xleftarrow{\$} \mathbb{Z}_p$, the constraints are once again combined into one equality:

$$z^2 v = z^2 \langle \mathbf{a}_L, \mathbf{2}^n \rangle + z \langle \mathbf{a}_L - \mathbf{1}^n - \mathbf{a}_R, \mathbf{y}^n \rangle + \langle \mathbf{a}_L, \mathbf{a}_R \circ \mathbf{y}^n \rangle$$

Finally, this equality can be rewritten as a single inner-product:

$$z^2 v + \psi(y, z) = \langle \mathbf{a}_L - z\mathbf{1}^n, \mathbf{y}^n \circ (\mathbf{a}_R + z\mathbf{1}^n) + z^2 \mathbf{2}^n \rangle$$

where ψ is a quantity that the verifier can easily calculate as:

$$\psi(y,z) = (z-z^2)\langle \mathbf{1}^n, \mathbf{y}^n\rangle - z^3\langle \mathbf{1}^n, \mathbf{2}^n\rangle \in \mathbb{Z}_p.$$

These are the Bulletproofs constraints and the derivation of formulas for the non-aggregate RP, which are also inherited in the aggregate RP version used in our multi-transfer proof system. In the next subsection, we introduce the aggregate RP, highlighting the adjustments made to the formulas as discussed in section (2.8.1). For completeness, we provide the steps below necessary for deriving the single inner-product equality.

$$z^2 v = z^2\langle \mathbf{a}_L, \mathbf{2}^n\rangle + z\langle \mathbf{a}_L - \mathbf{1}^n - \mathbf{a}_R, \mathbf{y}^n\rangle + \langle \mathbf{a}_L, \mathbf{a}_R \circ \mathbf{y}^n\rangle$$

$$z^2 v = z^2\langle \mathbf{a}_L, \mathbf{2}^n\rangle + z\langle \mathbf{a}_L, \mathbf{y}^n\rangle - z\langle \mathbf{a}_R, \mathbf{y}^n\rangle - z\langle \mathbf{1}^n, \mathbf{y}^n\rangle + \langle \mathbf{a}_L, \mathbf{a}_R \circ \mathbf{y}^n\rangle$$

$$z^2 v + z\langle \mathbf{1}^n, \mathbf{y}^n\rangle = z^2\langle \mathbf{a}_L, \mathbf{2}^n\rangle + z\langle \mathbf{a}_L, \mathbf{y}^n\rangle - z\langle \mathbf{1}^n, \mathbf{a}_R \circ \mathbf{y}^n\rangle + \langle \mathbf{a}_L, \mathbf{a}_R \circ \mathbf{y}^n\rangle$$

$$z^2 v + z\langle \mathbf{1}^n, \mathbf{y}^n\rangle = \langle \mathbf{a}_L, z^2\mathbf{2}^n\rangle + \langle \mathbf{a}_L, z\mathbf{y}^n\rangle + \langle -z\mathbf{1}^n, \mathbf{a}_R \circ \mathbf{y}^n\rangle + \langle \mathbf{a}_L, \mathbf{a}_R \circ \mathbf{y}^n\rangle$$

$$z^2 v + z\langle \mathbf{1}^n, \mathbf{y}^n\rangle = \langle \mathbf{a}_L, z^2\mathbf{2}^n + z\mathbf{y}^n + \mathbf{a}_R \circ \mathbf{y}^n\rangle + \langle -z\mathbf{1}^n, \mathbf{a}_R \circ \mathbf{y}^n\rangle$$

By adding the quantity $\langle -z\mathbf{1}^n, z^2\mathbf{2}^n + z\mathbf{y}^n\rangle$ to both sides we obtain:

$$z^2 v + z\langle \mathbf{1}^n, \mathbf{y}^n\rangle - \langle z\mathbf{1}^n, z^2\mathbf{2}^n + z\mathbf{y}^n\rangle$$

$$= \langle \mathbf{a}_L, z^2\mathbf{2}^n + z\mathbf{y}^n + \mathbf{a}_R \circ \mathbf{y}^n\rangle + \langle -z\mathbf{1}^n, z^2\mathbf{2}^n + z\mathbf{y}^n + \mathbf{a}_R \circ \mathbf{y}^n\rangle$$

$$z^2 v + z\langle \mathbf{1}^n, \mathbf{y}^n\rangle - \langle \mathbf{1}^n, z^3\mathbf{2}^n + z^2\mathbf{y}^n\rangle = \langle \mathbf{a}_L - z\mathbf{1}^n, z^2\mathbf{2}^n + z\mathbf{y}^n + \mathbf{a}_R \circ \mathbf{y}^n\rangle$$

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

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

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

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

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

$$z^2 v + \psi(y,z) = \langle \mathbf{a}_L - z\mathbf{1}^n, \mathbf{y}^n \circ (\mathbf{a}_R + z\mathbf{1}^n) + z^2\mathbf{2}^n\rangle$$

Hence, we obtained the single inner-product equality.

It remains to demystify how the vector polynomials $l(X), r(X) \in \mathbb{Z}_p^n$ used to derive the polynomial $t(X)$ are constructed. Given the single inner-product equality, the prover can compute the two un-blinded vectors $\mathbf{l} = \mathbf{a}_L - z\mathbf{1}^n$ and $\mathbf{r} = \mathbf{y}^n \circ (\mathbf{a}_R + z\mathbf{1}^n) + z^2\mathbf{2}^n$, considering that $z^2 v + \psi(y,z) = \langle \mathbf{l}, \mathbf{r}\rangle$. However, sending $\mathbf{l}$ and $\mathbf{r}$ to the verifier reveals information about v. The prover instead defines the vector polynomials $l(X), r(X)$ as follows:

$$l(X) := l_l + l_r \cdot X,\ \ r(X) := r_l + r_r \cdot X \in \mathbb{Z}_p^n[X]$$

where

$$\mathbf{l}_l = \mathbf{a}_L - z\mathbf{1}^n,$$
$$\mathbf{l}_r = \mathbf{s}_L,$$
$$\mathbf{r}_l = \mathbf{y}^n \circ (\mathbf{a}_R + z\mathbf{1}^n) + z^2\mathbf{2}^n,$$
$$\mathbf{r}_r = \mathbf{y}^n \circ \mathbf{s}_R,$$

and $\mathbf{s}_L, \mathbf{s}_R \xleftarrow{\$} \mathbb{Z}_p^n$ blinding vectors. Note that $\langle \mathbf{l}_l, \mathbf{r}_l \rangle = z^2 v + \psi(y, z)$.

Finally, the polynomial $t(X)$ is defined as the inner product between the vector polynomials $l(X), r(X)$ as follows:

$$t(X) = \langle l(X), r(X) \rangle = \sum_{i=0}^{d} t_i X^i \in \mathbb{Z}_p[X],$$

where $d = 2$.

At this point, for the prover to convince the verifier that the single equality of the inner product holds, it is enough to prove that the constant term t_0 of $t(X)$ is equal to $z^2 v + \psi(y, z)$ and that $t(X)$ is the correct polynomial. Proving that $t(X)$ is correct means proving that $l(X)$ and $r(X)$ are correctly formed and that $t(x) = \langle l(x), r(x) \rangle$.

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 $a_1, \ldots, a_j$ 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 $a_1, \ldots, a_j$, 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 2.8.1; 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, we adopt a similar strategy of Σ-Bullets. 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:

$$\begin{aligned} \mathcal{R}_{AggRP} : \{(g, h \in \mathbb{G},\ y, z, \psi(y,z), \hat{t} \in \mathbb{Z}_p;\ \hat{b} \in \mathbb{Z}_p, \mathbf{a} \in \mathbb{Z}_p^{m-1}, \tau_x \in \mathbb{Z}_p) : \\ g^{\hat{t} - \hat{b} \cdot z^2 - \sum_{i=1}^{m-1} a_i \cdot z^{i+2} - \psi(y,z)} h^{\tau_x} = T_{1,2}\} \end{aligned} \tag{3.1}$$

Where y and z are the challenges of the verifier, 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 (3.1); 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 3 *Aggregate RP* $\wedge$ Σ_{AggRP} interactive protocol

1: $\mathcal{P}$ computes
2: $\quad \mathbf{a}_L \in \{0,1\}^{m \cdot n}$ s.t.
3: $\quad\quad \langle \mathbf{2}^n, \mathbf{a}_{L[:n]} \rangle = \hat{b}$ 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 $\mathcal{P}$
$\mathcal{P} \rightarrow \mathcal{V}$: A, S
$\mathcal{V} \rightarrow \mathcal{P}$: $y, z \xleftarrow{\$} \mathbb{Z}_p^*$
$\mathcal{P}$ computes
$l(X) := l_l + l_r \cdot X \in \mathbb{Z}_p^{m\cdot n}[X]$
$r(X) := r_l + r_r \cdot X \in \mathbb{Z}_p^{m\cdot n}[X]$
$l_l \leftarrow \mathbf{a}_L - z\mathbf{1}^{n\cdot m}$
$l_r \leftarrow \mathbf{s}_L$
$r_l \leftarrow \mathbf{y}^{n\cdot m} \circ (\mathbf{a}_R + z\mathbf{1}^{n\cdot m}) + \sum_{k=1}^{m} z^{k+1} \cdot (\mathbf{0}^{n\cdot(k-1)} || \mathbf{2}^n || \mathbf{0}^{n\cdot(m-k)})$
$r_r \leftarrow \mathbf{y}^{n\cdot m} \circ \mathbf{s}_R$
$t(X) := \sum_{i=0}^{d} t_i X^i \in \mathbb{Z}_p[X]$, 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 $\mathcal{P}$
$\mathcal{P} \rightarrow \mathcal{V} : T_1, T_2$
$\mathcal{V} \rightarrow \mathcal{P} : x \xleftarrow{\$} \mathbb{Z}_p^*$
$\mathcal{P}$ 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 $\mathcal{P}$
$\mathcal{P} \rightarrow \mathcal{V} : \hat{t}, \mu, A_t$
$\mathcal{V} \rightarrow \mathcal{P}$: $c \xleftarrow{\$} \mathbb{Z}_p^*$
$\mathcal{P}$ 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 $\mathcal{P}$
$\mathcal{P} \rightarrow \mathcal{V} : s_{ba}, s_{\tau}$
$\mathcal{V}$ 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 $\mathcal{V}$

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

(described in section 2.8.2). As specified in [21], 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

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

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. The full specification of our inner-product argument with multi-exponentiation appears in appendix A.3.

Theorem 1. Aggregate RP $\wedge$ Σ_{AggRP} is a public-coin (perfectly) special honest-verifier argument of knowledge of the relation $\mathcal{R}_{AggRP}$ *(3.1).*

Proof. The *Aggregate RP* has perfect completeness, special soundness and perfect special honest verifier zero-knowledge from Theorem 2 of [21]. We now show that our Σ_{AggRP} is an argument of knowledge of the relation (3.1).

Perfect completeness follows from equality 46 of Protocol 3. 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:

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

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

$$= A_t \cdot (g^{(\hat{t}-\psi(y,z))-\hat{b}z^2-\sum_{i=1}^{n} a_i z^{i+2}} \cdot h^{\tau_x})^c$$

$$= A_t \cdot T_{1,2}^c$$

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:

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

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

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:

$$\frac{g^{c\cdot t - s_{ba}} h^{s_\tau}}{g^{c'\cdot t - s'_{ba}} h^{s'_\tau}} = g^{t(c-c') - (s_{ba} - s'_{ba})} h^{s_\tau - s'_\tau}$$

$$= g^{t(c-c') - (\hat{b}\cdot z^2 + \sum_{i=1}^n a_i \cdot z^{2+i})(c-c')} h^{\tau_x (c-c')}$$

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 $\mathcal{S}$ 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 $\mathcal{S}$, and so $A_t \in_R \mathbb{G}$ is a random group element, this means they are identically distributed as in the honest protocol. □

Σ-Protocol for multiple transfers. We now outline the remaining part of our interactive ZK-proof for the multi-transfer scheme. In particular, we

design a Σ-Protocol satisfying the zero-knowledge relation:

$$\mathcal{R}_{MultiTrans} : \{(C_L, C_R, D, g, y \in \mathbb{G},\ \mathbf{C}, \bar{\mathbf{C}}, \bar{\mathbf{y}} \in \mathbb{G}^{m-1};\ \mathbf{a} \in \mathbb{Z}_p^{m-1},\ \hat{b}, sk, r \in \mathbb{Z}_p) : \\ (C_i = g^{a_i} y^r \wedge \bar{C}_i = g^{a_i} \bar{y}_i^r \wedge D = g^r)_{i=1}^{m-1} \wedge \\ C_L \cdot (\prod_{i=1}^{m-1} C_i)^{-1} = g^{\hat{b}} \cdot (C_R \cdot (\prod_{i=1}^{m-1} D)^{-1})^{sk} \wedge \\ y = g^{sk}\} \tag{3.2}$$

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 i-th 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 $\mathcal{R}_{MultiTrans}$ (3.2), shown in the following protocol.

Protocol 4 $\Sigma_{MultiTrans}$ interactive protocol

$\mathcal{P}$ computes:
  $k_{sk}, k_r, k_{ba} \xleftarrow{\$} \mathbb{Z}_p$
  $A_y = g^{k_{sk}} \in \mathbb{G}$
  $A_D = g^{k_r} \in \mathbb{G}$
  $A_{ba} = ((C_R / \prod_{i=1}^{m-1} D)^{z^2} \cdot \prod_{i=1}^{m-1} D^{z^{i+2}})^{k_{sk}} \cdot g^{k_{ba}} \in \mathbb{G}$
  $A_{\bar{y}} = \prod_{i=1}^{m-1} (y \cdot \bar{y}_i^{-1})^{k_r} \in \mathbb{G}$
end $\mathcal{P}$
$\mathcal{P} \rightarrow \mathcal{V} : A_y, A_D, A_{ba}, A_{\bar{y}}$
$\mathcal{V} \rightarrow \mathcal{P} : c \xleftarrow{\$} \mathbb{Z}_p^*$
$\mathcal{P}$ computes:
  $s_{sk} = c \cdot sk + k_{sk} \in \mathbb{Z}_p$
  $s_r = c \cdot r + k_r \in \mathbb{Z}_p$
  $s_{ba} = c \cdot (\hat{b} z^2 + \sum_{i=1}^{m-1} (a_i z^{i+2})) + k_{ba} \in \mathbb{Z}_p$
end $\mathcal{P}$
$\mathcal{P} \rightarrow \mathcal{V} : s_{ba}, s_{sk}, s_r$
$\mathcal{V}$ requires:
  $g^{s_{sk}} \stackrel{?}{=} A_y y^c$
  $g^{s_r} \stackrel{?}{=} A_D D^c$
  $g^{s_{ba}} ((\frac{C_R}{\prod_{i=1}^{m-1} D})^{z^2} \cdot \prod_{i=1}^{m-1} D^{z^{2+i}})^{s_{sk}} \stackrel{?}{=} A_{ba} ((\frac{C_L}{\prod_{i=1}^{m-1} C_i})^{z^2} \cdot \prod_{i=1}^{m-1} C_i^{z^{2+i}})^c$
  $\prod_{i=1}^{m-1} (y \cdot \bar{y}_i^{-1})^{s_r} \stackrel{?}{=} A_{\bar{y}} \cdot (\prod_{i=1}^{m-1} C_i / \bar{C}_i)^c$

21: **end** $\mathcal{V}$

Theorem 2. $\Sigma_{MultiTrans}$ is a public-coin (perfectly) special honest-verifier argument of knowledge of the relation $\mathcal{R}_{MultiTrans}$ *(3.2).*

Proof. 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 4 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:

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

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

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

$$= A_{ba} \cdot \left[\left(\frac{C_L}{\prod_{i=1}^n C_i} \right)^{z^2} \cdot \prod_{i=1}^n C_i^{z^{i+2}} \right]^c$$

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:

$$\prod_{i=1}^n (y \cdot \bar{y}_i^{-1})^{s_r} =$$

$$= \prod_{i=1}^n (y \cdot \bar{y}_i^{-1})^{k_r} \cdot \prod_{i=1}^n (y \cdot \bar{y}_i^{-1})^{cr} = A_{\bar{y}} \cdot \prod_{i=1}^n \left(\frac{g^{a_i} y^r}{g^{a_i} \bar{y}_i^r} \right)^c$$

$$= A_{\bar{y}} \cdot \prod_{i=1}^{n} \left(\frac{C_i}{\bar{C}_i} \right)^c$$

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

We now show that Protocol 4 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 4 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. □

3.2.5 Security reductions

In this section, we provide a more detailed security analysis of the multi-transfer scheme Π_{MT} through suitable *security reductions*. Thus, we continue with further reductions, until we define the advantage of any adversary on the underlying cryptographic assumptions (e.g., DLOG). We first show that any efficient adversary that can succeed in the `overdraft-safety` experiment presented in 3.2.3, can be turned into an algorithm that efficiently breaks the hardness of DLOG assumption. Here, we denote the overdraft-safety experiment as OVS. In particular, suppose $\mathcal{A}$ is an efficient adversary attacking

Π_{MT} as defined in OVS experiment and with advantage $Adv^{OVS}_{\mathcal{A},\Pi_{MT}}$. Then, there exists an efficient DLOG adversary $\mathcal{B}$ with advantage $Adv^{DLOG}_{\mathcal{B},\mathbb{G}}$, such that

$$Adv^{OVS}_{\mathcal{A},\Pi_{MT}} \leqslant Adv^{DLOG}_{\mathcal{B},\mathbb{G}} \tag{3.3}$$

The security reductions proceed as follows:

- In the OVS experiment, the adversary $\mathcal{A}$ creates $\mathrm{tx}_{\mathrm{multi}}$ and subsequently withdraws from corrupted accounts. By the *nonce* mechanism, this withdraw query will not take effect in the same epoch of the transfer query. Hence, we reduce $\mathcal{A}$ into an adversary $\mathcal{A}'$ trying to break the *soundness* of the NIZK ZK-proof.

- By the assumption that NIZK ZK-proof inherits the soundness of the interactive version, we modify the previous experiment by taking an adversary $\mathcal{A}''$ attacking the soundness of the *interactive* ZK-proof.

- We now introduce a (not elementary) wrapper adversary $\mathcal{B}$ of $\mathcal{A}''$, that outputs witnesses $\mathbf{w}$ satisfying the multi-transfer relation (3.2), thus breaking the soundness of the ZK-proof.

- The adversary $\mathcal{B}$ plays the role of an extractor, denoted with $\mathcal{E}$, which gets from $\mathcal{A}''$ accepting transcripts, by rewinding, and later extracts a valid witness. This experiment is summarized below, which shows that from the existence of the extractor $\mathcal{E}$, the advantage of every adversary $\mathcal{A}''$ in the experiment is negligible.

$$Pr\begin{bmatrix} \sigma \leftarrow \mathtt{Setup}(1^\lambda),\ (x, tr, tr') \leftarrow \mathcal{A}''(\sigma), \\ w \leftarrow \mathcal{E}(\sigma, tr, tr')\ : \\ \langle \mathcal{A}''(\sigma, x, w), \mathcal{V}(\sigma, x)\rangle = 1, (x, w) \notin \mathcal{R}. \end{bmatrix} \leqslant \mathtt{negl}(\lambda)$$

  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 `negl` is a negligible function in the security parameter λ.

We have shown that the overdraft-safety security can be reduced to the soundness of our ZK-proof. Then, by the existence of an extractor $\mathcal{B}$ interacting with $\mathcal{A}''$ to obtain accepting transcripts, we can break the soundness of the ZK-proof by extracting the witness. We already proved the existence of several extractors for our ZK-proof, as the following corollary states.

Corollary 1. *In the light of the Theorems 1 and 2 we can say that from the soundness of our ZK-proof in interactive form, the Multi-Transfer scheme* Π_{MT} *has overdraft-safety security.*

It remains to show the inequality (3.3). Since we have several extractors in AND composition trying to break DLOG relations, the proof is given for only one extractor. Moreover, we have used the *rewinding* technique for 3-round Σ-protocols which requires *two accepting transcripts*. We can therefore make use of the following lemma [15]:

Lemma 1 (Rewinding Lemma)**.** *Let* $\mathbb{S}$ *and* $\mathbb{T}$ *be finite, non-empty sets, and let* $f : \mathbb{S} \times \mathbb{T} \to \{0, 1\}$ *be a function. Let* X, Y *and* Y' *be mutually independent random variables, where* X *takes value in the set* $\mathbb{S}$*, and* Y *and* Y' *are each uniformly distributed over* $\mathbb{T}$*. Let* $\epsilon := Pr[f(X, Y) = 1]$ *and* $N := |\mathbb{T}|$*. Then*

$$Pr[f(X, Y) = 1 \wedge f(X, Y') = 1 \wedge Y \neq Y'] \geqslant \epsilon^2 - \frac{\epsilon}{N}$$

Intuitively, the probability that $\mathcal{A}''$ generates one accepting transcript is $\epsilon \geqslant 1/N$, with N be the size of the *challenge space*. Further, the probability that $\mathcal{B}$ gets two accepting transcripts is approximately $\epsilon' \approx \epsilon^2$. Applying the lemma (1), we find that the probability that $\mathcal{B}$ gets two accepting transcripts with different challenges is at least $\epsilon' \geqslant \epsilon^2 - \epsilon/N$. We can now prove the inequality (3.3) i.e., $\epsilon' \approx \epsilon^2 \geqslant \epsilon$. Assuming that $\mathcal{B}$ successfully obtained the two accepting transcripts:

$$\epsilon' \geqslant \epsilon^2 - \frac{\epsilon}{N} = \epsilon^2 - \frac{2\epsilon}{N} + \frac{\epsilon}{N} \geqslant (\text{by the fact } \epsilon \geqslant 1/N)$$

$$\geqslant \epsilon^2 - \frac{2\epsilon}{N} + \frac{1}{N^2} = (\epsilon - \frac{1}{N})^2$$

by the assumption that the challenge space is *super-poly*, we get:

$$\epsilon' \geqslant (\epsilon - \frac{1}{N})^2 \implies \sqrt{\epsilon'} + \frac{1}{N} \geqslant \epsilon \implies \epsilon' \approx \epsilon^2$$

This shows that $Adv^{DLOG}_{\mathcal{B},\mathbb{G}} = \epsilon' \approx \epsilon^2$ hence $Adv^{DLOG}_{\mathcal{B},\mathbb{G}} \geqslant Adv^{OVS}_{\mathcal{A},\Pi_{MT}}$.

A "similar" proof strategy is used to prove that Π_{MT} has privacy. Here the strategy involves an adversary attacking the zero-knowledge property of the interactive ZK-proof. Starting from the `privacy` experiment (PRV) presented in 3.2.3, we proceed with the following reductions:

- Given that the elements of tx_{multi} are indistinguishable from random elements under DDH assumption, we modify the PRV experiment with an adversary $\mathcal{A}$ that breaks the zero-knowledge property of the NIZK ZK-proof.
- By the assumption that NIZK ZK-proof inherits the zero-knowledge property of the interactive version, we modify the previous experiment

by taking an adversary $\mathcal{A}'$ attacking the zero-knowledge property of the *interactive* ZK-proof. In particular, from the existence of the simulator $\mathcal{S}$, the advantage of every adversary $\mathcal{A}'$ in the following experiment is defined:

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

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.

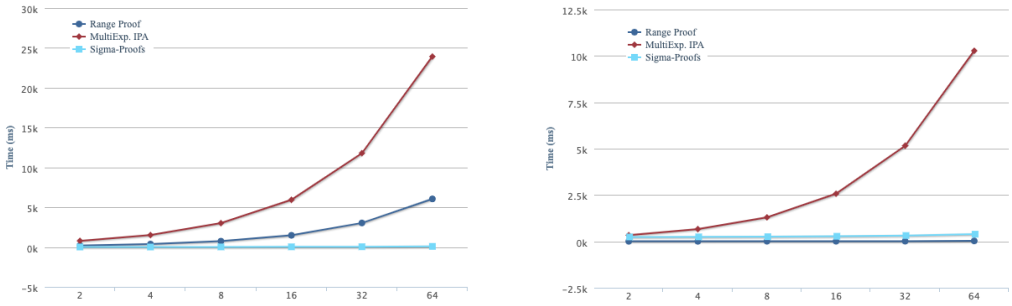
We have shown that the privacy can be reduced to the zero-knowledge property of our interactive ZK-proof. We already proved the existence of several simulators for our ZK-proof, as the following corollary states.

Corollary 2. *In the light of the Theorems 1 and 2 we can say that from the zero-knowledge of our ZK-proof in interactive form, the Multi-Transfer scheme Π_{MT} has the property of privacy.*

An alternative approach to that of guessing an hidden bit b, is to let the adversary $\mathcal{A}'$ to generate the transcripts on its own, rather than obtaining them from the challenger. Then, the adversary $\mathcal{A}'$ could perform a *direct attack* where an honest verifier $\mathcal{V}$ is actively involved. This requires that $\mathcal{A}'$ uses the simulator to prepare an accepting transcript and hope that the $\mathcal{V}$'s challenge matches the prepared one, making $\mathcal{V}$ to accept. To this end, and only if Π_{MT} is HVZK, we can introduce another adversary $\mathcal{B}$, which acts as an elementary wrapper around $\mathcal{A}'$ and runs the simulator for $\mathcal{A}'$. In summary, the idea is to show that transcripts obtained from a challenger do not help $\mathcal{A}'$ because $\mathcal{A}'$ could have generated the transcripts on its own. Finally, the challenge space must be super-poly to ensure security.

3.3 Implementation and Evaluation

In this section, we present our implementation and concrete evaluation of ZeroMT. The implementation of the ZK-proof system is modular and allow us to estimate the execution times of prover and verifier algorithms, as well as the associated proof sizes, for each unit independently. We also provide a comparison of key metrics with concurrent works and an estimate of gas consumption when ZeroMT is implemented on an Ethereum smart contract.



(a) prover (b) verifier

Figure 3.3: Execution times when the number of transfers increases.

3.3.1 Code implementation and benchmarks

The ZeroMT source code written in Rust can be found on GitHub [36] and utilizes `merlin` [73] and `arkworks` [8] ecosystems. These libraries provide a Fiat-Shamir heuristic for building non-interactive proof systems through the STROBE-based transcript [1] 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 3.2, 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. Figure 3.3 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 3.2, 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 3.2: Evaluation of multi-transfer ZK-proof. `Range Proof` refers to the implementation of Protocol 3 in section 3.2.4. Σ-Proof refers to the implementation of Protocol 4 in section 3.2.4. `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	1509	10	1891
8	737	3002	14	3753
16	1477	5930	24	7431
32	3023	11778	43	14844
64	6049	23921	82	30052
Verifier Times (ms)				
2	5	331	12	348
4	5	663	18	686
8	6	1296	27	1329
16	8	2571	47	2626
32	14	5157	88	5259
64	24	10275	161	10460
Proof sizes (bytes)				
2	448	848	416	1712
4	448	976	416	1840
8	448	1104	416	1968
16	448	1232	416	2096
32	448	1360	416	2224
64	448	1488	416	2352

3.3.2 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 [38] and Anonymous Zether [34]. Unfortunately, basic Zether [20] 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 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 3.3.

Table 3.3: 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	$log(n)$	$N + log(n)$	13408	26060
Anon. Zether	2478	152	$log(N)$	N	3776	6148
Our Work (ZeroMT)	7431	2626	$log(n \cdot m)$	N	2096	5104

3.3.3 Ethereum smart contract gas estimation

The ZeroMT verifier can be executed in a blockchain system supporting smart contracts, such as Ethereum. In Ethereum, a smart contract code execution requires an amount of *gas* to be fulfilled. The exact amount of gas required depends on what the smart contract carries out in terms of computation. Understanding the gas amount required for the execution of a ZeroMT verifier within a smart contract might lead to the conclusion that implementing it is unfeasible due to high gas costs. The Ethereum yellow paper [70] provides the gas consumption for every mathematical operation that can be performed natively. In our case, the operations on the BN-254 elliptic curve are supported thanks to the *Ethereum Improvement Proposal* (EIP) 196 of the precompiled smart contracts [58]. Within this EIP, the costs for arithmetic operations over the elliptic curve are included, such as point addition and scalar-point multiplication. Such costs have been successively reduced in various Ethereum hard forks, such as *Istanbul*, with the adoption of the *EIP-1108* [27]. A summary of all the arithmetic operations is shown in table 3.4.

We now present an estimation of the gas consumption of our ZeroMT verifier. The estimation is iterated multiple times with different values of n (bit-range domain) and m (aggregate transfers). For the elliptic curve operations, the costs within the EIP-1108 are considered. Results are shown in table 3.5.

Table 3.4: Required gas amount per operation (units).

Mnemonic	**Type**	**Gas per execution**
`ADD`	Scalar addition	3
`SUB`	Scalar subtraction	3
`MUL`	Scalar multiplication	5
`DIV`	Scalar division	5
`EXP`	Scalar exponentiation	$(exp == 0)?10 :$ $(10 + 10 \cdot (1 + log_{256}(exp)))$
`ECADD`	Curve point addition	500 (EIP-196) - 150 (EIP-1108)
`ECMUL`	Curve point and scalar multiplication	40,000 (EIP-196) - 6,000 (EIP-1108)

Table 3.5: Gas consumption for ZeroMT verifier - EIP-1108 (units).

n	**m**	**Range proof**	**MultiExp. IPA**	**Σ-proofs**	**Total**
16	2	26,334	1,028,736	103,737	1,158,807
16	4	28,337	1,831,373	166,229	2,025,939
16	8	32,500	3,424,779	291,221	3,748,500
16	16	41,134	6,600,191	541,223	7,182,548
16	32	59,143	12,940,691	1,041,267	14,041,101
16	64	96,455	25,613,197	2,041,429	27,751,081
32	2	28,284	1,930,140	103,737	2,062,161
32	4	32,391	3,523,490	166,229	3,722,110
32	8	40,908	6,698,785	291,221	7,030,914
32	16	58,672	13,039,040	541,223	13,638,935
32	32	95,477	25,711,039	1,041,267	26,847,783
32	64	171,575	51,050,300	2,041,429	53,263,304
64	2	32,338	3,721,117	103,737	3,857,192
64	4	40,799	6,896,356	166,229	7,103,384
64	8	58,446	13,236,494	291,221	13,586,161
64	16	95,006	25,908,248	541,223	26,544,477
64	32	170,597	51,247,002	1,041,267	52,458,866
64	64	326,613	101,927,236	2,041,429	104,295,278

For simple comparison, executing the Zether transfer requires 7188k gas consumption for a single transfer, considering $n = 16$ bit-range. In contrast, with ZeroMT we are able to execute up to 16 aggregate transfers with a gas cost of around 7183k.

A note on block size. At the time of writing, each Ethereum block has

an associated *gas limit* value ranging from 15 million to 30 million gas. The results presented in table 3.5 highlight that for values of $n \cdot m > 1,024$ an execution of the ZeroMT verifier in a smart contract would be unfeasible, as an amount of gas greater than 30 million units is needed to verify the ZK-proof. Instead, for lower values of $n \cdot m$ the Ethereum block size would not exceed and a transaction with such ZK-proof would be verified without issues.

3.3.4 Aggregation savings

In the previous sections, we have seen that the ZeroMT costs are mainly for generating and verifying the ZK-proof and the question arises of how convenient is the aggregate proof of multiple transfers. Hence, we compare the costs for generating and verifying an aggregate proof, together with the proof size, to those of executing multiple transfers separately. We have collected these savings in table 3.6.

Table 3.6: Aggregation savings in terms of prover and verifier execution times and proof size.

n	m	Proving time	Verifying time	Proof size
16	2	0 %	0 %	0 %
16	4	37,58 %	37,46 %	63,97 %
16	8	45,69 %	49,22 %	83,41 %
16	16	50,80 %	54,16 %	91,72 %
16	32	52,79 %	57,20 %	95,73 %
16	64	54,84 %	58,14 %	97,77 %
32	2	0 %	0 %	0 %
32	4	34,82 %	34,29 %	64,17 %
32	8	44,56 %	45,44 %	83,58 %
32	16	48,77 %	49,69 %	91,84 %
32	32	50,48 %	51,25 %	95,81 %
32	64	50,67 %	52,29 %	97,82 %
64	2	0 %	0 %	0 %
64	4	33,97 %	35,20 %	64,35 %
64	8	43,61 %	44,90 %	83,73 %
64	16	47,41 %	48,77 %	91,94 %
64	32	49,39 %	50,22 %	95,88 %
64	64	48,65 %	49,66 %	97,86 %

As shown in the table, for $n = 16$ bit-range, 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.

3.4 Applications and Limitations

3.4.1 ZeroMT applications

In this section, we discuss the instantiation of our multi-transfer scheme in real-world scenarios, specifically the smart lock (presented in section 3.1.4) 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 [30]. 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 [16]. 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 [16] 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.

3.4.2 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 [18], 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 [34] primitive could potentially be integrated into our ZK-proof, thereby forming an anonymity set within the multi-transfer scheme.

CHAPTER 4

Σ-IPA: IMPROVED ZERO-KNOWLEDGE PROOFS

4.1 Introduction

The Inner Product Argument (IPA) is an interactive proof between prover $\mathcal{P}$ and verifier $\mathcal{V}$, which engage an argument of knowledge of two vectors of scalars satisfying an *inner product relation*. Bootle et al. in [17] introduce the Inner Product Argument secure under discrete logarithm (DLOG) assumptions. Later, Bünz et al. in [21] propose Bulletproofs (BPs), that are arguments of knowledge for range proofs and general arithmetic circuits and use the IPA as a subroutine. The authors also optimize the IPA using a *folding strategy* with recursive composition, so that the resulting proof has logarithmic size as the size of the statement behind the inner product relation increases. Another interesting property of the BP argument of knowledge is that it does not come with trusted setup, which makes it attractive for trustless cryptographic protocols. From this result, BP has become widely adopted, especially in public blockchain cryptographic protocols, such as anonymous and private transactions. In these contexts, it is worth mentioning Quisquis [38] and Zether [20], which propose private transactions using BP to prove that amounts and balances are non-negative. Similarly, Lelantus [47] and Monero [3] hide the values of input coins, and prove that the outputs in a spend transaction are in the range of admissible values. However, the trade-off of gaining a trustless protocol results in worse scalability and higher fees. This is because trustless comes with non-constant size of the proofs and linear verification time.

An amortization strategy is proposed through our *Zk-based Multi-Transfer scheme* [30, 63], namely *ZeroMT* and presented in chapter 3. Here, the transaction fees can be amortized by performing multiple transfers in a single transaction equipped with an aggregate *Zero-Knowledge proof* (ZK-proof), obtained by combining the aggregation technique of BP and Σ-protocol theory. Hence, the cost that a private transaction would have for a single transfer is now spread across multiple transfers. However, it turns out that the IPA subroutine still weighs significantly on the overall ZK-proof verification time.

With the aim of optimizing the IPA's verification time, some proposals arises such as the one of Bowe et al. in [18]. Here, the authors introduce a new inner-product relation which leads to an argument for polynomial commitment evaluation. Along the same lines, Bünz et al. in [22] propose an amortized succinctness introducing accumulators for polynomial commitment schemes. Applying this technique to the IPA, the verifier asymptotically results in a logarithmic cost barring a single linear time check. In another line of research, there are proposals for the IPA based on pairing-friendly groups, landing into the *inner-pairing product* [33, 23, 50]. However, they may result in expensive pairing operations once applied.

While not specifically aimed at optimizing the IPA, closely related is the compressed Σ-protocol theory of Attema et al. [10, 9]. In their notable works, the authors reconcile the BP compression mechanism with the Σ-protocol theory. This allows the design of RPs or arithmetic circuits within an established theory and in DLOG assumptions, with the same communication complexity of BPs. Their intuition is simple: a prover $\mathcal{P}$ provides a proof of knowledge of a committed vector $\mathbf{x}$ to a verifier $\mathcal{V}$, whose commitment satisfies an arbitrary and public linear relation $L(\mathbf{x})$. We follow this intuition, but in our case we want that $\mathcal{P}$ and $\mathcal{V}$ engage a proof of knowledge of two secret vectors committed under a single compact Pedersen commitment, satisfying a public linear form corresponding to the inner-product relation. Therefore, our specific instantiation of compressed Σ-protocol lies on the problem that the IPA tries to solve. Finally, with respect to the original IPA, we observe that our interactive proof algorithms require fewer expensive computations at the same communication complexity. This leads to a more computationally efficient IPA for prover and verifier algorithms.

4.1.1 Our contribution

We propose a new argument to prove that two secret vectors, which are committed under a single compact Pedersen commitment, satisfy the inner-product relation. To this end, we develop the compressed Σ-Inner-Product Argument (Σ-IPA) following the compressed Σ-protocols theory and the BP folding strategy. We prove security and provide a Σ-IPA that is complete

and has soundness in standard DLOG setting. Finally, we conduct an efficiency analysis showing that our IPA reduces the computational complexity of prover and verifier algorithms by a factor of 2 compared to the BP's IPA.

4.1.2 Related works

Bootle et al. [17] propose an inner product argument where the soundness relies on discrete logarithm assumption in prime order groups. The authors also present the notion of *witness extended emulation* from which the security property of soundness for the IPA is derived. Bünz et al. [21] optimizes the communication complexity of the IPA by a factor of 3, introducing a *folding strategy* in the recursive composition. The authors also propose Bulletproofs (BPs), zero-knowledge arguments for range proofs and general arithmetic circuits based on DLOG assumptions. Despite the proofs have logarithmic size, verification time is linear with respect to the length of the witnesses.

Bowe et al. [18] propose an amortization strategy for the IPA verification time. This strategy follows a different inner-product relation, which turns out to be satisfied by a polynomial evaluation argument. Bünz et al. [22] establish a generalized result from the previous one. The authors demonstrate that any polynomial commitment scheme based on DLOG assumption has an accumulation scheme. With an accumulation scheme, the IPA verifier results in a logarithmic cost barring a final linear time opening check. Here, security is based on random oracle model. In our previous work [65], we apply the new relation of [18] to the IPA of BP and see that it could improve the communication complexity to a constant size. The security inherits that of [18], however the analysis requires further details.

Another side of the research is devoted to the IPA based on pairing-friendly groups and universal setup. In that direction, Daza et al. [33] achieve logarithmic verification complexity in the circuit size based on the work of [17]. Bünz et al. [23] achieve a logarithmic-time verifier for a generalized IPA in pairing settings. Lee [50] proposes an argument of knowledge from inner-pairing products with a transparent setup, where the verifier has an asymptotic logarithmic time plus the cost for a number of pairings. However, pairing operations may result expensive once applied.

Attema et al. [10] propose compressed Σ-protocols, reconciling the BPs compression mechanism with the theory of Σ-protocols and achieving the same communication complexity. The authors provide a general relation for the proof of knowledge of a vector commitment with arbitrary linear form openings. Instead, our proposal essentially develops a compressed Σ-inner-product argument for the original inner-product relation of BPs, thus considering the security properties from the theory of compressed Σ-protocols. A summary of related works is given in table 4.1.

Table 4.1: Proposals for improvement of the IPA.

Reference	Proposal	Result
[17]	Inner-Product-Argument (IPA)	Introduces an interactive proof for an inner-product relation based on DLOG assumption
[21]	Bulletproofs (BP)	Optimizes the IPA verification time of a factor of 3 by using a folding strategy (a recursive composition of the interactive proof)
[18]	Halo	Modifies the inner-product relation resulting in an new interactive proof for a "polynomial evaluation proof" (prove that an evaluation equals a certain value).
[22]	Proof-of-carrying data	Shows that any "polynomial evaluation proof" has an accumulation scheme. With this new strategy the (Halo-style) IPA verifier-time results in a logarithmic cost barring a final linear time check.
[33, 23, 50]	Inner-Pairing-Product	IPA based on pairing-friendly groups and pairing operations, but these operations are not efficient in practice.
[10]	Compressed-Sigma Protocol Theory	Reconciles the folding strategy of BP with Sigma-Protocol theory. This result is shown through an interactive proof (of knowledge) of one committed vector satisfying an arbitrary relation.

4.2 Σ-Inner-Product Argument Protocol

This section presents our *compressed Σ-IPA* and is organized as follows. Section 4.2.1 provides an IPA with inefficient communication and related security proofs. Section 4.2.2 provides an IPA with logarithmic communication, improved computational cost and related security proofs. Section 4.2.3 analyzes the computational costs of the optimized IPA. Section 4.2.4 are some discussions about our result.

4.2.1 Σ-inner-product with constant rounds

We present a 3-move IPA protocol denoted by Π_1 with inefficient communication complexity, shown in protocol 5. The resulting protocol is what we call Σ-IPA, a proof system that satisfies the inner-product relation $\mathcal{R}_{IPA}$ (2.4) discussed in section 2.8.2.

Before engaging in Π_1, the prover and verifier run in a 2-move protocol Π_0 where, upon the verifier samples and sends $y \xleftarrow{\$} \mathbb{Z}_p^*$, both compute

$$T' = T \cdot u^{y \cdot c}$$

with $T = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}$, $c = \langle \mathbf{a}, \mathbf{b} \rangle$ and generators $\mathbf{g}, \mathbf{h}$ public parameters. Then, prover and verifier engage in Π_1 with the inputs substitution $u^y \to u$ and $T' \to T$, as specified in [21].

Protocol 5 Σ-IPA Π_1 for relation 2.4

input: $(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, u, T \in \mathbb{G};\ \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n)$
  $\mathcal{P}$'s input: $(\mathbf{g}, \mathbf{h}, u, T, \mathbf{a}, \mathbf{b})$
  $\mathcal{V}$'s input: $(\mathbf{g}, \mathbf{h}, u, T)$
output: $\mathcal{V}$ accepts or rejects

  $\mathcal{P}$ computes:
    $L = \mathbf{g}_{lo}^{\mathbf{a}_{hi}} \cdot \mathbf{h}_{hi}^{\mathbf{b}_{lo}} \cdot u^{\langle \mathbf{a}_{hi}, \mathbf{b}_{lo} \rangle} \in \mathbb{G}$
    $R = \mathbf{g}_{hi}^{\mathbf{a}_{lo}} \cdot \mathbf{h}_{lo}^{\mathbf{b}_{hi}} \cdot u^{\langle \mathbf{a}_{lo}, \mathbf{b}_{hi} \rangle} \in \mathbb{G}$
  end $\mathcal{P}$
  $\mathcal{P} \rightarrow \mathcal{V} : L, R$
  $\mathcal{V} : x \xleftarrow{\$} \mathbb{Z}_p^*$
  $\mathcal{V} \rightarrow \mathcal{P} : x$
  $\mathcal{P}$ computes:
    $\mathbf{a}' = x \cdot \mathbf{a}_{lo} + \mathbf{a}_{hi} \in \mathbb{Z}_p^{n/2}$
    $\mathbf{b}' = \mathbf{b}_{lo} + x \cdot \mathbf{b}_{hi} \in \mathbb{Z}_p^{n/2}$
  end $\mathcal{P}$
  $\mathcal{P} \rightarrow \mathcal{V} : \mathbf{a}', \mathbf{b}'$
  $\mathcal{V}$ computes and checks:
    $T' = L \cdot T^x \cdot R^{x^2} \in \mathbb{G}$
    $T' \stackrel{?}{=} \mathbf{g}_{lo}^{\mathbf{a}'} \cdot \mathbf{g}_{hi}^{x\mathbf{a}'} \cdot \mathbf{h}_{lo}^{x\mathbf{b}'} \cdot \mathbf{h}_{hi}^{\mathbf{b}'} \cdot u^{\langle \mathbf{a}', \mathbf{b}' \rangle}$
  end $\mathcal{V}$

The protocol Π_1 is inefficient since the communication cost is only reduced by a factor of 2, i.e., the prover sends two vectors $\mathbf{a}', \mathbf{b}'$ of size $n/2$ with respect to the witness size of n. The protocol Π_1 is not required to be zero-knowledge, but it is complete and special-sound as stated in theorem 3.

Theorem 3 (Σ-IPA)**.** *The Σ-protocol Π_1 is a 3-move perfectly complete and 3-special-sound argument for the relation (2.4). Moreover, the communication complexity is:*

- $\mathcal{P} \rightarrow \mathcal{V}$*:* 2 *elements of* $\mathbb{G}$ *and* 2 *elements of* $\mathbb{Z}_p^{n/2}$ *of size* $n/2$ *with respect to the witness size* n*.*
- $\mathcal{V} \rightarrow \mathcal{P}$*:* 1 *element of* $\mathbb{Z}_p^*$*.*

Proof. **Completeness.** Following the definition (18) of *perfect completeness*, if the prover follows the protocol the proof is always accepted. Hence, it is sufficient to show that the following holds.

$$T' = \mathbf{g}_{lo}^{\mathbf{a}'} \cdot \mathbf{g}_{hi}^{x\mathbf{a}'} \cdot \mathbf{h}_{lo}^{x\mathbf{b}'} \cdot \mathbf{h}_{hi}^{\mathbf{b}'} \cdot u^{\langle \mathbf{a}', \mathbf{b}' \rangle}$$

$$= \mathbf{g}_{lo}^{(x\mathbf{a}_{lo} + \mathbf{a}_{hi})} \cdot \mathbf{g}_{hi}^{(x^2\mathbf{a}_{lo} + x\mathbf{a}_{hi})} \cdot \mathbf{h}_{lo}^{(x\mathbf{b}_{lo} + x^2\mathbf{b}_{hi})} \cdot \mathbf{h}_{hi}^{(\mathbf{b}_{lo} + x\mathbf{b}_{hi})} \cdot u^{\langle x\mathbf{a}_{lo} + \mathbf{a}_{hi}, \mathbf{b}_{lo} + x\mathbf{b}_{hi} \rangle}$$

$$= (\mathbf{g}_{lo}^{\mathbf{a}_{hi}} \cdot \mathbf{h}_{hi}^{\mathbf{b}_{lo}}) \cdot (\mathbf{g}_{lo}^{\mathbf{a}_{lo}} \cdot \mathbf{g}_{hi}^{\mathbf{a}_{hi}} \cdot \mathbf{h}_{lo}^{\mathbf{b}_{lo}} \cdot \mathbf{h}_{hi}^{\mathbf{b}_{hi}})^{x} \cdot (\mathbf{g}_{hi}^{\mathbf{a}_{lo}} \cdot \mathbf{h}_{lo}^{\mathbf{b}_{hi}})^{x^2} \cdot$$
$$\cdot u^{x\langle \mathbf{a}_{lo}, \mathbf{b}_{lo} \rangle} \cdot u^{\langle \mathbf{a}_{hi}, \mathbf{b}_{lo} \rangle} \cdot u^{x^2 \langle \mathbf{a}_{lo}, \mathbf{b}_{hi} \rangle} \cdot u^{x \langle \mathbf{a}_{hi}, \mathbf{b}_{hi} \rangle}$$
$$= L \cdot T^{x} \cdot R^{x^2}$$

3**-Special-Soundness.** We follow the definition (19) of *k-special-soundness*. Let $((L, R), x_i, (\mathbf{a'}_i, \mathbf{b'}_i,))$ for $i = 1, ..., 3$, be the accepting transcripts obtained by rewinding the prover three times after the prover sends L, R. Assuming $x_1, x_2, x_3 \in \mathbb{Z}_p$ are pairwise distinct challanges, we can find three values $v_1, v_2, v_3 \in \mathbb{Z}_p$ by defining the Vandermonde matrix

$$M = \begin{pmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ x_1^2 & x_2^2 & x_3^2 \end{pmatrix} \in \mathbb{Z}_p^{3\times 3}$$

with non-zero determinant $(x_2 - x_1)(x_3 - x_1)(x_3 - x_2) \in \mathbb{Z}_p^*$, and by computing

$$\begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = M^{-1} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

since the matrix M is invertible. Now it can be shown that, for each $\mathbf{a'}, \mathbf{b'}$

$$\mathbf{g}_{lo}^{\mathbf{a}_{lo}} \cdot \mathbf{g}_{hi}^{\mathbf{a}_{hi}} \cdot \mathbf{h}_{lo}^{\mathbf{b}_{lo}} \cdot \mathbf{h}_{hi}^{\mathbf{b}_{hi}} \cdot u^{c}$$
$$= \mathbf{g}_{lo}^{\sum_{i=1}^{3} v_i \cdot \mathbf{a'}} \cdot \mathbf{g}_{hi}^{\sum_{i=1}^{3} v_i \cdot x_i \mathbf{a'}} \cdot \mathbf{h}_{lo}^{\sum_{i=1}^{3} v_i \cdot x_i \mathbf{b'}} \cdot \mathbf{h}_{hi}^{\sum_{i=1}^{3} v_i \cdot \mathbf{b'}} \cdot u^{\sum_{i=1}^{3} v_i \cdot \langle \mathbf{a'}, \mathbf{b'} \rangle}$$
$$= L^{v_1 + v_2 + v_3} \cdot T^{v_1 x_1 + v_2 x_2 + v_3 x_3} \cdot R^{v_1 x_1^2 + v_2 x_2^2 + v_3 x_3^2}$$
$$= T$$

Hence, the tuples $\bar{\mathbf{a}} := \sum_{i=1}^{3} (v_i \mathbf{a'}, v_i x_i \mathbf{a'})$, $\bar{\mathbf{b}} := \sum_{i=1}^{3} (v_i x_i \mathbf{b'}, v_i \mathbf{b'})$ and $c = \sum_{i=1}^{3} v_i \cdot \langle \mathbf{a'}, \mathbf{b'} \rangle$ are valid extracted witnesses for relation (2.4). Given that the statement $\mathbf{g}^{\bar{\mathbf{a}}} \mathbf{h}^{\bar{\mathbf{b}}} \cdot u^{c} = T$ holds, this completes the proof. □

We now need one additional rewind for the 2-move protocol Π_0. By the soundness of protocol Π_1 the extractor can obtain witnesses $\mathbf{a}$ and $\mathbf{b}$ such that:

$$T \cdot u^{y \cdot c} = \mathbf{g}^{\mathbf{a}} \mathbf{h}^{\mathbf{b}} u^{y \cdot \langle \mathbf{a}, \mathbf{b} \rangle}.$$

The extractor rewinds and runs the prover with a different challenge y', thus obtaining:

$$T \cdot u^{y' \cdot c} = \mathbf{g}^{\mathbf{a'}} \mathbf{h}^{\mathbf{b'}} u^{y' \cdot \langle \mathbf{a'}, \mathbf{b'} \rangle}.$$

Then, by combining the two equalities we get:

$$\mathbf{g}^{\mathbf{a} - \mathbf{a'}} \mathbf{h}^{\mathbf{b} - \mathbf{b'}} u^{y \cdot \langle \mathbf{a}, \mathbf{b} \rangle - y' \cdot \langle \mathbf{a'}, \mathbf{b'} \rangle} = u^{c \cdot (y - y')}.$$

Hence, either we have found a non-trivial DLOG relation from definition (7), or $\mathbf{a} = \mathbf{a'}$ and $\mathbf{b} = \mathbf{b'}$. In the latter case, it fallows that $u^{(y - y') \langle \mathbf{a}, \mathbf{b} \rangle} = u^{c \cdot (y - y')}$, which implies $c = \langle \mathbf{a}, \mathbf{b} \rangle$. Hence, we have found valid witnesses $\mathbf{a}$ and $\mathbf{b}$ for the statement $T = \mathbf{g}^{\mathbf{a}} \mathbf{h}^{\mathbf{b}} \wedge c = \langle \mathbf{a}, \mathbf{b} \rangle$.

4.2.2 Compressed Σ-inner-product with LOG rounds

From section 4.2.1 it turns out that $\Pi_0 \diamond \Pi_1$ is $(2,3)$-special-sound for the Σ-IPA, following the definition (22) of composable proofs. However, the size of the proof is only reduced by a factor of 2. We can further reduce the proof to a logarithmic size with respect to the witness, following the recursive composition of the *folding strategy* of Bulletproofs. Furthermore, this is a similar strategy that leads the standard Σ-protocols to the *compressed* form of Attema et al. [10]. Thus, we can apply recursion multiple times until the two vectors $\mathbf{a}, \mathbf{b}$ have constant size. Such recursive composition is shown in protocol 6, denoted with Π_2, and considers vectors whose initial size is a power of two, i.e., $n = 2^\mu$ for some $\mu \in \mathbb{N}$.

Protocol 6 Compressed Σ-IPA Π_2 for relation 2.4

input: $(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, u, T \in \mathbb{G};\ \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n)$
  $\mathcal{P}$'s input: $(\mathbf{g}, \mathbf{h}, u, T, \mathbf{a}, \mathbf{b})$
  $\mathcal{V}$'s input: $(\mathbf{g}, \mathbf{h}, u, T)$
output: $\mathcal{V}$ accepts or rejects

  $\mathcal{P}$ computes:
    $n = 2^\mu \in \mathbb{N}$
    $L_1 = \mathbf{g}_{lo}^{\mathbf{a}_{hi}} \cdot \mathbf{h}_{hi}^{\mathbf{b}_{lo}} \cdot u^{\langle \mathbf{a}_{hi}, \mathbf{b}_{lo} \rangle} \in \mathbb{G}$
    $R_1 = \mathbf{g}_{hi}^{\mathbf{a}_{lo}} \cdot \mathbf{h}_{lo}^{\mathbf{b}_{hi}} \cdot u^{\langle \mathbf{a}_{lo}, \mathbf{b}_{hi} \rangle} \in \mathbb{G}$
  end $\mathcal{P}$
  $\mathcal{P} \to \mathcal{V} : L_1, R_1$
  $\mathcal{V} : x_1 \xleftarrow{\$} \mathbb{Z}_p^*$
  $\mathcal{V} \to \mathcal{P} : x_1$
  $\mathcal{P}$ and $\mathcal{V}$ compute:
    $\mathbf{g}^{(2)} = \mathbf{g}_{lo} \circ \mathbf{g}_{hi}^{x_1} \in \mathbb{G}^{n/2}$
    $\mathbf{h}^{(2)} = \mathbf{h}_{lo}^{x_1} \circ \mathbf{h}_{hi} \in \mathbb{G}^{n/2}$
    $T_2 = L_1 \cdot T^{x_1} \cdot R_1^{x_1^2} \in \mathbb{G}$
  end $\mathcal{P}$ and $\mathcal{V}$
  $\mathcal{P}$ computes:
    $\mathbf{a}^{(2)} = x_1 \cdot \mathbf{a}_{lo} + \mathbf{a}_{hi} \in \mathbb{Z}_p^{n/2}$
    $\mathbf{b}^{(2)} = \mathbf{b}_{lo} + x_1 \cdot \mathbf{b}_{hi} \in \mathbb{Z}_p^{n/2}$
  end $\mathcal{P}$
  $\vdots$
  $\mathcal{P}$ computes:
    $\mathbf{a}' := \mathbf{a}^{(\mu-1)}$, $\mathbf{b}' := \mathbf{b}^{(\mu-1)} \in \mathbb{Z}_p^2$
    $\mathbf{g}' := \mathbf{g}^{(\mu-1)}$, $\mathbf{h}' := \mathbf{h}^{(\mu-1)} \in \mathbb{G}^2$
    $L_{\mu-1} = \mathbf{g}'^{\mathbf{a}'_{hi}}_{lo} \cdot \mathbf{h}'^{\mathbf{b}'_{lo}}_{hi} \cdot u^{\langle \mathbf{a}'_{hi}, \mathbf{b}'_{lo} \rangle} \in \mathbb{G}$
    $R_{\mu-1} = \mathbf{g}'^{\mathbf{a}'_{lo}}_{hi} \cdot \mathbf{h}'^{\mathbf{b}'_{hi}}_{lo} \cdot u^{\langle \mathbf{a}'_{lo}, \mathbf{b}'_{hi} \rangle} \in \mathbb{G}$
  end $\mathcal{P}$

29: $\mathcal{P} \rightarrow \mathcal{V} : L_{\mu-1}, R_{\mu-1}$
30: $\mathcal{V} : x_\mu \xleftarrow{\$} \mathbb{Z}_p^*$
31: $\mathcal{V} \rightarrow \mathcal{P} : x_\mu$
32: $\mathcal{P}$ and $\mathcal{V}$ compute:
33: $g := g^{(\mu)} = \mathbf{g'}_{lo} \circ \mathbf{g'}_{hi}^{x_\mu} \in \mathbb{G}$
34: $h := h^{(\mu)} = \mathbf{h'}_{lo}^{x_\mu} \circ \mathbf{h'}_{hi} \in \mathbb{G}$
35: $T_\mu = L_{\mu-1} \cdot T_{\mu-1}^{x_\mu} \cdot R_{\mu-1}^{x_\mu^2} \in \mathbb{G}$
36: **end** $\mathcal{P}$ and $\mathcal{V}$
37: $\mathcal{P}$ computes:
38: $a := a^{(\mu)} = x_\mu \cdot \mathbf{a'}_{lo} + \mathbf{a'}_{hi} \in \mathbb{Z}_p$
39: $b := b^{(\mu)} = \mathbf{b'}_{lo} + x_\mu \cdot \mathbf{b'}_{hi} \in \mathbb{Z}_p$
40: **end** $\mathcal{P}$
41: $\mathcal{P} \rightarrow \mathcal{V} : a, b$
42: $\mathcal{V}$ computes and checks:
43: $c = \langle a, b \rangle$
44: $T_\mu \stackrel{?}{=} g^a \cdot h^b \cdot u^c$
45: **end** $\mathcal{V}$

The protocol Π_2 is efficient in terms of communication costs, that is, it is a $(2\mu+1)$-round protocol with $\mu = log_2(n)$ and n is the witness length. This implies the proof is logarithmic-sized considering that the prover exchanges $(L_1, R_1)...(L_{\mu-1}, R_{\mu-1})$ group element and 2 elements of $\mathbb{Z}_p$ with the verifier, while the verifier sends $log_2(n)$ elements of $\mathbb{Z}_p$.

The protocol Π_2 is complete $(3, ..., 3)$-special-sound argument, always satisfying relation (2.4) as stated in theorem 4, but this time with halved-size witnesses in each round.

Theorem 4 (Compressed Σ-IPA)**.** *The Σ-protocol Π_2 is a $(2\mu+1)$-move perfectly complete and $(3, ..., 3)$-special-sound argument for the relation (2.4). Moreover, the communication complexity is:*

- $\mathcal{P} \rightarrow \mathcal{V}$*:* $2 \cdot (log(n) - 1)$ *elements of* $\mathbb{G}$ *and* 2 *elements of* $\mathbb{Z}_p$*, with respect to the witness size* n*.*
- $\mathcal{V} \rightarrow \mathcal{P}$*:* $log(n)$ *elements of* $\mathbb{Z}_p^*$*.*

Proof. **Completeness.** It follows directly.

$(3, ..., 3)$-Special-Soundness. Following the definition (20), it can be shown that Π_2 is $(k_1, ..., k_\mu)$-special-sound, where $k_i = 3$ for all $i \in [1, \mu]$. That is a generalization of the extractor analysis of Π_1, where now we have $\mu = log_2(n)$. It follows that we can use the same extractor, but this time it takes in total a 3-ary tree of accepting transcripts of depth $log_2(n)$, thus it runs in polynomial time in n. This complete the proof. □

We now can replace the Π_1 protocol with Π_2 and compose $\Pi_0 \diamond \Pi_2$ to obtain a $(2\mu + 3)$-move complete and $(2, 3, ..., 3)$-special-sound IPA with logarithmic-size proof. From definition (22) of composable proofs, the composition is well-defined: the transcript $(y, (\mathbf{a}, \mathbf{b}))$ for Π_0 on public input $(\mathbf{g}, \mathbf{h}, u, T, c)$, is accepting if and only if the tuple $(\mathbf{a}, \mathbf{b})$ is a witness for statement $x = \psi(\mathbf{g}, \mathbf{h}, u, T, c, y) \mapsto (T \cdot u^{y \cdot c})$ with ψ an efficient computation, and thus $(x; (\mathbf{a}, \mathbf{b})) \in \mathcal{R}_{BP}$. Similarly, Π_2 has an efficient computation ψ' and considers accepting transcripts if and only if the witnesses $(\mathbf{a}, \mathbf{b})$ belong to a new relation $\mathcal{R}_{IPA}$ (2.4) with halved-size witnesses at each recursive step.

4.2.3 Efficiency analysis

We now give an analysis of the efficiency of Σ-protocols Π_1 and Π_2 (protocol 5 and 6 respectively) with regards to communication and computational complexity.

The protocol Π_1 has constant rounds but inefficient communication complexity. Indeed, prover sends to the verifier: 2 elements of $\mathbb{G}$ and 2 elements of $\mathbb{Z}_p^{n/2}$ of size $n/2$ with respect to the witness size n. The verifier sends only 1 element of $\mathbb{Z}_p^*$ to the prover. The protocol Π_2 reduces the communication complexity to logarithmic. Indeed, the prover sends to the verifier: $2 \cdot (\log(n) - 1)$ elements of $\mathbb{G}$ and 2 elements of $\mathbb{Z}_p$, with respect to the witness size n. The veirifer sends $\log(n)$ elements of $\mathbb{Z}_p^*$ to the prover.

Finally, protocol Π_2 improves the computational complexity of the BP's IPA by a factor of 2. Indeed, our prover and verifier algorithms execute $\sum_{j=1}^{\log_2(n)} \frac{n}{2^{j-1}}$ exponentiations and 0 inversions for computing the new generators $\mathbf{g}, \mathbf{h}$ in each j-th round. Instead, prover and verifier algorithms of the BP's IPA execute $\sum_{j=1}^{\log_2(n)} \frac{2n}{2^{j-1}}$ exponentiations and 2 inversions in each j-th round.

4.2.4 Further optimizations and discussions

In summary, the steps for constructing the compressed Σ-IPA, thus the steps for optimizing the computational cost of the IPA, are:

- reconcile the IPA with the Σ-Protocol theory, landing into the Σ-IPA;
- optimize the Σ-IPA applying the folding strategy, landing into the compressed Σ-IPA.

The whole strategy can be said in one sentence as: "reconcile the IPA with the compressed Σ-Protocol theory".

In its current form, the compressed Σ-IPA could replace the IPA subroutine in range proofs of protocols such as ZeroMT, Zether, and Monero. This

would lead to benefits such as decreased transaction costs, given the reduced computational cost.

Moreover, the compressed Σ-IPA represents a starting point from which an important theoretical result could be derived. Specifically, one could apply the *multi-exponentiation* optimization of verifier-side to defer exponentiations in the computation of generators (as seen in section 2.8.2) to the final round of the verifier. Moreover, this makes the compressed Σ-IPA appealing to accumulation strategies such as that presented in *proof-of-carrying data* [22]. With such a strategy, the IPA verifier would result in logarithmic asymptotic computational cost. Finally, given the underlying security assumptions of the compressed Σ-IPA, the accumulation strategy could be demonstrated under standard DLOG assumptions, which is not yet present in the state of the art.

PART IV

CONCLUSIONS

CHAPTER 5

CONCLUSIONS & FUTURE WORKS

This thesis is divided into two main contributions: *ZeroMT*, which enhances privacy-preserving cryptographic protocols in the public blockchain, and *compressed* Σ*-IPA*, which focuses on improving trustless zero-knowledge proofs in the standard DLOG setting.

ZeroMT proposes a new secure and private multi-transfer scheme for CT protocols. In summary, ZeroMT enriches the CT protocols 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 and balances 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 is suitable for account-based blockchains and smart contract protocols like Zether. Benefits are also shown in terms of scalability: 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 following the EIP-1108 [27], and when the `Verify` method is executed in a smart contract).

ZeroMT also contributes in optimizing the use of ZK-proofs in CT protocols. 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.

Finally, the concept of multi-transfer can also be extended for applications such as multi-party off-chain protocols and multiple continuous payments.

Thanks to the modularity of its proof system implementation, ZeroMT highlights that the IPA component is expensive in terms of computational costs. Hence, the final part of this thesis proposes the *compressed* Σ*-IPA*, an argument of knowledge of two committed vectors following the compressed Σ-protocols theory and the original IPA relation. Our compressed Σ-IPA maintains the same logarithmic communication complexity of the original IPA, while reducing the computational complexity by a factor of 2. Finally, the compressed Σ-IPA is proved secure in the standard DLOG setting.

Future works. The current version of ZeroMT does not provide transaction unlinkability, that is, senders and receivers are identifiable. A future direction could be to hide the link between senders and receivers from the multi-transfer through a more complex ZK-proof. In this direction, proposals such as [34, 72] could be used to strengthen our multi-transfer scheme by building anonymity sets. For a practical contribution, we could also evaluate ZeroMT in real scenarios involving streams of sensor data [61, 53, 69].

As another future work, we could introduce *accumulators* [22] in the compressed Σ-IPA to enhance the overall verification time with logarithmic efficiency. Finally, such accumulation strategy could be proved under standard DLOG assumptions, a result that is not yet present in the state of the art.

BIBLIOGRAPHY

[1] Strobe protocol framework.

[2] Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. 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*, pages 418–433. Springer, 2002.

[3] Kurt M Alonso et al. Zero to monero, 2020.

[4] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15, 2018.

[5] Elli Androulaki, Ghassan O Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. 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*, pages 34–51. Springer, 2013.

[6] Diego F Aranha, Youssef El Housni, and Aurore Guillevic. A survey of elliptic curves for proof systems. *Designs, Codes and Cryptography*, 91(11):3333–3378, 2023.

[7] Arkworks-rs. docs.rs/ark-bn254/.

[8] Arkworks-rs. github.com/arkworks-rs.

[9] T Attema. *Compressed Σ-protocol theory*. PhD thesis, Leiden University, 2023.

[10] Thomas Attema and Ronald Cramer. Compressed-protocol theory and practical application to plug & play secure algorithmics. In *Annual International Cryptology Conference*, pages 513–543. Springer, 2020.

[11] Thomas Attema, Serge Fehr, and Michael Klooß. 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*, pages 113–142. Springer, 2022.

[12] Paulo SLM Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *International workshop on selected areas in cryptography*, pages 319–331. Springer, 2005.

[13] Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza. Secure sampling of public parameters for succinct zero knowledge proofs. In *2015 IEEE Symposium on Security and Privacy*, pages 287–304. IEEE, 2015.

[14] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 329–349. 2019.

[15] Dan Boneh and Victor Shoup. A graduate course in applied cryptography. *Draft 0.5*, 2020.

[16] EunSeong Boo, Joongheon Kim, and JeongGil Ko. Litezkp: Lightening zero-knowledge proof-based blockchains for iot and edge platforms. *IEEE Systems Journal*, 16(1):112–123, 2021.

[17] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35*, pages 327–357. Springer, 2016.

[18] Sean Bowe, Jack Grigg, and Daira Hopwood. Recursive proof composition without a trusted setup. *Cryptology ePrint Archive*, 2019.

[19] Daniel RL Brown. Sec 2: Recommended elliptic curve domain parameters. *Standards for Efficient Cryptography*, 2010.

[20] Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards privacy in a smart contract world. In *International Conference on Financial Cryptography and Data Security*, pages 423–443. Springer, 2020.

[21] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 315–334. IEEE, 2018.

[22] Benedikt Bünz, Alessandro Chiesa, Pratyush Mishra, and Nicholas Spooner. Proof-carrying data from accumulation schemes. *Cryptology ePrint Archive*, 2020.

[23] Benedikt Bünz, Mary Maller, Pratyush Mishra, Nirvan Tyagi, and Psi Vesely. Proofs for inner pairing products and applications. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 65–97. Springer, 2021.

[24] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform.

[25] Diletta Cacciagrano, Flavio Corradini, Gianmarco Mazzante, Leonardo Mostarda, and Davide Sestili. Off-chain execution of iot smart contracts. In *International Conference on Advanced Information Networking and Applications*, pages 608–619. Springer, 2021.

[26] Diletta Cacciagrano, Flavio Corradini, and Leonardo Mostarda. Blockchain and iot integration for society 5.0. In *International Conference on Society 5.0*, pages 1–12. Springer, 2021.

[27] Antonio Salazar Cardozo and Zachary Williamson. Eip-1108: Reduce alt bn128 precompile gas costs. *Ethereum Improvement Proposals*, (1108), 2018.

[28] Wren Chan and Aspen Olmsted. Ethereum transaction graph analysis. In *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 498–500. IEEE, 2017.

[29] Flavio Corradini, Alessandro Marcelletti, Andrea Morichetta, Andrea Polini, Barbara Re, Emanuele Scala, and Francesco Tiezzi. Model-driven engineering for multi-party business processes on multiple blockchains. *Blockchain: Research and Applications*, 2(3):100018, 2021.

[30] Flavio Corradini, Leonardo Mostarda, and Emanuele 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.

[31] Ivan Damgård, Oded Goldreich, Tatsuaki Okamoto, and Avi Wigderson. Honest verifier vs dishonest verifier in public coin zero-knowledge

proofs. In *Annual International Cryptology Conference*, pages 325–338. Springer, 1995.

[32] Ivan B Damgård. Interactive hashing can simplify zero-knowledge protocol design without computational assumptions. In *Annual International Cryptology Conference*, pages 100–109. Springer, 1993.

[33] Vanesa Daza, Carla Ràfols, and Alexandros Zacharakis. Updateable inner product argument with logarithmic verifier and applications. In *IACR International Conference on Public-Key Cryptography*, pages 527–557. Springer, 2020.

[34] Benjamin 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.

[35] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

[36] EmanueleSc. github.com/emanuelesc/zeromt.

[37] Ethereum. go-ethereum/crypto/bn256.

[38] Prastudy Fauzi, Sarah Meiklejohn, Rebekah Mercer, and Claudio 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.

[39] Amos Fiat and Adi 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.

[40] Amos Fiat and Adi 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.

[41] Michael Fleder, Michael S Kester, and Sudeep Pillai. Bitcoin transaction graph analysis. *arXiv preprint arXiv:1502.01657*, 2015.

[42] Oded Goldreich et al. Probabilistic proof systems: A primer. *Foundations and Trends® in Theoretical Computer Science*, 3(1):1–91, 2008.

[43] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 285–306. 2019.

[44] Jens Groth. On the size of pairing-based non-interactive arguments. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 305–326. Springer, 2016.

[45] Zhangshuang Guan, Zhiguo Wan, Yang Yang, Yan Zhou, and Butian Huang. Blockmaze: An efficient privacy-preserving account-model blockchain based on zk-snarks. *IEEE Transactions on Dependable and Secure Computing*, 2020.

[46] Feng Hao. Schnorr non-interactive zero-knowledge proof. Technical report, 2017.

[47] Aram Jivanyan. Lelantus: Towards confidentiality and anonymity of blockchain transactions from standard assumptions. *IACR Cryptol. ePrint Arch.*, 2019:373, 2019.

[48] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2020.

[49] Taechan Kim and Razvan Barbulescu. Extended tower number field sieve: A new complexity for the medium prime case. In *Annual international cryptology conference*, pages 543–571. Springer, 2016.

[50] Jonathan Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In *Theory of Cryptography Conference*, pages 1–34. Springer, 2021.

[51] Lindell. Parallel coin-tossing and constant-round secure two-party computation. *Journal of Cryptology*, 16:143–184, 2003.

[52] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM (JACM)*, 39(4):859–868, 1992.

[53] Nadeem Qaisar Mehmood, Rosario Culmone, and Leonardo Mostarda. Modeling temporal aspects of sensor data for MongoDB NoSQL database. *Journal of Big Data*, 4(1), March 2017.

[54] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, Manubot, 2019.

[55] Shen Noether, Adam Mackenzie, et al. Ring confidential transactions. *Ledger*, 1:1–18, 2016.

[56] Andrew Poelstra. Mimblewimble. 2016.

[57] Mayank Raikwar and Danilo Gligoroski. Dos attacks on blockchain ecosystem. In *European Conference on Parallel Processing*, pages 230–242. Springer, 2021.

[58] Christian Reitwiessner. Eip 196: Precompiled contracts for addition and scalar multiplication on the elliptic curve alt_bn128. *Ethereum Improvement Proposals, Ethereum, Zug, Switzerland, Tech. Rep*, 196, 2017.

[59] Dorit Ron and Adi Shamir. Quantitative analysis of the full bitcoin transaction graph. In *International Conference on Financial Cryptography and Data Security*, pages 6–24. Springer, 2013.

[60] Antoine Rondelet and Michal Zajac. Zeth: On integrating zerocash on ethereum. *arXiv preprint arXiv:1904.00905*, 2019.

[61] Giovanni Russello, Leonardo Mostarda, and Naranker Dulay. A policy-based publish/subscribe middleware for sense-and-react applications. *Journal of Systems and Software*, 84(4):638–654, April 2011.

[62] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE, 2014.

[63] Emanuele Scala, Changyu Dong, Flavio Corradini, and Leonardo Mostarda. 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*, pages 461–472. Springer, 2023.

[64] Emanuele Scala, Changyu Dong, Flavio Corradini, and Leonardo Mostarda. Zeromt: Towards multi-transfer transactions with privacy for account-based blockchain. *Journal of Information Security and Applications*, 83:103794, 2024.

[65] Emanuele Scala and Leonardo Mostarda. Range proofs with constant size and trustless setup. In *International Conference on Advanced Information Networking and Applications*, pages 301–310. Springer, 2023.

[66] Emanuele Scala and Leonardo Mostarda. Efficient inner-product argument from compressed σ-protocols and applications. In *International Conference on Advanced Information Networking and Applications*, pages 326–337. Springer, 2024.

[67] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology—CRYPTO'89 Proceedings 9*, pages 239–252. Springer, 1990.

[68] Adi Shamir. Ip= pspace. *Journal of the ACM (JACM)*, 39(4):869–877, 1992.

[69] Claudia Vannucchi, Michelangelo Diamanti, Gianmarco Mazzante, Diletta Cacciagrano, Rosario Culmone, Nikos Gorogiannis, Leonardo Mostarda, and Franco Raimondi. Symbolic verification of event–condition–action rules in intelligent environments. *Journal of Reliable Intelligent Environments*, 3(2):117–130, February 2017.

[70] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[71] Alex Luoyuan Xiong, Binyi Chen, Zhenfei Zhang, Benedikt Bünz, Ben Fisch, Fernando Krell, and Philippe Camacho. Veri-zexe: Decentralized private computation with universal setup. *Cryptology ePrint Archive*, 2022.

[72] Tianyu Zheng, Shang Gao, Yubo Song, and Bin Xiao. Leaking arbitrarily many secrets: Any-out-of-many proofs and applications to ringct protocols. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 2533–2550. IEEE, 2023.

[73] zkcrypto. Merlin: composable proof transcripts for public-coin arguments of knowledge.

APPENDIX A

INNER-PRODUCT ARGUMENT SYSTEM

A.1 IPA first protocol

Protocol 7 Inner-product argument first protocol

input: $(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, T \in \mathbb{G}, c \in \mathbb{Z}_p;\ \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n) : T = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} \wedge c = \langle \mathbf{a}, \mathbf{b} \rangle$
    $\mathcal{P}$'s input : $(\mathbf{g}, \mathbf{h}, T, c, \mathbf{a}, \mathbf{b})$
    $\mathcal{V}$'s input : $(\mathbf{g}, \mathbf{h}, T, c)$
output: $\mathcal{V}$ accepts or rejects

$\mathcal{V} : y \xleftarrow{\$} \mathbb{Z}_p^*$
$\mathcal{V} \rightarrow \mathcal{P} : y$
$\mathcal{P}$ and $\mathcal{V}$ compute:
    $T' = T \cdot u^{y \cdot c} \in \mathbb{G}$
end $\mathcal{P}, \mathcal{V}$
Run *IPA second recursive protocol* on Input $(\mathbf{g}, \mathbf{h}, u^y, T'; \mathbf{a}, \mathbf{b})$

A.2 IPA second recursive protocol

Protocol 8 Inner-product argument second recursive protocol

input: $(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, u, T \in \mathbb{G};\ \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n) : T = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}u^{\langle \mathbf{a},\mathbf{b} \rangle}$
$\mathcal{P}$'s input : $(\mathbf{g}, \mathbf{h}, u, T, \mathbf{a}, \mathbf{b})$
$\mathcal{V}$'s input : $(\mathbf{g}, \mathbf{h}, u, T)$
output: $\mathcal{V}$ accepts or rejects

if $n == 1$:
    $\mathcal{P} \rightarrow \mathcal{V} : a, b$
    $\mathcal{V}$ computes and verifies:
        $c = a \cdot b \in \mathbb{Z}_p$
        $T \stackrel{?}{=} g^a h^b u^c \in \mathbb{G}$ — if yes, $\mathcal{V}$ accepts; otherwise it rejects
    end $\mathcal{V}$
else if $n > 1$:
    $\mathcal{P}$ computes:
        $n' = n/2$
        $c_L = \langle \mathbf{a}_{[:n']}, \mathbf{b}_{[n':]} \rangle \in \mathbb{Z}_p$
        $c_R = \langle \mathbf{a}_{[n':]}, \mathbf{b}_{[:n']} \rangle \in \mathbb{Z}_p$
        $L = \mathbf{g}_{[n':]}^{\mathbf{a}_{[:n']}} \cdot \mathbf{h}_{[:n']}^{\mathbf{b}_{[n':]}} \cdot u^{c_L} \in \mathbb{G}$
        $R = \mathbf{g}_{[:n']}^{\mathbf{a}_{[n':]}} \cdot \mathbf{h}_{[n':]}^{\mathbf{b}_{[:n']}} \cdot u^{c_R} \in \mathbb{G}$
    end $\mathcal{P}$
    $\mathcal{P} \rightarrow \mathcal{V} : L, R$
    $\mathcal{V} : x \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$
    $\mathcal{V} \rightarrow \mathcal{P} : x$
    $\mathcal{P}$ and $\mathcal{V}$ compute:
        $\mathbf{g}' = \mathbf{g}_{[:n']}^{x^{-1}} \circ \mathbf{g}_{[n':]}^{x} \in \mathbb{G}^{n'}$
        $\mathbf{h}' = \mathbf{h}_{[:n']}^{x} \circ \mathbf{h}_{[n':]}^{x^{-1}} \in \mathbb{G}^{n'}$
        $T' = L^{x^2} \cdot T \cdot R^{x^{-2}} \in \mathbb{G}$
    end $\mathcal{P}, \mathcal{V}$
    $\mathcal{P}$ computes:
        $\mathbf{a}' = x \cdot \mathbf{a}_{[:n']} + x^{-1} \cdot \mathbf{a}_{[n':]} \in \mathbb{Z}_p^{n'}$
        $\mathbf{b}' = x \cdot \mathbf{b}_{[n':]} + x^{-1} \cdot \mathbf{b}_{[:n']} \in \mathbb{Z}_p^{n'}$
    end $\mathcal{P}$
    Recursively run *IPA second protocol* on input $(\mathbf{g}', \mathbf{h}', u, T'; \mathbf{a}', \mathbf{b}')$

A.3 IPA with multi-exponentiation

Protocol 9 Inner-product argument with multi-exponentiation

input: $(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, u, T \in \mathbb{G};\ \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n) : T = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}u^{\langle \mathbf{a},\mathbf{b}\rangle}$

$\mathcal{P}$'s input : $(\mathbf{g}, \mathbf{h}, u, T, \mathbf{a}, \mathbf{b})$

$\mathcal{V}$'s input : $(\mathbf{g}, \mathbf{h}, u, T)$

output: $\mathcal{V}$ accepts or rejects

if $n == 1$: *jth round*

$\mathcal{P} \rightarrow \mathcal{V} : a, b$

$\mathcal{V}$ computes and verifies:

$c = a \cdot b \in \mathbb{Z}_p$

$\mathbf{s} = \{s_1, \ldots, s_n\} \in \mathbb{Z}_p^{m \cdot n}$ where:

$\forall i \in [1, n] : s_i = \prod_{j=1}^{log_2(m \cdot n)} x_j^{b(i,j)}$

$$b(i,j) = \begin{cases} 1 & \text{if bit } (j-1) \text{ of } (i-1) \text{ is } 1 \\ -1 & \text{otherwise} \end{cases}$$

$g = \prod_{i=1}^{n} g_i^{s_i} \in \mathbb{G}$

$h = \prod_{i=1}^{n} h_i^{1/s_i} \in \mathbb{G}$

$T \stackrel{?}{=} g^a h^b u^c \in \mathbb{G}$ if yes, $\mathcal{V}$ accepts; otherwise it rejects

end $\mathcal{V}$

else if $n > 1$:

$\mathcal{P}$ computes, where $n' = n/2$:

$L = \mathbf{g}_{[n':]}^{\mathbf{a}_{[:n']}} \cdot \mathbf{h}_{[:n']}^{\mathbf{b}_{[n':]}} \cdot u^{\langle \mathbf{a}_{[:n']}, \mathbf{b}_{[n':]} \rangle} \in \mathbb{G}$

$R = \mathbf{g}_{[:n']}^{\mathbf{a}_{[n':]}} \cdot \mathbf{h}_{[n':]}^{\mathbf{b}_{[:n']}} \cdot u^{\langle \mathbf{a}_{[n':]}, \mathbf{b}_{[:n']} \rangle} \in \mathbb{G}$

end $\mathcal{P}$

$\mathcal{P} \rightarrow \mathcal{V} : L, R$

$\mathcal{V} : x \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$

$\mathcal{V} \rightarrow \mathcal{P} : x$

$\mathcal{P}$ computes:

$\mathbf{g}' = \mathbf{g}_{[:n']}^{x^{-1}} \circ \mathbf{g}_{[n':]}^{x} \in \mathbb{G}^{n'}$

$\mathbf{h}' = \mathbf{h}_{[:n']}^{x} \circ \mathbf{h}_{[n':]}^{x^{-1}} \in \mathbb{G}^{n'}$

$T' = L^{x^2} \cdot T \cdot R^{x^{-2}} \in \mathbb{G}$

$\mathbf{a}' = x \cdot \mathbf{a}_{[:n']} + x^{-1} \cdot \mathbf{a}_{[n':]} \in \mathbb{Z}_p^{n'}$

$\mathbf{b}' = x \cdot \mathbf{b}_{[n':]} + x^{-1} \cdot \mathbf{b}_{[:n']} \in \mathbb{Z}_p^{n'}$

Recursively runs *IPA MultiExp* on input $(\mathbf{g}', \mathbf{h}', u, T'; \mathbf{a}', \mathbf{b}')$

end $\mathcal{P}$

$\mathcal{V}$ computes:

$T' = L^{x^2} \cdot T \cdot R^{x^{-2}} \in \mathbb{G}$

Recursively runs *IPA MultiExp* on input $(\mathbf{g}, \mathbf{h}, u, T')$

end $\mathcal{V}$