Institut für Verteilte Systeme
Universität Ulm

CONFIDENTIAL TYPES IN TRANSACTION SYSTEMS

DISSERTATION

zur Erlangung des Doktorgrades Dr. rer. nat. der
Fakultät für Ingenieurwissenschaften, Informatik und Psychologie der
Universität Ulm

vorgelegt von

FELIX THEODOR ENGELMANN

aus Summertown

2022

Amtierender Dekan
Prof. Dr.-Ing. Maurits Ortmanns

Gutachter
Prof. Dr. rer. nat. Frank Kargl (Universität Ulm)
Prof. Dr. rer. nat. Andreas Peter (Universität Oldenburg)

Tag der Promotion
17.01.2022

KONTAKT

fe-research@nlogn.org
felix.nlogn.org

ABSTRACT

Motivation.

Decentralized transaction systems enable a network of participants to jointly keep track of balances for tokens which may represent a physical good such as gold. Once the network grows beyond a few peers, agreeing on a single understanding of value becomes challenging. A natural solution to this problem for most systems is to support different currencies. While providing this functionality, the system has to protect the anonymity and confidentiality of transactions to maintain the privacy of the participants. Detailed insight into financial data allows accurate profiling.

Problem statement.

Our goal is to propose a transaction system which has the functionality of separate currencies, known as types. Transactions must not reveal the types of tokens involved and provide sender as well as recipient anonymity. Having isolated types soon requires exchanges which efficiently and securely trade tokens of different types. Still, the confidentiality and anonymity has to be maintained.

Approach.

To build a privacy-preserving multi-type transaction system we formalize the exact security and privacy requirements as a rigorous definition. Based on the requirements we construct the system partially from existing components. To achieve specific properties, we design our own building blocks which are of general use. Finally we prove that our constructions satisfy all requirements and implement a proof of concept to show practicability.

Result.

As a result, we present a framework of components which can be differently assembled to achieve a privacy-preserving multi-type transaction system. The components fulfill the two critical invariants of a transaction: every spending operation must be authorized and the total supply must remain constant. For both, we propose efficient non-interactive zero-knowledge (NIZK) proofs. Additionally, we describe a new aggregatable signature scheme for decentralized exchanges. Complementary to proving our constructions formally, we build a possible application for digital license management on the basis of our system.

Conclusion & contributions.

Our work enables secure and confidential bookkeeping in scenarios without a trusted third party. It serves as a foundation platform for multiple applications to come. Our main contributions are 1. an anonymously aggregatable signature scheme 2. an asset conservation NIZK when using succinct commitments 3. the formalization of a privacy-preserving multi-type transaction system 4. the formalization and security definitions of a privacy-preserving multi-type transaction system which supports exchange operations with 5. efficient constructions for the components of our systems which are used in 6. a confidential license management system.

ZUSAMMENFASSUNG

Motivation.

Dezentrale Transaktionssysteme ermöglichen es einem Netzwerk von Teilnehmern, gemeinsam die Verteilung von Tokens zu verwalten, die ein physisches Gut wie Gold darstellen können. Sobald das Netzwerk über einige wenige Teilnehmer hinauswächst, wird es schwierig, sich auf ein einheitliches Verständnis von Wert zu einigen. Eine naheliegende Lösung für dieses Problem besteht für die meisten Systeme darin, verschiedene Währungen zu unterstützen. Dabei muss das System die Anonymität und Vertraulichkeit der Transaktionen schützen, um die Privatsphäre der Teilnehmenden zu wahren. Detaillierte Einblicke in die Finanzdaten ermöglichen umfangreiches Profiling.

Problemstellung.

Unser Ziel ist es ein Transaktionssystem zu erstellen, das die Funktionalität von separaten Währungen, so genannten Typen, unterstützt. Transaktionen dürfen den Typ der beteiligten Tokens nicht preisgeben und müssen sowohl dem Sender als auch dem Empfänger Anonymität bieten. Separate Typen führen bald zur Notwendigkeit eines Handels, der es ermöglicht, effizient und sicher Tokens verschiedener Typen umzutauschen. Die Vertraulichkeit und Anonymität muss jedoch gewahrt bleiben.

Eigener Ansatz.

Zum Aufbau eines datenschutz-freundlichen Mehrtyp-Transaktionssystems formalisieren wir die genauen Sicherheits- und Datenschutzanforderungen in Form einer strikten Definition. Basierend auf den Anforderungen konstruieren wir unser System aus teilweise bestehenden Komponenten. Um bestimmte, notwendige Eigenschaften zu erreichen, entwerfen wir darüber hinaus unsere eigenen Komponenten, die auch außerhalb unserer Anwendung nützlich sind. Schließlich beweisen wir, dass unsere Konstruktionen alle Anforderungen erfüllen und implementieren einen Proof of Concept, um die Praktikabilität zu zeigen.

Ergebnis.

Als Ergebnis präsentieren wir einen Framework an Komponenten, die unterschiedlich zusammengesetzt werden können, um datenschutzkonforme Multi-Typ-Transaktionssysteme zu erhalten. Die Komponenten erfüllen die beiden kritischen Teile einer Transaktion: jeder Zahlvorgang muss autorisiert sein und die Gesamtbilanz muss konstant bleiben. Für beides schlagen wir effiziente nicht-interaktive zero-knowledge-Beweise (NIZK) vor. Zusätzlich beschreiben wir ein neues aggregierbares Signaturverfahren für dezentrale Börsen. Ergänzend zum formalen Beweis unserer Konstruktionen entwickeln wir auf der Grundlage unseres Systems eine mögliche Anwendung für digitales Lizenzmanagement.

Zusammenfassung der Beiträge.

Unsere Arbeit ermöglicht eine sichere und vertrauliche Buchführung in Szenarien ohne vertrauenswürdige Drittparteien. Sie dient als Basisplattform für zahlreiche künftige Anwendungen. Unsere wichtigsten Beiträge sind 1. ein anonymes aggregierbares Signaturverfahren 2. einen NIZK zum Beweis der Gesamtbilanzerhaltung bei Verwendung kompakter Commitments 3. die Formalisierung eines Privatsphäre bewahrenden Mehrtyp-Transaktionssystems 4. die Formalisierung und Sicherheitsdefinitionen eines datenschutzfreundli-

chen Multityp-Transaktionssystems, das Handel ermöglicht zusammen mit einer 5. effiziente Konstruktionen für die Komponenten unserer Systeme, die in 6. einem vertraulichen Lizenzmanagementsystem verwendet werden..

CONTENTS

1 INTRODUCTION

1.1 MULTI-TYPE TRANSACTION SYSTEMS

Transacting money between participants of a society enabled trade beyond one-to-one exchange of value. This possibility accelerated progress in many areas, as people were able to focus on a single task and payed other people for services and goods.

One basic property of this system was that not everyone agreed on a common understanding of value. This led to multiple groups, each with their own currency. Our current global society still uses this concept of national currencies where sovereign states have control over the money used in their territory. This allows adjustments on policies best matched to the local circumstances.

In the described setting, each currency can use a transaction system which is suitable for their policies. Traditionally, there exist e.g. precious metals which are naturally limited or objects representing value like fiat money regulated by an authority. A drawback of such isolated systems is the incompatibility with neighbors. While the trade of goods is preferable, e.g. by geographic proximity, the two separate monetary systems have to interact. A common solution is to create an overarching system which respects the individual policies, but provides a standardized exchange of currencies. For fiat currencies, an example is the SWIFT[1] system.

Currently, most of the society trusts the policy makers and banks to follow their rules. An alternative, which requires less trust, is a publicly verifiable ledger of transactions. Publishing the log of all transfers allows the general public to verify that the bank did not corrupt the intended transactions. The only trust left in banks is the ordering of transactions and publishing an append-only log. Due to privacy concerns and compliance, there is no such system operating. If the participants of the system cannot agree on an entity for these tasks, consensus based protocols are a viable alternative. The combination of a transaction system and an ordering mechanism results in what is commonly known as a Blockchain [Nako8].

Ledger based transaction systems encounter the same issue as classical monetary systems. Separate groups like to have different policies, such as inflation rate, and control over their currency. Allowing separate, interoperable currencies in a ledger based transaction system requires support for a currency identifier for each amount. To abstract from the concrete currencies, we separate different policy regions by a type. Instead of representing a currency, type amount pairs are usable in non monetary transaction systems. As a consequence, each transaction in a multi-type ledger transaction system must not only specify an amount to be transferred, but also requires the type to make sure that an amount is always interpreted in the correct type.

1 https://www.swift.com

An already existing multi-type transaction system is the Ethereum ecosystem which enables the co-existence of tokens in newly defined types. Any participant in the system can create tokens of a new type and use them independently of the differently typed tokens. Interaction between tokens of different types works through decentralized exchanges which atomically swap the tokens of the respective types between owners.

1.2 CONFIDENTIALITY & ANONYMITY

The advantage of reduced trust in a central entity with public ledger based transaction systems comes at the cost of full transparency. All transactions are publicly visible for analysis by everyone. Full insight into everyone's financial details by everyone is not desirable and lowers the acceptance of such a system compared, to e.g., cash with its preferential anonymity.
Cryptographic tools enable transaction systems which allow public verification of transactions without compromising the anonymity of the parties involved and maintaining the confidentiality of the transaction content. Senders of a transaction should remain anonymous, such that a verifier is unable to efficiently differentiate if two transactions were created by the same or different parties. The same applies for recipients. Only the intended recipient can detect that they are the beneficiary of a transaction.
Given perfect sender and receiver anonymity, publishing a transaction with the amount and type visible might still be enough to recover information about senders and receivers. Every transaction output is linkable to all inputs with the same amount and type, which in the worst case for a unique amount is a one-to-one relation. The initial releases of Monero [Noe15] and Zerocoin [MGGR13] solved this by allowing only a single denomination. If all input and outputs have the same amount and no type, all output-input relations are possible and the public amounts (all constant) do not deanonymize users. To maintain the functionality of sending large sums without thousands of inputs to a transaction, Monero below version 0.10.0 used denominations in powers of two, each with their own disjoint anonymity set. Given around 20 denominations, most amounts are representable and the anonymity is only reduced by a factor of 20 as an observer learns which denomination was used. In later versions of Monero and Zerocash [SCG+14], amounts are hidden such that all inputs again share the full anonymity set.
A similar anonymity reduction exists with publicly visible types. If types of inputs and outputs are public, each type has it's own anonymity set. This is especially problematic for types with few transactions resulting in a tiny anonymity set. To prevent such attacks, it is important to have confidential amounts and types.
Therefore, transactions contain all important information only in a hidden form, e.g. as cryptographic commitments. This creates the issue that transactions are no longer publicly verifiable to check that they follow all rules. Instead of verifying the rules directly, another option is to require the sender to prove that the hidden information is correct without revealing it. This is pos-

sible with non-interactive zero-knowledge proofs (NIZK) [RS92]. A verifier only learns if a statement, e.g. the transaction, is valid or not. The NIZK assures that the transaction is valid according to public rules, such as no amount is created out of thin air and the sender has sufficient inputs to spend.
In summary, a multi-type privacy-preserving transaction system provides the following properties:

- The anonymity of the sender in a transaction is preserved. Especially the recipients in a transaction should not learn who initiated the transfer. We differentiate perfect sender anonymity where a transaction is equally likely to be created by any participant and statistical sender anonymity, which hides the sender in a smaller anonymity set.
- The recipients of a transaction must remain hidden from anyone except the sender. Senders specify recipients in their own transaction but are unable to detect the same recipients in a foreign transaction.
- All amounts are confidential and only the required parties (sender & receiver) get access to the plaintext values.
- Equivalent to amounts, types are confidential.

Two existing transaction systems, Monero [Noe15] and Zcash [SCG+14], each have a transaction NIZK optimized for slightly different parameters. While Zcash achieves perfect sender anonymity at the cost of a trusted setup, Monero only achieves statistical sender anonymity but has a transparent setup. Transparent setups are preferable as all system parameters are publicly generated while a trusted setup requires some randomness to be destroyed after generating the parameters. The statistical sender anonymity hides true senders within a subset of all possible ones. This is anonymous as long as the majority of senders do not collude. Both existing systems only support the first three properties and lack the notion of type confidentiality.

1.3 RESEARCH QUESTIONS

From both the classical financial systems and fully public ledger based transaction systems, we recognize the need for interoperable multi-type transactions. Concerns about fully transparent finances require a confidential and anonymous transaction system. Our goal is to provide a multi-type confidential transaction system which provides anonymity and confidentiality in the presence of types.
This leads to the first research question. Given the existing privacy-preserving systems which have no notion of types,

How to integrate confidential types into private transaction systems?

The main challenge of this research question is maintaining sender and receiver anonymity. The information about types must not increase an adversary's probability of deanonymizing senders or receivers. On a technical level,

the NIZK has to support a more complex conservation compared to summing up inputs and subtracting outputs of a single type. This requires a new and efficient protocol.
Once we have a multi-type transaction system, different types can co-exist and share the anonymity set. For interoperability, the participants still have to rely trusted external parties to exchange value between different types. To provide a self contained transaction system, we require a mechanism for participants to exchange tokens of different types without an external trusted party. This raises the second research questions:

How to enable fair (both parties get what they agreed upon) and anonymous trading in a multi-type system?

Especially, is a built-in exchange system able to provide fair trades without revealing the trading parties? Any change of token ownership is only fixed once a transactions is persisted on the ledger. This motivates performing trades with the least number of transactions. Is it possible to have single transaction exchanges of token ownership atomically executed? Using a database terminology, we call atomically executed exchanges of tokens atomic swaps. While multiple parties could jointly create a transaction in a multi round protocol, the challenge is to require only a single transmission from each participant.

1.4 CONTRIBUTIONS

In this work we present a solution in the form of a **framework to create multi-type transaction systems with fair, efficient exchanges**. We focus on systems with transparent setups, like Monero, which provide statistical sender anonymity. With the current proving systems for transparent setups, perfect sender anonymity is not efficiently achievable as the membership proofs for each input within the whole set of possible inputs have either linear run-time in the size of the set [Dia21] or a large coefficient for random oracle based trees like Merkle trees [GKR+21].
The NIZK of privacy-preserving transactions fulfills two main properties. First it assures that the sender of a transaction authorizes the spending of their funds and second, conservation requires that receivers get exactly the amount the sender spends. With both properties combined, the transactions are considered correct. Both parts have to be adjusted to support confidential types in a transaction.
For a rigorous security and privacy analysis of the resulting schemes, we present a **formalization of multi-type transaction systems** with and without swaps [EMP+21]. Swaps exchange tokens between parties where no one can abort the protocol with an unfair advantage.
There already exist commitment schemes for types which have favorable homomorphic properties, allowing type-separated computation on commitments. Unfortunately they are not space efficient, which is important as all transactions need to be persisted. Therefore we propose to use a succinct commitment scheme to store amounts, types and other attributes. For an existing

type homomorphic commitment (THC) scheme [PBF+17] and the succinct commitment scheme, we present matching **anonymous authorization and conservation NIZK schemes**.
These NIZK schemes enable transactions with confidential types, but exchanging tokens of different types requires an external trusted entity. To support integrated, fair, anonymous exchanges, we present an **anonymously aggregatable signature scheme** [EMP+21] where the authorization signatures of different signers are merged by an untrusted party which then applies a signature to assure conservation to get a final transaction.
Depending on the requirements of the deployment (with or without swap support, homomorphic commitments required), the presented versions of the components (conservation and authorization NIZKs and aggregatable signature) may be combined to achieve the best performance. Table 1.1 provides an overview of the resulting transaction sizes for the possible combinations. To summarize, the succinct commitments achieve smaller transaction sizes as THC based commitments. The reduction of the linear cost by a factor of two significantly reduces the transaction size which is composed of the NIZK proof (approximately 1 kB) and the outputs. While the proof size is similar for THC and succinct commitments due to its logarithmic dependency, the outputs of the transaction require only half the space. For a transaction with 10 outputs, this reduces the transaction size from about 1.5 kB to 1.25 kB. Support for anonymous swaps requires a linear cost for each additional input compared to a logarithmic one for systems without swaps. The transaction only system is asymptotically as efficient as a type unaware system.

Table 1.1: Sizes of our multi-type transactions and the state-of-the-art transaction system without type support. We use m inputs, n outputs and an anonymity set of size r.

	transactions only	with swap support
THC	$O(\log(rm+n))+2n$	$O(m\log(r+n))+2m+2n$
succinct com	$O(\log(rmn+n))+nO(m\log(r+n)+mn)+m+n$	
no-type	$O(\log(rm+n))+n$	—

While the applications in the financial domain are evident, we additionally sketch how our multi-type system provides a foundation for a **confidential licensing system** [ESG+21]. Thereby we demonstrate the flexible use-cases enabled by confidential types.

1.5 OVERVIEW

Most contributions of this thesis are published in peer reviewed conferences. To have a precise and detailed presentation of our contributions, we use verbatim sections from our papers. An overview of our papers and ongoing work is summarized in Figure 1.1.

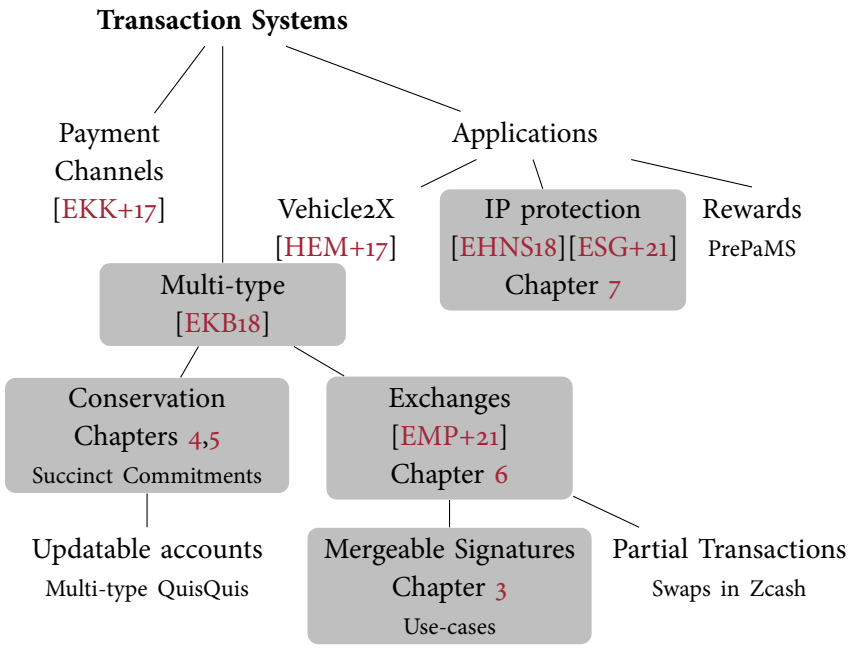


Figure 1.1: Overview of our research areas and our published papers. Contributions to this thesis are highlighted in gray along with chapter references. Ongoing projects are noted in small print.

In Chapter 2 we introduce the technical notation and present existing schemes. We continue with the building blocks, namely the **anonymously aggregatable signature scheme** in Chapter 3 which is described in its general form for applicability outside of transaction systems. The second building block is the **asset conservation NIZK** in Chapter 4 for our succinct commitment scheme for confidential types.
Given both building blocks, we present an efficient **multi-type privacy-preserving transaction scheme** in Chapter 5 and a transaction scheme which supports fair **swaps** in Chapter 6.
In Chapter 7 we discuss the **license management application** based on our multi-type transaction system.
We conclude our findings in Chapter 8 and point to future research topics which are prompted by this work.

2

2.1 RELATED WORK ON PRIVACY-PRESERVING DECENTRALIZED TRANSACTION SYSTEMS

The first generation of ledger based decentralized transaction systems like Bitcoin [Nako8] and Ethereum [Woo+14] only provides pseudonymity. Once a pseudonym is linked to a real world identity, all actions of this pseudonym are linkable [RH13; MPJ+13; RS13; FKP15; OKH13]. The possibility for any real identity to control nearly unlimited different pseudonyms allows for mixing based anonymity. Mixing systems like Mixcoin [BNM+14] or CoinShuffle [RMK14] create a transaction where a set of pseudonyms redistribute their tokens among each other, hiding the ownership in the set of participants. So, instead of an explicit history of a token, each mixing transaction introduces uncertainty about which history is the true one. Conveniently such mixing solutions can be deployed as smart contracts on top of existing systems like Möbius [MM18] on Ethereum. All mixers have the drawback that they require active participation and finding enough peers to mix with.
To avoid this limitation, Monero [Noe15] and Zcash [SCG+14] have transactions where senders hide themselves in an anonymity set of other participants without requiring their interaction. Together with the methods of confidential transactions [Max15] to hide the transferred amounts, both systems achieve high levels of anonymity. Without revealing who actually performed a transaction, past transactions cannot be discarded as they are still relevant.
Pruning information and maintaining a state size independent of the number of transactions is achieved by Zether [BAZB20] and QuisQuis [FMMO19]. They maintain the anonymity guarantees of Monero or Zcash but provide means to update the global state such that old transactions are no longer needed.

2.2 GLOSSARY

To clarify several key terms we use throughout the thesis, we present a description along with synonyms for each of them.

TOKEN A token is the smallest amount transferrable in a transaction system. It represents some base value.

TYPE (syn. Currency) A type specifies a domain in which tokens have a common meaning and value.

ASSET (syn. Typed-Token) An asset or typed-token is the tuple of a scalar, specifying the amount of tokens, and their type, e.g. 5€.

TRANSACTION A transaction consumes multiple assets as inputs and outputs a new set of assets. Importantly no assets are created or destroyed in a transaction.

INPUTS A transaction consumes inputs. Each input specifies at least an amount of tokens consumed and has an owner. If not implicitly given, an input additionally specifies the type of the tokens.

OUTPUT (syn. One-Time-Account) An output describes one beneficiary of a transaction. It specifies a new owner of the included tokens, if required with a type specification.

2.3 PRELIMINARIES

In this chapter, we provide an overview of existing work related to our research. This presents an opportunity to establish a common notation which is then used throughout the thesis.
There are multiple methods of keeping track of balances for a set of participants. One option is for everyone to have a single scalar representing their balance. A transaction then reduces the balance of one party and increases the other party's balance by the same amount. Importantly neither change of balance must be executed alone as otherwise the system gets imbalanced. A more robust system is to keep track of all past transactions and the current balance is calculated by adding up all incoming transactions and subtracting all outgoing ones. A common way to show the recipient of a transaction that the sender has a high enough balance is to reference a previous incoming transaction destined to the sender. If this incoming transaction was not yet spent, the recipient now owns the amount of this transaction.
We introduce a generic transaction system based on unspent transaction outputs (UTXO) and then show how such systems are built in a privacy-preserving way. An UTXO transaction system consists of transactions with inputs and outputs which each hold some assets. A transaction must conserve all value, i.e. the assets consumed by the inputs must be equal to the assets distributed in the outputs. It therefore only changes the owners of the assets. To relate all transactions to each other, the inputs of a transaction are references to outputs of previous transactions as seen in Figure 2.1. This is often represented as a directed acyclic graph (DAG) of transactions. If the transaction system assures that each output is referenced exactly once as input, no asset is spent twice, also known as a double spend. This property leads to the U in UTXO, such that only unspent (without a reference) outputs are usable as inputs. The total supply is updated by changing the set of unspent outputs from an outside mechanism. A common method in cryptocurrencies is to reward participants who to help maintain the network, known as miners. According to public rules, miners are allowed to create new UTXO which are usable in subsequent transactions just like any other transaction output. This increases the total supply of tokens in the system. To prevent theft, the

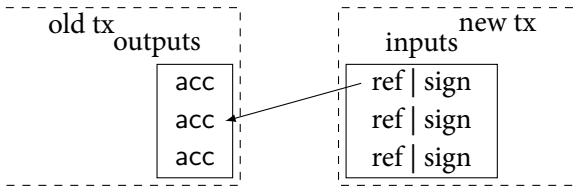


Figure 2.1: UTXO transactions referencing previous outputs as inputs. Each output is a one-time account and inputs are references together with a authorizing signature.

transaction creator has to include a valid signature to authorize the spending of the inputs.

A common architecture of privacy-preserving UTXO systems is to use one-time accounts to achieve recipient anonymity. Transaction outputs to the same recipient are unlinkable to each other and only the long term secret key of the recipient is able to detect them. This works by each participant having a long term public key. To specify a participant as new owner of an output, the sender uses a public algorithm to derive a one-time public key from the long term public key. Only the owner of the long term secret key is able to detect if a one-time key is derived from their key and then have an algorithm to recover the matching one-time secret key. We present existing constructions for two such schemes in Section 2.11.1.

Given unique output accounts, sender anonymity works by referencing not only the real input, but include a number of additional inputs, called mixins, which are not spent but provide an anonymity set for the sender. The two largest deployed privacy-preserving transaction systems (Monero [Noe15] and Zcash [SCG+14]) both use one-time accounts for recipient anonymity but differ in the selection of the sender anonymity set. Zcash stores all previous transaction outputs in a Merkle tree and provides a zero-knowledge inclusion proof for each input. This means that Zcash transactions achieve perfect sender anonymity, as all previous transaction outputs are possible candidates. The inclusion zero knowledge proof requires at least $\log_2(n)$ hash operations for n possible inputs. The non-arithmetic structure of hash functions leads to expensive proofs which are only efficiently instantiated by a succinct non-interactive arguments of knowledge (SNARK). SNARKs require linear proving time in the size of the arithmetic circuit describing the hash functions but they have the benefit of achieving constant verification time and proof transcript size. The trade off to these preferable properties is that to use SNARKs, the system has to generate a structured common reference string (CRS) with an unknown trapdoor. Whoever has knowledge of the trapdoor can fabricate arbitrary proofs. In the distributed setting of a transaction system, where participants have no common trusted party, generating this CRS is difficult as the input randomness must remain secret. Zcash's approach was running a multi-party protocol to spread the setup in a ceremony around the world under supervision of the

press and the public. Convincing the public that the multi party peers did not collude is difficult which dampens the trust in the whole system [1].
Another option is to use zero knowledge proof systems which work without a trusted setup, i.e. the CRS has no trapdoor and is created in public. One of the most efficient system, Bulletproofs [BBB+18], achieves linear runtime for generation and verification with a logarithmic proof size. While the storage requirements are low, the runtime prevents their usage for zero knowledge hashing in Merkle tree inclusion proofs.
To keep the computational requirement within reasonable bounds, transaction systems like Monero use a statistical model where the anonymity set of a transaction is a randomly sampled subset of all previous outputs.
Given the sender anonymity, inputs and outputs of a transaction are still linkable through their asset. Therefore all assets are stored in the outputs as commitments. A zero knowledge proof attached to each transaction assures that the assets in the output commitments are equal to the assets in the referenced input commitments. This convinces a verifier that no assets are created out of thin air.
Currently, Zcash and Monero use commitments to scalar values, representing the amount of a common asset, e.g. XMR or ZEC tokens. The conservation rule of a transaction ensures that the sum of amounts in the inputs is equal to the sum of amounts in the outputs. As our goal is to support multiple confidential types of tokens, we require commitments to a scalar and a type. The scalar represents the amount and the type the currency. In a setting with multiple types involved in a transaction, the amounts have to be balanced for each type individually. We discuss two existing approaches for commitments to a typed amount, namely Confidential Assets [PBF+17] and Cloaked Assets [And], and show the need for an improved multi-type system.
In the remainder of this chapter we first provide the cryptographic foundations for our protocols, which consists of zero-knowledge arguments of knowledge and efficient instantiations such as Bulletproofs and their extensions. Given these tools we present a formalization of a generic privacy-preserving UTXO chain and detail a concrete construction for Ring Confidential Transactions (RingCT) used by Monero and alike. Independent of RingCT systems we present related work about confidentiality of types. In combination with types, we investigate related work on privacy-friendly trading of assets.
Table 2.1 introduces the notation common to all protocols.

2.4 ASSUMPTIONS

Through the remainder of the thesis, we base our constructions on a computationally difficult problem which has a trapdoor. We rely on a cyclic group $\mathbb{G}$ with order q and a generator G. In the multiplicative notation, the group operation is the multiplication of elements, e.g. $G \cdot G'$. Thereby it holds that for any $\gamma \in \mathbb{Z}_q$, $G^\gamma \in \mathbb{G}$.

1 https://electriccoin.co/blog/the-design-of-the-ceremony/

Table 2.1: Notation

$\odot, \bigodot$	homomorphic operation for commitments
$\{a_i\}_{i=1}^{n}$	sets of size n are ordered if not specified otherwise
$[a]$	natural numbers up to $[a] := (1, \dots, a)$
$\vec{c}[i]$	i-th element of vector $\vec{c}$
$\mathbf{E}[i]$	i-th column vector of matrix $\mathbf{E}$
$\vec{x}^n := (x^0, x^1, \dots, x^{n-1})$	length n vector with x a scalar
$\vec{x}^\top$	transposed vector
$x \xleftarrow{\$} X$	x is uniformly randomly sampled from X
$\sum A := \sum_{a \in A} a$	sum of all elements in the set
$\prod A := \prod_{a \in A} a$	product of all elements in the set
$\{a_i\}_{i=1}^{k} \Leftrightarrow \vec{a} = (a_1, \dots, a_k)$	we interchangeably write ordered sets as vectors
$\mathbf{E}[j, i] := \mathbf{E}[i][j]$	row-column indexing of a 2D matrix
$\vec{a} \otimes \vec{b} := (a_1 \cdot \vec{b}, a_2 \cdot \vec{b}, \dots)$	Kronecker product of two vectors
$\mathsf{vec}(\mathbf{E}) := (E[1] \| E[2] \| \dots)$	concatenation of each column vector
$\mathsf{bin}(a) := (a_0, \dots, a_\beta)$	binary decomposition such that $a = \sum_{i=0}^{\beta} a_i 2^i$
$\vec{a} \circ \vec{b} = (a_1 \cdot b_1, a_2 \cdot b_2, \dots)$	element wise multiplication
$\vec{a}^{\circ \vec{b}} := (a_1^{b_1}, a_2^{b_2}, \dots)$	element wise exponentiation

The discrete logarithm assumption states that given G and G^γ, it is infeasible to calculate γ. To limit any party of the system from calculating γ by brute forcing all q values, adversaries are usually limited to perform only a polynomial number of computations in λ, where λ is the security parameter. These algorithms are abbreviated as probabilistic polynomial time (ppt), as they may use randomness and are not necessarily deterministic. As a concrete example, where this assumption holds, we use elliptic curve systems (ECC) but the constructions are not limited to ECC. To achieve a practical security level of 128 bit equivalent symmetrical security, we use the Curve 25519 [2].
We denote a negligible function in the security parameter λ as $\mathsf{negl}(\lambda)$ if for every positive polynomial $\mathsf{poly}(\lambda)$, there exists a N such that for all $\lambda > N$it holds that $|\mathsf{negl}(\lambda)| < \frac{1}{\mathsf{poly}(\lambda)}$.

2.5 COMMITMENTS

We use commitments to store values hidden from interested parties. A commitment is often described as a letter which is sealed in an envelope. Once sealed, the value inside cannot be changed and only the creator of the letter knows what is inside. Formally a commitment is defined as follows.

2 https://doc.dalek.rs/curve25519_dalek/

Definition 2.1 (Commitment)**.** *A commitment scheme consists of the PPT algorithms* $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$ *which takes the security parameter* λ *and outputs public parameters* pp *implicitly given to* $\mathsf{Commit}(a; r) \to \mathsf{com}$ *which takes a value* $a \in \mathbb{M}$ *from a message domain* $\mathbb{M}$ *and randomness* $r \in \mathbb{S}$ *from a randomness domain* $\mathbb{S}$ *(often* $\mathbb{M} = \mathbb{S}$*) and outputs a commitment* com*.*

The commitment has to satisfy the binding and hiding properties. A hiding commitment keeps the committed value secret from anyone seeing the commitment.

Definition 2.2 (Hiding)**.** *A commitment scheme is hiding if for any ppt adversary* $\mathcal{A}$*, it holds that*

$$\Pr\left[\begin{array}{r} b' \leftarrow \mathcal{A}(\mathsf{com}) \\ b' = b \end{array} : \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), b \xleftarrow{\$} \{0,1\}, r \xleftarrow{\$} \mathbb{S} \\ a_0, a_1 \leftarrow \mathcal{A}(\mathsf{pp}) \\ \mathsf{com} \leftarrow \mathsf{Commit}(a_b; r) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

A binding commitment prevents the creator of it to find two values which result in the same commitment. Thereby the creator is not able to change their mind after creating the commitment.

Definition 2.3 (Binding)**.** *A commitment scheme is binding if for any ppt adversary* $\mathcal{A}$*, it holds that*

$$\Pr\left[\begin{array}{r} \mathsf{com}_0 \leftarrow \mathsf{Commit}(a_0; r_0) \\ \mathsf{com}_1 \leftarrow \mathsf{Commit}(a_1; r_1) \\ \mathsf{com}_0 = \mathsf{com}_1 \\ \wedge a_0 \neq a_1 \end{array} : \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ a_0, r_0 \\ a_1, r_1 \end{array} \leftarrow \mathcal{A}(\mathsf{pp})\right] \leq \mathsf{negl}(\lambda)$$

Definition 2.4 (Homomorphism)**.** *A commitment scheme is homomorphic, if it holds that*

$$\mathsf{Commit}(a_0; r_0) \odot \mathsf{Commit}(a_1; r_1) = \mathsf{Commit}(a_0 + a_1; r_0 + r_1)$$

.

A common construction are Pedersen commitments [Ped91], which define $\mathsf{Commit}(a, r) := G^a H^r$ for two generators $G, H \in \mathbb{G}$.

2.5.1 *Vector Pedersen Commitments*

A generalized form of the Pedersen commitment is the vector Pedersen Commitment which commits to a vector of elements $\vec{a}$ instead of a single scalar a. Instead of two generators G and H, the commitment uses $|\vec{a}| + 1$ generators $G_1, \ldots, G_{|\vec{a}|}, H$ and commitment is defined as $\mathsf{Commit}(\vec{a}, r) := H^r \prod_{i=1}^{|\vec{a}|} G_i^{a_i}$

2.6 ZERO-KNOWLEDGE ARGUMENTS OF KNOWLEDGE

Many of our proposed transaction protocols rely on the fact that a transaction creator is able to convince other participants in the system that the proposed transaction is valid according to some agreed rules. One option is to reveal all values required to verify the rules. This is in contradiction to the confidentiality we want to achieve. We utilize zero-knowledge arguments of knowledge to keep the values confidential and still be able to convince others that the transaction satisfies the rules everyone agreed on.
A zero-knowledge Argument on Knowledge is an interactive protocol between a prover P and a verifier V. The goal of the prover is to convince the verifier that a given statement is true without revealing anything else about the statement. A basic example of such a statement is the knowledge of a secret key for a given public key. The prover convinces the verifier about the knowledge of the matching secret key. More formally an argument of knowledge consists of three interactive, ppt algorithms Setup, P, V. From the transcript of the interaction, written as $\langle P, V \rangle$, the verifier decides if they are convinced about the statement.
Let R be a relation such that a statement stmt and a witness wit is in the relation if the desired conditions hold. We write this in the form of a language $\mathcal{L} = \{\mathsf{stmt} \exists \mathsf{wit} \text{ s.t. } (\mathsf{stmt}, \mathsf{wit}) \in R\}$. For the example with the secret key, our statement is the public key, the witness is the secret key, and R is true for all matching pairs of secret and public keys.
The interaction between prover and verifier is an argument of knowledge if it is complete and has knowledge soundness. The completeness requires that for each valid statement and witness pair, the prover is able to convince the verifier.

Definition 2.5. *An Argument of knowledge is perfectly complete if for any adversary $\mathcal{A}$, it holds that*

$$\Pr\left[\begin{array}{rl} (\mathsf{stmt}, \mathsf{wit}) \notin R & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ \vee \langle P(\mathsf{stmt}, \mathsf{wit}), V(\mathsf{stmt}) \rangle = 1 & : \quad (\mathsf{stmt}, \mathsf{wit}) \leftarrow \mathcal{A} \end{array}\right] = 1$$

Knowledge soundness in general is the property that the prover is only able to convince the verifier, if the prover actually knows a correct witness. One possibility to show that the prover has a valid witness is to construct an extractor. The extractor interacts with the prover and must reconstruct the witness used by the prover. This contradicts the intuition that the verifier is only able to know if a condition holds or not. Therefore the extractor gets an additional power. It is allowed to rewind the prover to a specific point in the interaction and supply a different message, creating another transcript, which is equal to the first up to the point of rewinding. Given a polynomial number of transcripts, the extractor must provide a valid witness whenever the prover is able to convince the verifier.
A formalization of the extractor was defined by Groth and Ishai [GI08] and Lindell [Lin03] as computational witness-extended emulation.

Definition 2.6 (Witness-extended emulation)**.** *An Argument of knowledge has witness-extended emulation if there exists an efficient (ppt) extractor $\mathcal{E}$ such that for any adversaries $\mathcal{A}_1, \mathcal{A}_2$ it holds that*

$$\left| \Pr\left[\mathcal{A}_1(tr) = 1 \ : \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), (\mathsf{stmt}, \mathsf{wit}) \leftarrow \mathcal{A}_2(\mathsf{pp}) \\ tr \leftarrow \langle P(\mathsf{stmt}, \mathsf{wit}), V(\mathsf{stmt}) \rangle \end{array} \right] - \Pr\left[\begin{array}{r} \mathcal{A}_1(tr) = 1 \\ \wedge tr \text{ is accepted} \end{array} \ : \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{stmt}, \mathsf{wit}) \leftarrow \mathcal{A}_2(\mathsf{pp}) \\ (tr, \mathsf{wit}') \leftarrow \mathcal{E}^O(\mathsf{stmt}) \end{array} \right] \right| \leq \mathsf{negl}(\lambda)$$

where the extractor has oracle access to the prover P with rewinding.

The above definitions (2.5,2.6) so far do not assure that the verifier learns nothing about the witness, except that it is in the relation or not. One approach to show that the verifier learns nothing about the witness is to replace the prover with a simulator. The simulator has no access to the witness and needs to present an accepting proof for any statement. Similar to the extractor, this intuition is contradicting with the property that a prover needs to know a valid witness. Again, the simulator therefore gets additional power. The simulator is provided with all the messages of the verifier at the beginning of the interaction. Knowing all future replies of the verifier, the simulator is able to craft the messages in such a way that the transcript looks like it was performed between a prover and a verifier. In fact, if the transcript from real provers and simulators are indistinguishable, the prover does not leak the witness.
This definition of zero-knowledge only works if all verifier messages are independent of the prover messages and known in advance. Therefore the argument has to be public coin.

Definition 2.7 (Public Coin)**.** *An argument of knowledge is public coin, if all messages of the verifier are uniformly random and independent of the provers messages.*

Given an public coin argument of knowledge, we can formally define the simulatability of the protocol. The adversary should be unable to have an advantage in differentiating the transcript of the real interaction from the one generated by the simulator.

Definition 2.8 (Special Honest-Verifier Zero-Knowledge)**.** *A public coin argument of knowledge is special honest-verifier zero-knowledge, if there exists an efficient simulator* Sim *such that for all adversaries $\mathcal{A}_1, \mathcal{A}_2$ and the verifiers randomness ρ, it holds that*

$$\begin{aligned} &\Pr\left[\begin{array}{r} (\mathsf{stmt}, \mathsf{wit}) \in R \\ \wedge \mathcal{A}_1(tr) = 1 \end{array} \ : \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), (\mathsf{stmt}, \mathsf{wit}) \leftarrow \mathcal{A}_2(\mathsf{pp}) \\ tr \leftarrow \langle P(\mathsf{stmt}, \mathsf{wit}), V(\mathsf{stmt}, \rho) \rangle \end{array} \right] \\ = &\Pr\left[\begin{array}{r} (\mathsf{stmt}, \mathsf{wit}) \in R \\ \wedge \mathcal{A}_1(tr) = 1 \end{array} \ : \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), (\mathsf{stmt}, \mathsf{wit}) \leftarrow \mathcal{A}_2(\mathsf{pp}) \\ tr \leftarrow \mathsf{Sim}(\mathsf{stmt}, \rho) \end{array} \right] \end{aligned}$$

2.6.1 *Example*

To demonstrate the properties and notation of a zero-knowledge proof, we present a small example. The goal is convince a verifier that the prover knows the secret key to a given public key. Let $\mathbb{G}$ be a cyclic group with order q and a generator G in which the discrete logarithm assumption holds. A key pair is represented as a secret key $x \in \mathbb{Z}_q$ and its public key $G^x \in \mathbb{G}$. The language for this relation is defined as

$$\mathcal{L} = \{\mathsf{stmt} = P \exists \mathsf{wit} = x \text{ s.t. } G^x = P\}$$

The prover and the verifier have access to the statement (P) and engage in a sigma protocol as shown in Figure 2.2.
First the prover generates a random key pair $x', P' = G^{x'}$ and sends the public key to the verifier. The verifier samples a random element γ in $\mathbb{Z}_q$ to satisfy the public coin property (Def 2.7). Given the challenge γ, the prover calculates $z = x' - \gamma x$ such that the verifier can check the relation $P^\gamma G^z = P'$. If that equality holds, the verifier is convinced that the prover knows x.

Σ	
Prover$(P = G^x, x)$	**Verifier**(P)
$x' \xleftarrow{\$} \mathbb{Z}_q$	
$P' \leftarrow G^{x'}$	$\xrightarrow{P'}$
	$\xleftarrow{\gamma} \gamma \xleftarrow{\$} \mathbb{Z}_q$
$z \leftarrow x' - \gamma x$	$\xrightarrow{z}$
	$P^\gamma G^z = G^{\gamma x} G^{x' - \gamma x} = G^{x'} \stackrel{?}{=} P'$

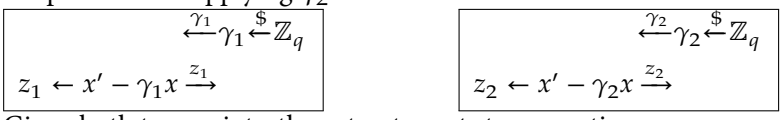
Figure 2.2: Example Sigma protocol

We proceed by proving that the protocol satisfies all properties of a zero knowledge argument of knowledge. The completeness is given because there are no undefined equations and G^x is defined for any $x \in \mathbb{Z}_q$. To show knowledge soundness, we construct an efficient extractor which is able to reconstruct the witness from rewinding access to the prover. The extractor runs the prover until the prover submits the random public key P':

Σ	
Prover$(P = G^x, x)$	**Verifier**(P)
$x' \xleftarrow{\$} \mathbb{Z}_q$	
$P' \leftarrow G^{x'}$	$\xrightarrow{P'}$

Note that now x and x' is fixed for the prover by P and P'. The extractor now proceeds in two ways to generate two transcripts. This is achieved by first

supplying the prover with γ_1 to which the prover replies z_1 and then rewinding the prover and supplying γ_2.



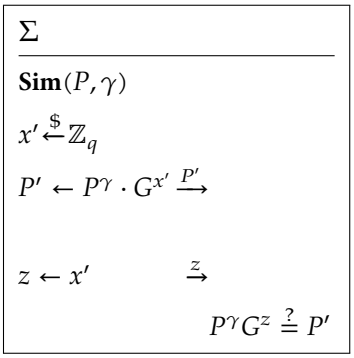
Given both transcripts, the extractor gets two equations

$$z_1 = x' - \gamma_1 x \text{ and } z_2 = x' - \gamma_2 x$$

which have the solution

$$x = \frac{z_1 - z_2}{\gamma_2 - \gamma_1}.$$

If the extractor selected the challenges such that $\gamma_2 - \gamma_1 \neq 0$, which happens with high probability, the resulting x is the witness, satisfying $G^x = P$.
We show the zero-knowledge property of the sigma protocol by constructing a simulator, which presents accepting transcripts indistinguishable from real prover interactions. The simulator has access to the statement (the public key P) and the randomness γ of the verifier. The simulator crafts the first message P' such that it is still uniformly random because of the x' but includes P^γ. P' is identically distributed as from a real prover. As second message, the simulator reveals the x' which is again identically distributed to the z presented by a prover. The verification equation always holds.



As the real and simulated transcripts and indistinguishable, the sigma protocol is zero-knowledge.

2.7 NON-INTERACTIVE ARGUMENTS

In many scenarios, it is impractical for the prover to convince a single verifier. Instead a prover prefers to generate a transcript, which may be verified by any verifier. A recurring example is that a transaction requires a proof that the committed amounts are correct. The transaction creator should not have to run the interactive protocol with every participant in the system to convince them that the transaction is valid.
One approach to this problem is to act as prover and verifier at the same time. The resulting transcript then serves as non-interactive proof that the prover has a valid witness. As we assume an honest verifier, the protocol is no longer

knowledge sound. A malicious prover knows the verifier randomness in advance and just uses the simulator to generate a valid proof for any statement. To force the prover to act honestly in the role of the verifier, the verifier messages must only become known at the time of communication. In our sigma protocol, that means that γ must only be known, once P' is transmitted to the verifier.
The verifier messages are all distributed uniformly at random. This allows the verifier to generate them deterministically from all previously received messages. Using a random oracle, the resulting verifier message cannot be calculated by the prover in advance. This transformation of an interactive honest public coin verifier into a random oracle is known as Fiat Shamir transformation [FS].
In our example of the sigma protocol, we use a hash function $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_q$ as random oracle and achieve a non-interactive argument of knowledge of the form:

Σ

Prover$(P = G^x, x)$		**Verifier**(P)
$x' \xleftarrow{\$} \mathbb{Z}_q$		
$P' \leftarrow G^{x'}$	$\xrightarrow{P'}$	
	$\xleftarrow{\gamma}$	$\gamma \leftarrow \mathsf{H}(P \Vert P')$
$z \leftarrow x' - \gamma x$	$\xrightarrow{z}$	
		$P^\gamma G^z = G^{\gamma x} G^{x' - \gamma x} = G^{x'} \stackrel{?}{=} P'$

A non-interactive zero-knowledge (NIZK) argument of knowledge (AoK) is defined by the following three algorithms:

$\mathsf{pp} \leftarrow \mathsf{AoK}[\mathcal{L}]\mathsf{Setup}(1^\lambda)$ takes the security parameter λ and a language $\mathcal{L}$ and outputs the matching public parameters of the AoK as well as possible parameters required for the specific language.

$\pi \leftarrow \mathsf{AoK}[\mathcal{L}]\mathsf{Prove}(\mathsf{stmt}, \mathsf{wit})$ takes a statement stmt and a witness wit for the language $(\mathsf{stmt}, \mathsf{wit}) \in R_\mathcal{L}$ where $R_\mathcal{L}$ is the relation of $\mathcal{L}$, and outputs a transcript π, serving as proof.

$b \leftarrow \mathsf{AoK}[\mathcal{L}]\mathsf{Verify}(\pi, \mathsf{stmt})$ takes a proof π and a statement stmt for the language $\mathcal{L}$ and outputs a bit b depending on the validity of the proof.

2.8 SIGNATURES OF KNOWLEDGE

Given a zero knowledge argument of knowledge as an interactive protocol, a Signature of Knowledge (SoK) is constructed by including a message in the random oracle compiler of the Fiat-Shamir transform described above. Thereby, the argument of knowledge is bound to a specific message and serves as a signature that is valid, if the signer has knowledge of a valid witness. The

conditions are described by a NP language $\mathcal{L}$ which parameterizes the SoK. For a SoK to be secure, it must – similar to an Argument of Knowledge – be complete, public coin, simulatable and extended-witness emulatable as defined by Chase et al. [CL06]. A SoK consists of the following three algorithms:

$\mathsf{pp} \leftarrow \mathsf{SoK}[\mathcal{L}]\mathsf{Setup}(1^\lambda)$ takes the security parameter λ and a language $\mathcal{L}$ and outputs the matching public parameters of the SoK as well as possible parameters required for the specific language.

$\sigma \leftarrow \mathsf{SoK}[\mathcal{L}]\mathsf{Sign}(\mathsf{stmt}, \mathsf{wit}, m)$ takes a statement stmt and a witness wit for the language $\mathcal{L}$ as well as a message m and outputs a signature σ.

$b \leftarrow \mathsf{SoK}[\mathcal{L}]\mathsf{Verify}(\sigma, \mathsf{stmt}, m)$ takes a signature σ and a statement stmt for the language $\mathcal{L}$ together with a message m and outputs a bit b depending on the validity of the signature.

2.9 BULLETPROOFS

After describing the general framework of zero-knowledge proofs, we present an overview of one concrete protocol to realize zero knowledge proofs. The example of Section 2.6.1 and similar sigma protocols enable proving complex conditions. However they have a linear transcript size. If we want to prove knowledge of two secret keys, the transcript will be twice as large. A more space efficient proof system is Bulletproofs [BBB+18]. The core of the Bulletproof protocol is a proof which is not zero-knowledge, but convinces a verifier that the inner product of two scalar vectors committed in a vector Pedersen commitment is equal to some value. Instead of the expected proof size linear in the length of the vectors, the protocol recursively folds the vectors in half and thereby requires only logarithmically many communication rounds, each with a constant size. This leads to logarithmically sized transcripts.
The disadvantage of the efficient inner product argument is that the verifier learns the content of the vectors and thereby the protocol is not zero-knowledge. Bulletproofs therefore build an outer protocol which encodes and hides the witness in such a way that even though the verifier learns the vectors of the inner product, they are unable to extract the witness. The original Bulletproof paper proposed outer protocols for generic arithmetic circuits and more popular, a highly optimized version for range proofs. A range proof convinces the verifier that a committed value is in a range between 0 and $2^\beta - 1$ for some integer β (64 in most systems).
The outer protocols share some common techniques, which can be adapted to prove other relations than range proofs. The first one is that a verifier is convinced with high probability that a vector is zero $\vec{a} = \vec{0}^{|\vec{a}|}$, if the inner product of $\vec{a}$ with a random vector of verifier provided values $\vec{y}$ of same length is zero: $\langle \vec{a}, \vec{y} \rangle = 0$. Instead of using $|\vec{y}|$ different challenge variables, a smaller communication size is achieved by only using one scalar y. If the inner product $\langle \vec{a}, \vec{y}^{|\vec{a}|} \rangle = 0$, where $\vec{y}^{|\vec{a}|}$ is a vector of consecutive powers of y, then by

the Schwarz-Zippel lemma, the resulting polynomial in y must have all zero coefficients.
With this outer protocol, constraints which are representable as inner products can be enforced. In the case of a range proof, the encoded witness $\vec{a}$ is the binary decomposition of a value v and the constraint is a vector of powers of 2. By showing that $\langle \vec{a}, \vec{2}^\beta \rangle = v$ and $\vec{a}$ is binary, i.e. $\vec{a} \in \{0,1\}^\beta$, the verifier is convinced that $v \in \{0, \dots, 2^\beta - 1\}$. To show that a vector is binary and only consists of bits is a common scheme and achieved by creating an auxiliary vector $\vec{b} = \vec{a} - \vec{1}^\beta$. $\vec{b}$ is 0 wherever $\vec{a}$ is 1 and the equivalent of -1 in the finite field wherever $\vec{a}$ is 0. By showing that $\vec{a} \circ \vec{b} = \vec{0}^\beta$, the verifier is assured that for each position, either $\vec{a}$ is zero or $\vec{b}$ is zero. Given a challenge variable y from the verifier, the prover assures that $\langle \vec{a}, \vec{b} \circ \vec{y}^\beta \rangle = 0$. With a third constraint, assuring that $\vec{b} = \vec{a} - \vec{1}^\beta$, all non-zero positions of $\vec{a}$ must be 1. The difference is shown as inner product $\langle \vec{a} - \vec{1}^\beta - \vec{b}, \vec{y}^\beta \rangle = 0$.
Multiple inner product constraints are combined by creating a polynomial in another verifier provided challenge variable z which is only zero at a random evaluation, if all coefficients are zero. This construction allows proving complex constraints on values. Most importantly, Bulletproofs allow the prover to provide the values in commitments. The value v in the above range prove example is available to the verifier only as commitment. This is sufficient to enable range proofs for existing transaction systems such as Monero.

2.9.1 *Extended Bulletproofs*

Bulletproofs connect the inner product constraint witness to committed values. This requires that the prover knows an opening to all commitments provided. For operations on anonymity sets, it is required to prove a property about a single value in one commitment, while using the remaining commitments as anonymity set [LRR+19].
Omniring proposed an extended outer protocol around the same efficient inner product protocol. The new outer protocol allows the prover to prove arbitrary discrete logarithm relations for public group elements. It works on groups where the discrete logarithm assumption holds. Instead of only proving that $\langle \vec{a}, \vec{b} \rangle = 0$ with $a, b \in {\mathbb{Z}_q}^m$, the extended protocol attaches a different constraint, namely for vector of public elements $\Lambda \in \mathbb{G}^n$ with $n \leq m$, it also holds that $\Lambda^{\circ a[:n]} = I$ is the identity. The combination of the group identity and the inner product constraints makes the extended Bulletproof protocol a versatile proof system with logarithmic transcript size.
There is recent work on providing efficient proof systems for arbitrary arithmetic circuit in a pairing based setting by Lai et al. [LMR19] and Attema et al. [ACR20]. One option is to specify the arithmetic circuit required for our relation. However by not using the protocol as a black-box, we improve the communication complexity.
The Omniring outer protocol uses challenge variables u, v, y, z sent by the verifier to compress multiple constraints into a single inner product relation. Each constraint is separated by a unique power of the challenge variables u, v, y, z

and if the resulting polynomial has as many roots as the maximum power, all constraints hold with overwhelming probability due to the Schwarz-Zippel lemma. Each constraint consists of a vector $\vec{v}_i$ of the same length as the encoded witnesses $\vec{c}_L$ and $\vec{c}_R$ and may be dependent on the challenge variables u, v, y. A constraint enforces a property on the witness by requiring an inner product relation of the witness and the constraint to result in a publicly computable scalar $\hat{v}_i$. We support four different relation classes between witness and constraint vector named by $\mathfrak{mul}, \mathfrak{dir}, \mathfrak{sum}, \mathfrak{one}$. I.e. if a constraint has the class $\mathfrak{mul}$, written as $\mathsf{cls}(i) = \mathfrak{mul}$, the relation $\langle \vec{c}_L, \vec{c}_R \circ \vec{v}_i \rangle = \hat{v}_i$ holds. A combination of these constraints allow a prover to convince the verifier that the witness has a specific structure. E.g. it is possible to assure that the first value of $\vec{c}_L$ is 5 larger than the value at position 2 ($\vec{c}_L = (x+5, x, \dots)$. Therefore create a constraint vector $\vec{v}_i$ of the class $\mathfrak{dir}$ which has the values $\vec{v}_i = (y, -y, 0, \dots)$ at the same first positions. As the constraint is $\mathfrak{dir}$, $\langle \vec{c}_L, \vec{v}_i \rangle = \hat{v}_i$ must hold and if $\hat{v}_i = 5y$, the verifier is convinced about the correct encoding of the witness. The Omniring protocol between prover $\mathcal{P}$ and verifier $\mathcal{V}$ with $m = |\vec{c}_L|$ and $n = |\Lambda(\dots)|$ proceeds as follows:

$\mathcal{V}$: $v, u \xleftarrow{\$} \mathbb{Z}_q, D \xleftarrow{\$} \mathbb{G}, \vec{P} \xleftarrow{\$} \mathbb{G}^n, \vec{G}' \xleftarrow{\$} \mathbb{G}^{m-n}, \vec{H} \xleftarrow{\$} \mathbb{G}^m$

$\mathcal{P} \leftarrow \mathcal{V}$: $v, u, D, \vec{P}, \vec{G}', \vec{H}$

$\mathcal{P}, \mathcal{V}$: For $w \in \mathbb{Z}_q$ define $\vec{G}_w := (\Lambda(\vec{K}, u, v)^{\circ w} \circ \vec{P} \| \vec{G}')$

$\mathcal{P}$: $r_A \xleftarrow{\$} \mathbb{Z}_q, A := D^{r_A} \vec{G}_0^{\vec{c}_L} \vec{H}^{\vec{c}_R}$

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

$\mathcal{V}$: $w \xleftarrow{\$} \mathbb{Z}_q$ and

$\mathcal{P} \leftarrow \mathcal{V}$: w

$\mathcal{P}$: 1. $\vec{s}_L \xleftarrow{\$} {\mathbb{Z}_q}^m, \vec{s}_R = (\forall i \in [m] : \begin{cases} 0 \text{ if } \vec{c}_R[i] = 0 \\ s \xleftarrow{\$} \mathbb{Z}_q \text{ else} \end{cases})$

2. $r_S \xleftarrow{\$} \mathbb{Z}_q, S := D^{r_S} \vec{G}_w^{\vec{s}_L} \vec{H}^{\vec{s}_R}$

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

$\mathcal{V}$: $y, z \xleftarrow{\$} \mathbb{Z}_q$

$\mathcal{P} \leftarrow \mathcal{V}$: y, z

Now the prover and the verifier compress the constraints of the parametrization. Each constraint $\vec{v}_i$ gets a unique index i and $\mathsf{cls}(\vec{v}_i)$ returns the class $\{\mathfrak{mul}, \mathfrak{dir}, \mathfrak{sum}, \mathfrak{one}\}$ of the constraint. Define $\delta := \langle \vec{\alpha}, \vec{\mu} \rangle + \langle \vec{1}^m, \vec{\nu} \rangle + \sum_i z^i \hat{v}_i$ and

$$\vec{\Theta} := \sum_{i:\mathsf{cls}(\vec{v}_i)=\mathfrak{mul}} z^i \vec{v}_i \qquad \vec{\mu} := \sum_{i:\mathsf{cls}(\vec{v}_i)\neq\mathfrak{mul}} z^i \vec{v}_i \qquad \vec{\nu} := \sum_{i:\mathsf{cls}(\vec{v}_i)=\mathfrak{one}} z^i \vec{v}_i$$
$$\vec{\omega} := \sum_{i:\vec{v}'_i \neq 0} z^i \vec{v}'_i \qquad \vec{\alpha} := \vec{\Theta}^{\circ -1} \circ (\vec{\omega} - \vec{\nu}) \qquad \vec{\beta} := \vec{\Theta}^{\circ -1} \circ \mu$$

$\mathcal{P}$: Define polynomials in X:

- $l(X) := \vec{c}_L + \vec{\alpha} + \vec{s}_L \cdot X$ and
- $r(X) := \vec{\Theta} \cdot (\vec{c}_R + \vec{s}_R \cdot X) + \vec{\mu}$ with
- $t(X) := \langle l(X), r(X) \rangle = \delta + t_1 X + t_2 X^2$

for some t_1 and t_2, let $\tau_1, \tau_2 \xleftarrow{\$} \mathbb{Z}_q, T_1 := G^{t_1} D^{\tau_1}, T_2 := G^{t_2} D^{\tau_2}$

$\mathcal{P} \to \mathcal{V}$: T_1, T_2

$\mathcal{V}$: $x \xleftarrow{\$} \mathbb{Z}_q, Q \xleftarrow{\$} \mathbb{G}$

$\mathcal{P} \leftarrow \mathcal{V}$ x, Q

$\mathcal{P}$:

1. $\tau := \tau_1 x + \tau_2 x^2$
2. $r := r_A + r_S x$
3. $(\vec{l}, \vec{r}, t) := (l(x), r(x), t(x))$
4. padd $\vec{l}$ and $\vec{r}$ with 0 to length $2^{\lceil \log_2(m) \rceil}$
5. $\pi_{\mathsf{IP}} \leftarrow \textbf{IPprove}(\vec{l}, \vec{r}, \vec{G}_w, \vec{H}^{\vec{\Theta}^{\circ -1}}, Q)$

$\mathcal{P} \to \mathcal{V}$: $\tau, r, \pi_{\mathsf{IP}}, t$

$\mathcal{V}$: verify $P = AS^x \vec{G}_w^{\vec{\alpha}} \vec{H}^{\vec{\beta}} \cdot D^{-r} \cdot Q^t$
and verify $\textbf{IPvf}(\pi_{\mathsf{IP}}, \vec{G}_w, \vec{H}^{\vec{\Theta}^{\circ -1}}, P, Q) = 1 \wedge G^t D^\tau = G^\delta T_1^x T_2^{x^2}$.

The inner product protocol (**IPprove**, **IPvf**) is equal to Bulletproofs which satisfies the language $\left((P, Q, G, H) \in \mathbb{G} : \exists \vec{l}, \vec{r} \in {\mathbb{Z}_q}^m \text{ s.t. } P = \vec{G}^{\circ \vec{l}} \vec{H}^{\circ \vec{r}} Q^{\langle \vec{l}, \vec{r} \rangle} \right)$.

2.10 UTXO TRANSACTION SYSTEMS

The common notation on signatures and arguments of knowledge allows us to describe the core concepts of existing transaction systems without distracting technical details. The description of the system is independent on if the system is privacy-preserving or not. It merely provides the basics to then define the security and privacy properties to achieve a privacy-preserving system.
UTXO transaction systems are one of the simplest form to keep track of balances. Similar to a bank ledger where all incoming and outgoing transactions are registered, an UTXO system redistributes tokens between accounts. They share the invariant that the total amount in circulation stays constant. Whatever is added to one account must be deducted from another. If we capture a snapshot of the current ownership as a `state`, a transaction `tx` represents a valid update to a new state. Depending on the the environment of the system, the state may be updated by outside actors. This is the case in most blockchain systems, where new tokens are mined and added to the set of spendable outputs. This is a state change, which not necessarily adheres to the invariant and must be controlled differently, e.g. through consensus. To ensure that every participant reaches the same `state`, all transactions must be applied in the same order. In a ledger, this is achieved by a consensus mechanism.

Definition 2.9 (UTXO system)**.** *An UTXO transaction system consists of the following ppt algorithms:*
$(\mathsf{pp}, \mathsf{state}) \leftarrow \mathsf{UTXO.Setup}(1^\lambda, \vec{\mathsf{bl}})$*: takes the security parameter* λ *and a vector of initial balances* $\vec{\mathsf{bl}}$ *and outputs an initial* state *which holds accounts for each balance and public parameters* pp *implicitly available to all other algorithms.*
$(\mathsf{lts}, \mathsf{ltp}) \leftarrow \mathsf{UTXO.KeyGen}()$*: generates a long term key pair* $(\mathsf{lts}, \mathsf{ltp})$ *to participate in the system.*
$(\mathsf{acc}, \mathsf{ck}) \leftarrow \mathsf{UTXO.OTGen}(\mathsf{ltp}, v)$*: takes a long term public key* ltp *and an amount* $a \in \mathcal{M}$ *and outputs a one-time account* acc *along with secrets subsumed as coin key* ck*.*
$\mathsf{tx} \leftarrow \mathsf{UTXO.Spend}(\mathcal{S}, \mathcal{T}, \mathsf{state})$*: takes a set of inputs* $\mathcal{S} := \{\mathsf{acc}_i^{\mathcal{S}}, \mathsf{sk}_i, a_i^{\mathcal{S}}\}_{i=1}^{|\mathcal{S}|}$ *with* acc_i^S *belonging to secret keys* sk_i *and amount* $a_i^{\mathcal{S}}$*. The second parameter is a set of outputs* $\mathcal{T} = \{\mathsf{pk}_i, a_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}$ *for a long term key* pk_i *and an amount* $a_i^{\mathcal{T}}$*. The spend algorithm may use the current* state *and then outputs a transaction* tx*.*
$\mathsf{state}'/\bot \leftarrow \mathsf{UTXO.Verify}(\mathsf{state}, \mathsf{tx})$*: takes a current* state *and a transaction* tx *and outputs either a new state* state' *or fails with* $\bot$*.*
$(\mathsf{sk}, a)/\bot \leftarrow \mathsf{UTXO.Receive}(\mathsf{lts}, \mathsf{acc})$*: takes a long term secret key* lts *and an one-time account* acc *and if the account belongs to the owner of* lts*, it outputs the amount stored a and a one-time secret key* sk*.*

The transaction system is correct if honestly generated one-time accounts, known as transaction outputs, are correctly received, i.e., the amount is equal. Any honestly generated transaction which uses inputs present in the state have an equal sum of input and output amounts, and the secret keys sk_i are correct for the inputs, is valid. Formally it is defined as follows:

Definition 2.10 (Correctness)**.** *An UTXO system is correct if all the following statements hold:*

- *For all* $\lambda \in \mathbb{N}$ *with* $\mathsf{pp} \leftarrow \mathsf{UTXO.Setup}(1^\lambda, \emptyset)$ *and all* $(\mathsf{lts}, \mathsf{ltp}) \leftarrow \mathsf{UTXO.KeyGen}()$ *it holds that for any* $a \in \mathcal{M}$*, generate* $(\mathsf{acc}, \mathsf{ck}) \leftarrow \mathsf{UTXO.OTGen}(\mathsf{pk}, a)$ *and receive the same amount with* $\mathsf{UTXO.Receive}(\mathsf{sk}, \mathsf{acc}) = (\mathsf{sk}, a)$*.*
- *All honestly created transaction are valid: For all* state*,* $\mathcal{S}$ *and* $\mathcal{T}$ *with the structure from above satisfying*
  - $\forall i \in [|\mathcal{S}|] : \mathsf{acc}_i^{\mathcal{S}} \in \mathsf{state.UTXO}$*, where* state.UTXO *is the set of all unspent outputs.*
  - $\forall i \in [|\mathcal{S}|] : \mathsf{UTXO.Receive}(\mathsf{lts}_i, \mathsf{acc}_i^{\mathcal{S}}) = (a_i^{\mathcal{S}}, \mathsf{sk}_i)$
  - $\sum_{i=1}^{|\mathcal{S}|} a_i^{\mathcal{S}} = \sum_{i=1}^{|\mathcal{T}|} a_i^{\mathcal{T}}$

  it holds that $\mathsf{UTXO.Verify}(\mathsf{state}, \mathsf{UTXO.Spend}(\mathcal{S}, \mathcal{T}, \mathsf{state})) = \mathsf{state}'$ *and* $\mathsf{state}' \neq \bot$ *with* $\mathsf{state}'.\mathsf{UTXO}$ *containing the new accounts of* $\mathcal{T}$ *and having all accounts* $\{\mathsf{acc}_i^{\mathcal{S}}\}_{i=1}^{|\mathcal{S}|}$ *removed.*

All UTXO systems share a common security property which reflects the intuitive notion that the system prevents thefts of any kind. Theft can either occur from another participant or from thin air. How this property is actually achieved depends on the specific system. We will present one possibility, as used in Monero like systems, which is based on Tags. Every transaction input gets a Tag assigned. A new transaction is valid if it only uses inputs with new tags. Thereby reusing the same transaction output twice is prevented as it could be detected. Every one-time account is used exactly once, theft is prevented, if the outputs of a transaction spend exactly as much as the inputs consume. Then the transaction is balanced.
Regarding privacy, a transaction may provide sender and receiver anonymity and value confidentiality. More formal definitions are available for concrete systems, such as Zcash [SCG+14] or RingCT [LRR+19]. In Zcash, a transaction achieves perfect sender anonymity, while RingCT systems build upon smaller anonymity sets in a trade-off for transparency and efficiency.

2.11 RINGCT

Due to the requirement for a trusted setup in Zcash like transaction systems, we focus our further investigations on transaction systems with transparent setups. A trusted setup allows for efficient constant size proofs but to set up the system, a common reference string (CRS) with a trapdoor is needed to which no one must know the secret. Generating such a CRS in a setting, where nobody can agree on a trusted party, this is impossible. To build a privacy-preserving UTXO chain, there are a few core concepts which enable achieving the desired properties.

- Transaction outputs, also described as one-time accounts must hide the amount and remain unlinkable to the long term account they belong to. As each transaction output is then distributed uniformly at random, the system has receiver anonymity.
- Instead of a single reference to a previous transaction output, an input is defined by a set of references to multiple previous outputs. Only one reference is actually spent, while all other references contribute to sender anonymity.
- Sender anonymity in UTXO systems introduces the issue that all participants except for the sender cannot determine if an output is spent or not. The UTXO property requires that only unspent outputs are used as inputs. To prevent using the same input twice while hiding which inputs are spent, privacy-preserving UTXO systems use a tag. Each real input has a deterministically derived, unlinkable tag. All tags are stored as a set like the outputs. If two transactions use the same tag, they intend to spend the same transaction output which is detectable. Thereby double spend detection is reduced to comparing tags. The correctness of a tag for a specific input is proven in zero knowledge.

– To achieve amount confidentiality, RingCT systems use commitments and a zero knowledge proof that the amounts are conserved in a transaction.

The formalization of these notions have been published by the authors of the Omniring system [LRR+19], which is the first to rigorously formalize the RingCT transaction system. The popular system Monero operates with the same principles, albeit with outdated NIZK protocols.

2.11.1 *Tagging Scheme*

The tagging scheme of a privacy-preserving UTXO system defines the relation between a secret key, belonging to a transaction output, and a tag which marks the output as spent. Intuitively, the tag must not be linkable to the public key. A transaction creator is then able to publish the tags belonging to each real input. The corresponding referenced outputs are then marked as spent and each subsequent transaction which tries to spend the same output is detected by having the same tag.
The tagging scheme consists of the algorithms $\mathsf{TAG} = (\mathsf{TagSetup}, \mathsf{TagKGen}, \mathsf{TagEval})$. It uses a secret key space $(\chi, +)$, a public key space $(\mathcal{X}, \cdot)$ and a tag space ψ. $\mathsf{TagKGen}$ is homomorphic, i.e. for any $x, x' \in \chi$, $\mathsf{TagKGen}(x) \cdot \mathsf{TagKGen}(x') = \mathsf{TagKGen}(x + x')$. $\mathsf{TagEval}$ takes a key $x \in \chi$ and outputs a $\mathsf{tag} \in \mathcal{X}$. We require related-input one-wayness and pseudorandomness.
The one-wayness of the scheme assures that given a tag and a partial secret key in the form of a homomorphic part of the key, it is infeasible to recover the full secret key. This is required as otherwise the tag key relation can be detected by the creator of the one-time account. The pseudorandomness makes honestly generated keys indistinguishable from randomly drawn elements in the tag space ψ.
Tho examples of tagging systems are e.g.

OMNIRING: $\mathsf{TagKGen}(x) = H^x$ and $\mathsf{TagEval}(x) = G^{\frac{1}{x}}$ with $G, H \in \mathbb{G}$

MONERO: $\mathsf{TagKGen}(x) = H^x$ and $\mathsf{TagEval}(x) = \mathsf{H}(H^x)^x$ with $H \in \mathbb{G}$ and a random oracle $\mathsf{H} : \mathbb{G} \to \mathbb{G}$. The Monero tagging scheme is not homomorphic.

They both satisfy one-wayness and pseudorandomness as proven by the Omniring paper.

2.11.2 *Ring Sampling*

The main difference between Zcash and RingCT based transaction system is the size of the anonymity set for input references. As the most efficient proof systems with transparent setup have linear verification time, RingCT based systems have to heuristically select the anonymity set as a subset of all previous transaction outputs. Which subset is selected greatly influences the possibility to analyze the transaction graph and deanonymize senders.

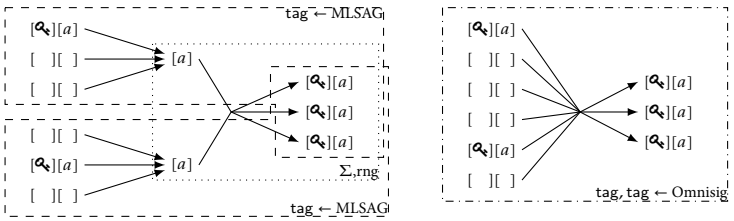
One approach is Binned Mixin Sampling [MSH+18], which samples temporally local groups of previous outputs to counter timing attacks and protect against an adversary who controls many outputs. This means that for some chunks of history, we achieve perfect anonymity and the more of these chunks we include in the set, the better the anonymity.
Anonymity set sampling, also called ring sampling, is outside the scope of this work and we refer to Ronge et al. [REL+20] for in depth research on this topic.

2.11.3 *Monero*

Combining the privacy-preserving methods explained before, Monero is the most prominent system using RingCT. Previous versions of Monero used a different transaction system, but the current structure of Monero transactions is called RingCT-simple. As shown in Figure 2.3, such a transaction consists of multiple parts:

- For each input, the sender samples an anonymity set which hides exactly one real input. In our example, each real input is hidden in a set of size 3.
- The amount of the one true input is stored in the commitment as part of the referenced output. This exact same amount is committed again for each input, but with new randomness.
- Each output of a transaction is composed of a one-time derived key from the tagging scheme and a commitment to an amount.
- Monero then uses one signature of knowledge for each input. The condition is that the signer knows the secret key of the real input, the amount in the newly created commitment is equal to the real input and the tag is correct according to the tagging function. The message of the SoK is the whole transaction and importantly the full set of outputs. As the concrete instantiation, Monero uses a Multi-Layer Linkable Spontaneous Ad-hoc Group Signature (MLSAG).
- For the balance of a transaction, it remains to show that the amounts of inputs and outputs are balanced. The homomorphic property of the commitments allows Monero to use a basic Σ-protocol AoK (Section 2.6.1) to show the balance. To prevent negative outputs, each output amount must be in a positive range, which is asserted by a Bulletproof range proof.

Monero demonstrated the practicability and real world application of a RingCT based system. However is has multiple disadvantages. The MLSAG construction signature has a linear size dependent on the size of the anonymity set, financially incentivizing users to rely on a small anonymity set. The current default is 10 decoy inputs for each real one. Safely upgrading Monero to more efficient signatures is a lengthy process and requires careful



(a) Monero RCT-simple (b) Omniring

Figure 2.3: Transaction structures for (a) Monero RCT-simple and (b) Omniring transactions. a represent arbitrary amounts. Dashed lines are signatures to authorize spending and dotted lines ensure token conservation. $[\cdot]$ denotes a commitment to the value and if not blank, the value is known to the signer. [EMP+21]

testing before deploying any change to the live system. Without these limitations, a clean slate approach to RingCT can be made more efficient. This is exactly what Omniring provides.

2.11.4 *Omniring*

The Omniring [LRR+19] authors narrowed down the essential parts of a RingCT transaction and provided a rigorous formalization of the security and anonymity such a system provides. A RingCT system needs to anonymously select a set of real inputs from an anonymity set and show that the outputs are balanced with regard to the inputs. Figure 2.3b shows an overview of the Omniring transaction structure. Instead of multiple different signatures in Monero, Omniring encompasses all constraints on the system into a single SoK. The SoK checks that

- for each used input in the reference set, the sender knows a secret key
- the tags included in the transaction correspond to the used input keys
- and the sum of amounts in the real inputs is equal to the sum of output amounts.

They build an efficient SoK by extending the Bulletproof protocol.
As we adapt the Omniring formalization to formalize our transaction systems, we provide the original definitions to facilitate the comparison.
Compared to the generic UTXO definition 2.9, Omniring uses an explicit ring $\mathcal{R}$ to capture the anonymity set. This $\mathcal{R}$ is the subset of previous outputs which are possible inputs to an Omniring transaction. The beauty of this definition is the fact that it captures not only RingCT but also Zcash like systems. In Zcash, $\mathcal{R}$ is the complete set of previous transaction outputs. The second change is an explicit definition of a tag, which is used to mark outputs as used.

Definition 2.11. *A RingCT scheme consists of a tuple of PPT algorithms* (Setup, KeyGen, OTGen, Spend, VfTx, Receive) *defined as follows:*

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\alpha, 1^\beta)$ *takes the security parameter* λ *and integers* α *for a maximum of* 2^α *outputs of a transaction where each has an amount maximum of* $2^\beta - 1$. *Then it outputs public parameters* pp *which are implicitly given to all the following algorithms. Setup is called once when a RingCT system is initialized.*

$(\mathsf{ltp}, \mathsf{lts}) \leftarrow \mathsf{KeyGen}()$ *generates a long term secret key* lts *with the corresponding long term public key* ltp *for participants to initially join the system. The* ltp *is distributed and serves as a recipient address.*

$\mathsf{acc}, \mathsf{ck} \leftarrow \mathsf{OTGen}(\mathsf{ltp}, a)$ *creates a one-time account* acc *with coin key* ck *from a long term public key* ltp *and an amount* a *to then use this account as an output in a transaction.*

$\mathsf{t} \leftarrow \mathsf{Spend}(\mathcal{S}, \mathcal{R}, \mathcal{T})$ *takes the inputs*

- $\mathcal{S} = \{(\mathsf{tag}_i, j_i, \mathsf{sk}_i, a_i^{\mathcal{S}}, \mathsf{ck}_i^{\mathcal{S}})\}_{i=1}^{|\mathcal{S}|}$ *is a set of inputs with a* tag_i *corresponding to* $\mathsf{acc}_{j_i}^{\mathcal{R}}$ *at index* $j_i \in [|\mathcal{R}|]$, *secret key* sk_i, *amount* $a_i^{\mathcal{S}}$ *with coin key* $\mathsf{ck}_i^{\mathcal{S}}$.
- $\mathcal{R} = \{\mathsf{acc}_i^{\mathcal{R}}\}_{i=1}^{|\mathcal{R}|}$ *is a set of ring accounts to hide the real inputs.*
- $\mathcal{T} = \{(\mathsf{acc}_i^{\mathcal{T}}, a_i^{\mathcal{T}}, \mathsf{ck}_i^{\mathcal{T}})\}_{i=1}^{|\mathcal{T}|}$ *is a set of accounts* $\mathsf{acc}_i^{\mathcal{T}}$ *with amount* $a_i^{\mathcal{T}}$ *and coin key* $\mathsf{ck}_i^{\mathcal{T}}$.

It outputs a signature t *as authorization to spend the inputs to the designates outputs.*

$b \leftarrow \mathsf{VfTx}(\mathsf{tx}, \mathsf{t})$ *takes a transaction defined as*

$$\mathsf{tx}(\mathcal{S}, \mathcal{R}, \mathcal{T}) := \left(\{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{acc}_i^{\mathcal{R}}\}_{i=1}^{|\mathcal{R}|}, \{\mathsf{acc}_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}\right)$$

and the signature t *and returns a bit* b *representing the validity.*

$(\mathsf{tag}, \mathsf{sk}, a, \mathsf{ck}) \leftarrow \mathsf{Receive}(\mathsf{acc}, \mathsf{lts})$ *gets an account* acc *and a long term secret* lts *and returns the matching* tag, *secret key* sk, *amount* a *and coin key* ck *for* acc *if* lts *owns the account.*

Further, we require the following two auxiliary algorithms to define the security properties.

$b \leftarrow \mathsf{ChkAcc}(\mathsf{acc}, a, \mathsf{ck})$ *takes an account* acc, *an amount* a *and a coin key* ck *and checks if they are consistent.*

$b \leftarrow \mathsf{ChkTag}(\mathsf{acc}, \mathsf{tag}, \mathsf{sk})$ *takes an account* acc, *a* tag *and a secret key* sk *and returns* 1 *if consistent,* o *otherwise.*

Definition 2.12 (Correctness)**.** *A RingCT scheme is correct, if for all* $\lambda, \alpha, \beta \in \mathbb{N}$ *and all* $\mathsf{pp} \in \mathsf{Setup}(1^\lambda, 1^\alpha, 1^\beta)$*:*

- *Honestly generated payments are received correctly:*
  For any $\mathsf{ltp}, \mathsf{lts} \in \mathsf{KeyGen}()$, *any amount* $a \in \{0, \dots, 2^\beta - 1\}$, *any* $(\mathsf{acc}, \mathsf{ck}) \in \mathsf{OTGen}(\mathsf{ltp}, a)$, *and any* $(\cdot, a', \mathsf{ck}') \in \mathsf{Receive}(\mathsf{acc}, \mathsf{lts})$, *it holds that* $(a, \mathsf{ck}) = (a', \mathsf{ck}')$.

- *Honestly received payments have a valid amount and tag: For any* $(\mathsf{tag}, \mathsf{sk}, a, \mathsf{ck}) \in \mathsf{Receive}(\mathsf{acc}, \mathsf{lts})$, $\mathsf{ChkAcc}(\mathsf{acc}, a, \mathsf{ck}) = 1$ *and* $\mathsf{ChkTag}(\mathsf{acc}, \mathsf{tag}, \mathsf{sk}) = 1$ *holds.*

- *Honestly generated transactions are valid: For each* $\mathcal{S}, \mathcal{R}, \mathcal{T}$, *defined as above, that satisfy*
  - $\forall i \in [|\mathcal{S}|], \mathsf{ChkTag}(\mathsf{acc}^{\mathcal{R}}_{j_i}, \mathsf{tag}_i, \mathsf{sk}_i) = 1$
  - $\forall i \in [|\mathcal{S}|], \mathsf{ChkAcc}(\mathsf{acc}^{\mathcal{R}}_{j_i}, a^{\mathcal{S}}_i, \mathsf{ck}^{\mathcal{S}}_i) = 1$
  - $\forall i \in [|\mathcal{T}|], \mathsf{ChkAcc}(\mathsf{acc}^{\mathcal{T}}_i, a^{\mathcal{T}}_i, \mathsf{ck}^{\mathcal{T}}_i) = 1$
  - $\sum\{a^{\mathcal{S}}_i\}_{i=1}^{|\mathcal{S}|} = \sum\{a^{\mathcal{T}}_i\}_{i=1}^{|\mathcal{T}|}$

  and for any signature $\mathfrak{t} \in \mathsf{Spend}(\mathcal{S}, \mathcal{R}, \mathcal{T})$, *it holds that* $\mathsf{VfTx}(\mathsf{tx}, \mathfrak{t}) = 1$ *with* $\mathsf{tx} = \mathsf{tx}(\mathcal{S}, \mathcal{R}, \mathcal{T})$.

Given this definition of a RingCT scheme, Omniring provides the security and anonymity properties which match the intuitive understanding of Monero's security.
Most importantly, Omniring specifies a balance property. An adversary must not be able to create a transaction which is valid but spends more amount in the outputs than it consumes. The privacy of the system requires a more concrete specification of sender and receiver anonymity. The goal is that any subset of $|\mathcal{S}|$ inputs from the total set $\mathcal{R}$ has equal probability. Regarding recipient anonymity, only the owner of the long term secret key lts is able to detect, link, and receive an output. Both properties are captured by a security experiment, where an adversary presents two partial transactions with two versions of instructions on how to complete the transaction given access to honest accounts. The experiment creates one version and the adversary wins if they are able to correctly guess which instructions were executed. This provides the adversary with the freedom to attack the anonymity on the sender and receiver side, even in the case where a transaction uses corrupted accounts as decoy inputs.
For the formal security and privacy experiments, we refer to the original Omniring paper [LRR+19].

2.12 ASSETS

Previous work established a well formalized and proven secure UTXO transaction system. As the goal of this thesis is to extend it to handle multiple asset types confidentially, we provide an overview of existing work on confidential tokens. An asset type or simply a type in this thesis is a property of an amount which may be read as currency. It is equivalent to color in our preliminary work [EKB18] or colored Bitcoin.

The most important factor to a confidential transaction system is the storage of values. While previous UTXO systems used Pedersen commitments to store amounts without a notion of currency or type, we require typed commitments. Such commitments store an additional type, indicating the currency the amount is in.
We provide the generalized formalization of the definitions from Confidential Assets [PBF+17]:

Definition 2.13 (Typed Commitment Scheme)**.** *A type aware commitment scheme* TC *consists of the PPT algorithms* $\mathsf{pp} \leftarrow \mathsf{TC.Setup}(1^\lambda)$ *which takes the security parameter* λ *and outputs public parameters* pp *implicitly given to* $\mathsf{com} \leftarrow \mathsf{TC.Commit}(\mathsf{ty}, a; r)$ *which takes a type* $\mathsf{ty} \in \mathbb{T}$*, an amount* $a \in \mathbb{M}$ *and randomness* $r \in \mathbb{S}$ *and outputs a commitment* com*.*

The commitment has to satisfy the binding and hiding properties similar to regular commitments.

Definition 2.14 (Hiding)**.** *A typed commitment scheme* TC *is hiding if for any adversary* $\mathcal{A}$*, any* $\mathsf{ty} \in \mathbb{T}$*, any amount* $a \in \mathbb{M}$ *it holds that*

$$\Pr\left[\begin{array}{r} b' \leftarrow \mathcal{A}(\mathsf{com}) \\ b' = b \end{array} : \begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), b \xleftarrow{\$} \{0,1\}, r \xleftarrow{\$} \mathbb{S} \\ \mathsf{ty}_0, a_0, \mathsf{ty}_1, a_1 \leftarrow \mathcal{A}(\mathsf{pp}) \\ \mathsf{com} \leftarrow \mathsf{TC.Commit}(\mathsf{ty}_b, a_b; r) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Definition 2.15 (Binding)**.** *A typed commitment scheme* TC *is binding if for any adversary* $\mathcal{A}$*, any* $\mathsf{ty} \in \mathbb{T}$*, any amount* $a \in \mathbb{M}$ *it holds that*

$$\Pr\left[\begin{array}{r} \mathsf{com}_0 \leftarrow \mathsf{TC.Commit}(\mathsf{ty}_0, a_0; r_0) \\ \mathsf{com}_1 \leftarrow \mathsf{TC.Commit}(\mathsf{ty}_1, a_1; r_1) \\ \mathsf{com}_0 = \mathsf{com}_1 \\ \wedge (\mathsf{ty}_0, a_0) \neq (\mathsf{ty}_1, a_1) \end{array} : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ \\ \mathsf{ty}_0, a_0, r_0 \\ \mathsf{ty}_1, a_1, r_1 \end{array} \leftarrow \mathcal{A}(\mathsf{pp})\right] \leq \mathsf{negl}(\lambda)$$

One crucial feature of some commitments is their homomorphic property. That means that two commitments can be combined to form a commitment to some reasonable operation of the committed values. In Pedersen commitments, the homomorphic operation results in the addition of the values. I.e. $(G^a H^r) \cdot (G^b H^s) = G^{a+b} H^{r+s}$. With commitment to multiple values, the homomorphism can have multiple variants. In the following two definitions, we specify how the homomorphic property applies to the committed values. In the first case, the commitments are additively homomorphic in each domain, i.e. the amounts are reasonably added in $\mathbb{M}$, but the types in $\mathbb{T}$ are added, which is undefined or results in a totally unrelated type. Therefore the homomorphic addition of commitments does not represented the intuitive addition of the amounts in each type. Instead of a homomorphic operation, commitments are accompanied by a NIZK that their openings adhere to the constraints.

Definition 2.16 (Homomorphic Property)**.** *If typed commitments* TC *satisfy the additional property that for all* $\mathsf{ty}_0, \mathsf{ty}_1 \in \mathbb{T}$ *and all* $a_0, a_1 \in \mathbb{M}$ *it holds*

that $\mathsf{TC.Commit}(\mathsf{ty}_0, a_0; r_0) \odot \mathsf{TC.Commit}(\mathsf{ty}_1, a_1; r_1) = \mathsf{TC.Commit}(\mathsf{ty}_0 + \mathsf{ty}_1, a_0 + a_1; r)$ *with an efficiently computable* r*, they have a homomorphic property.*

A succinct homomorphic TC scheme can be instantiated by a vector Pedersen commitment. Let $\mathcal{G} = (\mathbb{G}, q, G, F, H)$ be a group of order q with generators G, F, H where the discrete logarithm assumption holds. A commitment to the tuple $(\mathsf{ty}, a) \in {\mathbb{Z}_q}^2$ and randomness $r \in \mathbb{Z}_q$ is defined as $\mathsf{TC.Commit}(\mathsf{ty}, a) := G^a F^{\mathsf{ty}} H^r$.
A different homomorphic property is what we define as typed homomorphic property. There commitments are additively homomorphic in respect to the amounts if the types match. Combined commitments of unequal types should only have a valid opening if the sum of amounts in each type is 0. This is useful to show that for any types available in a conservation, the amounts all sum up to 0.

Definition 2.17 (Typed Homomorphic Commitments)**.** *Typed homomorphic commitments* THC *are typed commitments* TC *which satisfy the additional property that within the same type if for all* $\mathsf{ty} \in \mathbb{T}$ *and all* $a_0, a_1 \in \mathbb{M}$ *it holds that*

$$\begin{aligned}\mathsf{THC.Commit}(\mathsf{ty}, a_0; r_0) \odot \mathsf{THC.Commit}(\mathsf{ty}, a_1; r_1)\\ = \mathsf{THC.Commit}(\mathsf{ty}, a_0 + a_1; r)\end{aligned}$$

with an efficiently computable r*.*

As described by Poelstra et al. [PBF+17], THC can be constructed with $\mathbb{T} = \{0,1\}^*$ from a random oracle $\mathsf{H} : \{0,1\}^* \to \mathbb{G}$ as $\mathsf{THC.Commit}(\mathsf{ty}, a) := (C, V)$ with $C = \mathsf{H}(\mathsf{ty}) H^r$ and $V = C^a H^s$ with randomness r and s. For a simplified notation, we explicitly add an algorithm $\mathsf{ComTypeGen}(n \in \{0,1\}^*) := \mathsf{H}(n)$.
THC are updatable such that they commit to the same type but with new randomness.

Theorem 2.1 (Updatable)**.** *For any* $\mathsf{ty}, \mathsf{ty}' \in \mathbb{T}$ *and any* $a \in \mathbb{M}$ *with* $(C, V) = \mathsf{Commit}(\mathsf{ty}, a; (r, s))$ *and* $(C', V') = \mathsf{Commit}(\mathsf{ty}', a'; (r', s'))$ *it holds that* $\mathsf{ty} = \mathsf{ty}'$*, if there exists a PPT algorithm to compute* ϕ_1 *in* $C \cdot C'^{-1} = G^{\phi_1}$*. If additionally* ϕ_2 *in* $V \cdot V'^{-1} = G^{\phi_2}$ *is PPT computable,* $\mathsf{ty} = \mathsf{ty}'$ *and* $a = a'$ *holds.*

Given these basic commitments schemes, there are two existing approaches on proving that a transaction is balanced for each type separately.

2.12.1 *Confidential Assets*

With the type homomorphic property of the confidential assets construction [PBF+17], showing that the inputs and outputs of a transaction are balanced reduces to a sigma protocol. When homomorphically combining two balanced sets of commitments with one inverted, the value of every type is zero. If the

prover can show an opening of a commitment to zero, the balance holds. It remains to show that the newly created output commitments actually use a valid type. This is verified by an asset surjection proof. We use the updatability of typed homomorphic commitments to show that the type of each output commitment is a rerandomization of an input commitment. If this rerandomization proof provides set anonymity, the verifier does not learn which input and output pairs are of the same type.

2.12.2 *Cloaked Assets*

A different approach pursued by Stellar [And] is to use two Pedersen commitments. One for a type and one for a value. They designed an arithmetic circuit that constrains the input and output commitments in a way that only if the sum of amounts is equal in each type, the proof is accepted. They use the arithmetic circuit outer protocol of Bulletproofs to create an efficient construction.
One drawback of Confidential and Cloaked Assets is the limitation that they require two group elements to commit to a typed amount. To reduce the overall transaction size, the commitments should be as small as possible, which is a single element.

2.13 TRADING

A simple multi-type transaction system in itself only provides limited functionality over separate transaction systems each dedicated to one type. Participants need to have a mechanism to exchange tokens of different types with each other. Between different systems, there are multiple proposals summarized by Deshpande et al. [DH20] which include provable privacy guarantees. However, the meta data leakage when operating multiple systems is unavoidable. An example are two privacy-preserving type unaware transaction systems. One representing a fungible currency and the other a highly valued asset which is rarely transferred. An uptake in transactions of the rare asset serves as indicator of more volatility and might trigger financial decisions. Having both assets share an anonymity set, the transaction frequency cannot be attributed to a specific asset.
To fairly exchange tokens of different type within a single system, there exists an approach by Gao et al. [GXK+19]. They introduce an optional sibling output to each transaction output and allow to store debt in transaction outputs. The sibling output cannot be spent before their parent output and all debt attached to inputs needs to be compensated by appropriate additional inputs to cancel the debt.
An exchange then takes the following sequence of transactions shown in Figure 2.4.

1. Party A creates a transaction which has at least two special outputs. One parent note with spendable funds in type blue and a debt of red tokens. To balance the transaction, there is a sibling output (dashed) which can-

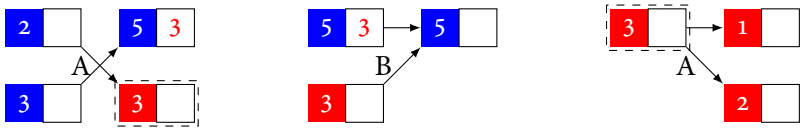


Figure 2.4: Example of an atomic swap with debt and sibling outputs.

cels the debt. As this sibling is only spendable after the parent output was spent, i.e. the debt is payed off, this does not violate the balance property.

2. Party A shares the parent note with the exchange party B who pays off the debt in red tokens and is allowed to reclaim the blue tokens of A.

3. The sibling output of A is now spendable and A has access to the red tokens.

In case party B does not cooperate, A recombines the parent and sibling output to get back the blue tokens offered for exchange.
While this exchange mechanism is fair, i.e. atomic, it requires three transaction to complete the exchange.

2.14 CONCLUSION

In a conclusion, this chapter covered the basic cryptographic tools needed for our constructions and presented existing work related to our research.
We identified the following six properties required by a privacy-preserving multi-type ledger transaction system:

SENDER AND RECEIVER ANONYMITY (SRA) ensures that senders of funds should remain hidden to all parties and that recipients are known only to the sender of a transaction. This enables basic user privacy in transactions.

CONFIDENTIAL AMOUNTS (COA) hide the amounts transferred and thereby prevent heuristic deanonymization of rational transactors.

CONFIDENTIAL TYPES (COT) support independent token types within one transaction system and hide the type to provide a single joint anonymity set over all types.

DECENTRALIZED SETUP (DSE) ensures that the security and privacy of the system does not rely on a trusted party or ceremony. This increases the trust for a truly decentralized system.

NON-INTERACTIVE TRANSACTIONS (NIT) guarantee that transactions consist of a single, non-interactive broadcast message. Any direct communication between senders and receivers decreases the sender anonymity as the receiver learns the network identity of the sender through meta-data. Additionally it reduces performance and practicability of the scheme.

Table 2.2: Overview of systems and their supported properties. Partial support of a feature is noted with (✓).

	SRA	COA	COT	DSE	NIT	DEX
MimbleWimble [FOS19]		✓	✓[ZYD+20]	✓		(✓)
Confidential Assets [PBF+17]		✓	✓	✓	✓	
Stellar [AGN+19]		✓	✓	✓	✓	
Zcash [SCG+14]	✓	✓	✓[GXK+19]		✓	(✓[GXK+19])
X-Chain Swap [DH20]	✓	✓		✓	✓	(✓)
Monero [AH18]	✓	✓		✓	✓	
Omniring [LRR+19]	✓	✓		✓	✓	

DECENTRALIZED EXCHANGES (DEX) should be used to trade tokens of different types. Trading is achieved by supporting transactions which exchange the ownership of tokens of different types. As they atomically swap the ownership of tokens, we name them *atomic swap transactions*. To satisfy the NIT property, atomic swaps need to be non-interactive and work without a trusted third party. By non-interactive we mean to restrict the sender to a single message to e.g. an exchange or a peer. This is analogous to a classical exchange, where a party submits a bid to an exchange and then forgets about it until it is fulfilled. Our non-interactivity especially excludes the offering party to actively participate in the merging of offers. For sender anonymity, swap transactions hide the number of participants resulting in the indistinguishability of simple transfers and swaps providing additional privacy by a joint anonymity set.

Table 2.2 provides an overview of our related work and presents in which properties the existing systems lack support. No existing system supports all properties in a combined way.
With these tools and systems, we will present our contributions in the following chapters. These are motivated by the shortcomings of existing solutions.

- As existing privacy-preserving transaction systems do not support a notion of type, we extend the RingCT system to accommodate multiple types. Importantly, types must remain confidential and thereby all transactions share a common anonymity set.

- To achieve space efficient Multi-type RingCT transactions, we require an efficient Multi-type conservation proof which allows the use of succinct commitments. Neither Confidential Assets nor Cloaked Assets support succinct commitments.

- For efficient, fair and anonymous trading of tokens in a Multi-type RingCT system, our goal is to develop a novel exchange mechanism such that a trade is executed in a single transaction.

I

BUILDING BLOCKS

3

3.1 OVERVIEW

In this chapter, we introduce an essential building block to aggregate signed messages and achieve anonymity of the participating signers. The work presented here was previously published in [EMP+21]. In our scenario, we have multiple parties who jointly want to sign a set of messages. In the swap application, the signers represent the authorization and the messages are the new outputs of a transaction. To sign, they use one or more keys which are unlinkable to each other and the signer's identity. Each participant contributes some messages and signs them with one or more signatures. The goal is that each participant non-interactively creates their part and then submits it to an untrusted merging party. The merger is able to aggregate all parts into a signature signed by all participants for the union of all messages. Each signer is assured that if one of their signatures is part of the aggregate, their set of messages is included in the union of messages.

A naive solution without anonymity works as follows: Each participant creates a digital signature for each key and their subset of messages. An untrusted merger aggregates them by concatenating all parts. This solution has two drawbacks. Anyone is able to deaggregate the merged signature into its original parts and may combine them differently. The possibility to split the signature and verify parts of it individually leads to an anonymity issue. The number of all possible combinations between signers and messages is small enough to brute force the mapping. We require anonymity of all participants in the final signature. I.e., the number of participants and the mapping between signatures and messages must be hidden. Therefore only the complete merged signature must be valid and any parts must not be able to be validated on their own.

As our first contribution, we build such an aggregatable signature based on carefully distributing balanced randomness. An overview is shown in Figure 3.1. For each message m_i, the signer generates a random proxy message s_i. The signers with secret keys sk_j then sign random messages x_j. The requirement

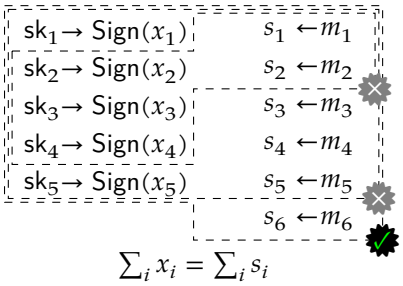


Figure 3.1: Overview of our aggregatable signature.

for a valid signature is that the sum of message proxies $\sum_i s_i$ is equal to the sum of random signed messages $\sum_j x_j$. This property is maintained when aggregating two parts of a signature. Each part has the same sum of randomness in message proxies and signed messages so the equality holds when combining the parts. For any subset of signatures and messages, the sum of randomness is not equal and thereby the signature is not valid. A verifier first checks that each signature is valid with the help of the public keys and then checks that $\sum_i x_i = \sum_i s_i$. The issue using the scheme like this, is that the partial verification is still possible. Brute force comparing the sum of all subsets of x-s with the subsets of s-s will reveal the parts which were aggregated. To prevent this attack, we use a homomorphic, hiding commitment to hide the values x and s. To allow the verifier checking the equality between x and s, they need access to the hiding factors. However it is sufficient to reveal only the sum of the hiding factors. This aggregation of the hiding factors is the irreversible operation when merging two signatures. To prevent reuse of the commitments and bind the proxies s to the actual messages, we add SoKs using the openings of the commitments as witness.
In the following sections, we provide a formalization of our signature scheme and then define its security and privacy properties. We present an efficient construction from the discrete log assumption and prove its security. As an outlook we present possible relation of our scheme to an open problem on merging signatures.

3.2 FORMALIZATION

Our formalization has little requirements on the type of signature used, so we model it with a flexible, generic Signature of Knowledge (SoK) with any NP-complete language $\mathcal{L}$ as defined in Section 2.8. The generalized problem is a set of signers, each with a set of messages $\{m_j\}_{j=1}^{|\mathcal{J}|}$ and a set of statements and witnesses $\{(\mathsf{stmt}_i, \mathsf{wit}_i)\}_{i=1}^{|\mathcal{S}|}$ for the SoK. Each of the signers create signatures $\{\mathfrak{r}_i\}_{i=1}^{|\mathcal{S}|}$ which bind the messages to the given signatures. However, linking signatures to specific messages must be infeasible. Verification must only succeed on the full set of all signatures and messages.

Definition 3.1 (Anonymous Aggregatable Signature)**.** *An anonymous aggregatable signature scheme consists of* $\mathsf{AS} = (\mathsf{AS.Setup}, \mathsf{AS.Sign}, \mathsf{AS.Verify}, \mathsf{AS.Merge})$ *parametrized with an NP Language* $\mathcal{L}$ *and the corresponding relation* $\mathbf{R}_{\mathcal{L}}$*. The algorithms are defined as follows:*

$\mathsf{pp} \leftarrow \mathsf{AS.Setup}(1^\lambda, \mathcal{L})$ *takes the security parameter* λ *and the language* $\mathcal{L}$ *which parametrizes the SoK and outputs the public parameters* pp*. The public parameters* pp *are implicit input to the subsequent algorithms.*

$(\{\mathfrak{r}_i\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a}) \leftarrow \mathsf{AS.Sign}(\{(\mathsf{stmt}_i, \mathsf{wit}_i)\}_{i=1}^{|\mathcal{S}|}, \{m_j\}_{j=1}^{|\mathcal{J}|})$ *takes a set of statement and witness tuples* $\{(\mathsf{stmt}_i, \mathsf{wit}_i)\}_{i=1}^{|\mathcal{S}|}$ *and a set of messages* $\{m_j\}_{j=1}^{|\mathcal{J}|}$ *and outputs a set of signatures* $\{\mathfrak{r}_i\}_{i=1}^{|\mathcal{S}|}$ *and a proof* $\mathfrak{a}$*.*

$\mathfrak{a} \leftarrow \mathsf{AS.Merge}(\mathfrak{a}_1, \mathfrak{a}_2)$ *takes two proofs* $\mathfrak{a}_1, \mathfrak{a}_2$ *and outputs a combined proof* $\mathfrak{a}$.

$b \leftarrow \mathsf{AS.Verify}(\{(\mathfrak{r}_i, \mathsf{stmt}_i)\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a}, \{m_j\}_{j=1}^{|\mathcal{J}|})$ *takes the signatures* $\mathfrak{r}_i$ *and statements* stmt_i*, a proof* $\mathfrak{a}$ *and the messages* $\{m_j\}_{j=1}^{|\mathcal{J}|}$ *and outputs a bit* b *depending on the validity of the signatures and the proof.*

Definition 3.2 (Correctness)**.** *The anonymous aggregatable signature scheme is correct if*

1. *Honestly signed messages are valid: For all* $(\mathsf{stmt}_i, \mathsf{wit}_i) \in \mathbf{R}_{\mathcal{L}}$ *and all messages* $\{m_j \in \{0,1\}^*\}_{j=1}^{|\mathcal{J}|}$ *with*

$$(\{\mathfrak{r}_i\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a}) \leftarrow \mathsf{AS.Sign}(\{(\mathsf{stmt}_i, \mathsf{wit}_i)\}_{i=1}^{|\mathcal{S}|}, \{m_j\}_{j=1}^{|\mathcal{J}|})$$

it holds that

$$\mathsf{AS.Verify}(\{(\mathfrak{r}_i, \mathsf{stmt}_i)\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a}, \{m_j\}_{j=1}^{|\mathcal{J}|}) = 1$$

2. *Honestly merged signatures are valid: With* $t \in \{1,2\}$*, for any* $(\mathsf{stmt}_{t,i}, \mathsf{wit}_{t,i}) \in \mathbf{R}_{\mathcal{L}}$ *and all messages* $M_t := \{m_{t,j} \in \{0,1\}^*\}_{j=1}^{|\mathcal{J}_t|}$ *with*

$$(\{\mathfrak{r}_{t,i}\}_{i=1}^{|\mathcal{S}_t|}, \mathfrak{a}_t) \leftarrow \mathsf{AS.Sign}(\{\mathsf{stmt}_{t,i}, \mathsf{wit}_{t,i}\}_{i=1}^{|\mathcal{S}_t|}, M_t)$$

and $\mathfrak{a} \leftarrow \mathsf{AS.Merge}(\mathfrak{a}_1, \mathfrak{a}_2)$ *it holds that*

$$\mathsf{AS.Verify}(\bigcup_{t\in\{1,2\}} \{(\mathfrak{r}_{t,i}, \mathsf{stmt}_{t,i})\}_{i=1}^{|\mathcal{S}_t|}, \mathfrak{a}, \bigcup_{t\in\{1,2\}} M_t) = 1$$

3.3 SECURITY

We define the security property that if a signature is included, all the intended messages must be included in the unified set of messages. More formally, if a SoK $\mathfrak{r}_i$ is part of the merged signature, then the set of messages must be a superset of the signed messages, such that all previously signed messages are included.

Definition 3.3 (Security)**.** *Given a secure SoK scheme, an anonymous aggregatable signature scheme* AS *is secure, if for all PPT adversaries* $\mathcal{A}$*, all statements and witnesses* $\{(\mathsf{stmt}_i, \mathsf{wit}_i) \in \mathbf{R}_{\mathcal{L}}\}_{i=1}^{|\mathcal{S}|}$ *and all messages* $\{m_j \in \{0,1\}^*\}_{j=1}^{|\mathcal{J}|}$ *with*

$$(\{\mathfrak{r}_i\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a}) \leftarrow \mathsf{AS.Sign}(\{(\mathsf{stmt}_i, \mathsf{wit}_i)\}_{i=1}^{|\mathcal{S}|}, \{m_j\}_{j=1}^{|\mathcal{J}|})$$

it must hold that

$$\mathsf{Pr}\left[\begin{array}{c} \{\mathfrak{r}'_i\}_{i=1}^{|\mathcal{S}'|} \cap \{\mathfrak{r}_i\}_{i=1}^{|\mathcal{S}|} \neq \emptyset, \mathcal{J}' \not\supseteq \mathcal{J} \\ \mathsf{AS.Verify}(\{(\mathfrak{r}'_i, \mathsf{stmt}'_i)\}_{i=1}^{|\mathcal{S}'|}, \mathfrak{a}', \{m'_j\}_{j=1}^{|\mathcal{J}'|}) = 1 \end{array}\middle| \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{AS.Setup}(1^\lambda) \\ (\{(\mathfrak{r}'_i, \mathsf{stmt}'_i)\}_{i=1}^{|\mathcal{S}'|}, \mathfrak{a}', \{m'_j\}_{j=1}^{|\mathcal{J}'|}) \\ \leftarrow \mathcal{A}(\{(\mathfrak{r}_i, \mathsf{stmt}_i)\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a}, \{m_j\}_{j=1}^{|\mathcal{J}|}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

In the security definition, we assume that $\mathfrak{r}$ *is not rerandomized by the adversary. One option is to use the public key of the signature as unique identifier.*

To show that the scheme does not reveal the witness, we require that an efficient simulator exists to produce an indistinguishable transcript without the witness.

Definition 3.4 (Simulatability)**.** AS *is simulatable if there exists an efficient PPT simulator* AS.Sim *for which it holds that*

$$\left\{ x \middle| \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{AS.Setup}(1^\lambda, \mathcal{L}) \\ x \leftarrow \mathsf{AS.Sign}(\{(\mathsf{stmt}_i, \mathsf{wit}_i)\}_{i=1}^{|\mathcal{S}|}, \{m_j\}_{j=1}^{|\mathcal{T}|}) \end{array} \right\}$$
$$= \left\{ x \middle| \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{AS.Setup}(1^\lambda, \mathcal{L}) \\ x \leftarrow \mathsf{AS.Sim}(\{\mathsf{stmt}_i\}_{i=1}^{|\mathcal{S}|}, \{m_j\}_{j=1}^{|\mathcal{T}|}) \end{array} \right\}$$

3.4 PRIVACY

The privacy of the AS scheme is expressed by the security experiment in Algorithm 3.1. The mapping of signatures to messages and the number of participants must remain hidden. An Adversary $\mathcal{A}$ generates a valid set of signatures Σ, statements STMT and messages $\mathcal{M}$ and provides instructions I, J for the experiment to append signatures and messages. For two cases of the parameter $b \in \{0, 1\}$, the adversary $\mathcal{A}$ specifies which party $k \in \mathfrak{U}_t$ gets access to the witnesses wit_i where $u_{t,i}^{\mathcal{S}} = k$ and messages m_i with $u_{t,i}^{\mathcal{T}} = k$. $\mathcal{A}$ wins by calculating b correctly.

The parties on the signing and message side need to be equal, meaning there cannot be a signer without messages or messages without a signer

Definition 3.5 (Privacy)**.** *An* AS *scheme is private, if for all PPT adversaries* $\mathcal{A}$ *it holds that*

$$\left|\Pr[\mathsf{ASPrivacy}_{\mathcal{A}}^{0}(1^\lambda) = 1] - \Pr[\mathsf{ASPrivacy}_{\mathcal{A}}^{1}(1^\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

with $\mathsf{ASPrivacy}_{\mathcal{A}}^{b}(1^\lambda)$ *defined in Algorithm 3.1.*

3.5 CONSTRUCTION

We present a novel construction for the aggregatable signature scheme AS which allows the non-interactive merging of offers. Related aggregatable signature schemes [dGTP17; BJ10] are not applicable in our setting as they either require communication or aggregate values within a single messages.
Regarding privacy, our scheme provides anonymity of the mapping between individual messages and signatures. Regarding security, tampering of messages is detected by verifying the full set of messages and signatures as a whole. The important feature which results from these properties is the possibility for multiple parties to generate such signatures which are later combined into a single signature valid for the union of signatures and messages. The aggregated signature is indistinguishable from one created by a single party.
We achieve this balance by introducing randomness in the form of commitments and then revealing just enough of this randomness such that verification is feasible. Let $\mathcal{G} = (\mathbb{G}, q, G, H)$ be a cyclic group $\mathbb{G}$ of prime order q with

Algorithm 3.1 $\mathsf{ASPrivacy}^{b}_{\mathcal{A}}(1^{\lambda})$

$\mathsf{pp} \leftarrow \mathsf{AS.Setup}(1^{\lambda})$
$(I, J, \Sigma, \mathsf{STMT}, \mathcal{M}, \mathfrak{a}) \leftarrow \mathcal{A}(\mathsf{pp})$
$\Sigma_0 := \Sigma_1 := \Sigma, \mathcal{M}_0 := \mathcal{M}_1 = \mathcal{M}, \mathfrak{a}_0 := \mathfrak{a}_1 := \mathfrak{a}$
parse Σ as $\{\mathfrak{r}_i\}_{i=1}^{|\Sigma|}$ and STMT as $\{\mathsf{stmt}_i\}_{i=1}^{|\Sigma|}$
if $\mathsf{AS.Verify}(\{(\mathfrak{r}_i, \mathsf{stmt}_i)\}_{i=1}^{|\Sigma|}, \mathfrak{a}, \{m_i\}_{i=1}^{|\mathcal{M}|}) = 0$ **then return** o
parse I as $\{(\{u_{t,i}^{\mathcal{S}}\}_{t=0}^{1}, \mathsf{stmt}_i, \mathsf{wit}_i)\}_{i=1}^{|I|}$
parse J as $\{(\{u_{t,i}^{\mathcal{T}}\}_{t=0}^{1}, m_i)\}_{i=1}^{|J|}$
for all $t \in \{0, 1\}$ **do**
  $\mathfrak{U}_t^I := \{u_{t,i}^{\mathcal{S}}\}_{i=1}^{|I|}, \mathfrak{U}_t^J = \{u_{t,j}^{\mathcal{T}}\}_{j=1}^{|J|}$
  if $\mathfrak{U}_t^I \neq \mathfrak{U}_t^J$ **then return** o
  for all $k \in \mathfrak{U}_t^I$ **do**
    $\mathsf{STMT}_t^k := \{\mathsf{stmt}_i | u_{t,i}^{\mathcal{S}} = k\}_{i=1}^{|I|}$
    $\mathsf{WIT}_t^k := \{(\mathsf{stmt}_i, \mathsf{wit}_i) | u_{t,i}^{\mathcal{S}} = k\}_{i=1}^{|I|}$
    $\mathcal{M}_t^k := \{m_i | u_{t,i}^{\mathcal{T}} = k\}_{i=1}^{|J|}$
    $(\Sigma_t^k, \mathfrak{a}_t^k) \leftarrow \mathsf{AS.Sign}(\mathsf{WIT}_t^k, \mathcal{M}_t^k)$
    // % zips: $|A| = |B|$ and $A\%B := \{(a_1, b_1), \ldots, (a_{|A|}, b_{|B|})\}$
    if $\mathsf{AS.Verify}(\Sigma_t^k \% \mathsf{STMT}_t^k, \mathfrak{a}_t^k, \mathcal{M}_t^k) = 0$ **then return** o
    $\mathcal{M}_t := \mathcal{M}_t \cup \mathcal{M}_t^k, \Sigma_t := \Sigma_t \cup \Sigma_t^k, \mathfrak{a}_t \leftarrow \mathsf{Merge}(\mathfrak{a}_t, \mathfrak{a}_t^k)$
$b' \leftarrow \mathcal{A}(\Sigma_b, \mathfrak{a}_b)$ **return** b'

generators G and H where the discrete logarithm assumption holds. Further, we require a random oracle $\mathfrak{h} : \{0,1\}^* \to \mathbb{Z}_q$ which could be implemented using a hash function. These, along with the language of the actual SoK ($\mathcal{L}$) are returned as public parameters by AS.Setup.

Algorithm 3.2 $\mathsf{AS.Sign}(\{(\mathsf{stmt}_i, \mathsf{wit}_i)\}_{i=1}^{|\mathcal{S}|}, \{m_j\}_{j=1}^{|\mathcal{T}|})$

$\vec{s}, \vec{r} \xleftarrow{\$} \mathbb{Z}_q^{|\mathcal{T}|}, C := D := \emptyset$
for all $j \in [|\mathcal{T}|]$ **do**
  $C_j = G^{s_j} H^{r_j}$
  $\pi_j^{\mathcal{T}} \leftarrow \mathsf{SoK}[\mathcal{L}^{\mathsf{ped}}]\mathsf{Sign}(\mathsf{stmt} := C_j, \mathsf{wit} := (s_j, r_j), m_j)$
  $h_j = \mathfrak{h}(m_j || C_j)$
$\vec{x} \xleftarrow{\$} \mathbb{Z}_q^{|\mathcal{S}|-1}, x_{|\mathcal{S}|} := \sum_{j=1}^{|\mathcal{T}|}(h_j + s_j) - \sum_{i=1}^{|\mathcal{S}|-1} x_i$
for all $i \in [|\mathcal{S}|]$ **do**
  $D_i := G^{x_i}, \pi_i^{\mathcal{S}} \leftarrow \mathsf{SoK}[\mathcal{L}^{\mathsf{com}}]\mathsf{Sign}(stmt := D_i, wit := (x_i), 42)$
  $\mathfrak{r}_i \leftarrow \mathsf{SoK}[\mathcal{L}]\mathsf{Sign}(\mathsf{stmt}_i, \mathsf{wit}_i, D_i)$
$\mathfrak{a} := (\{(\pi_i^{\mathcal{S}}, D_i)\}_{i=1}^{|\mathcal{S}|}, \{(\pi_j^{\mathcal{T}}, C_j)\}_{j=1}^{|\mathcal{T}|}, \sum_{j=1}^{|\mathcal{T}|} r_j)$
return $(\{\mathfrak{r}_i\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a})$

3.5.1 *Signing (Algorithm 3.2)*

Using the messages m_j directly as messages in $\mathsf{SoK}[\mathcal{L}]\mathsf{Sign}$ reveals the link between SoK signatures and messages since the correct message is required for verification. Therefore we generate a Pedersen commitment $C_j = G^{s_j}H^{r_j}$ to a random value $s_j \in \mathbb{Z}_q$ with a blinding factor $r_j \in \mathbb{Z}_q$ for each message m_j. To assure that the prover knows the randomness (s_j, r_j) and link the proof to the message, we require a $\mathsf{SoK}[\mathcal{L}^{\mathsf{ped}}]$ proof $\pi_j^{\mathcal{J}}$ over the two exponents in each commitment C_j. The language is defined as $\mathcal{L}^{\mathsf{ped}} := \{C : \exists(s,r) \text{ s.t. } C = G^s \cdot H^r\}$. To get a scalar in $\mathbb{Z}_q$ we hash the concatenation of the message m_j and the commitment C_j to get $h_j = \mathfrak{h}(m_j \| C_j)$. This links the commitment to the signature as otherwise s_j and r_j are sufficient to change the message without having access to the signers secret key. With the commitment C_j and the correct message, a verifier calculates $G^{h_j} \cdot C_j = G^{h_j+s_j} \cdot H^{r_j}$ to verify if the messages belong to the signatures.
The signature $\pi_j^{\mathcal{J}}$ is necessary, proving knowledge about the values in the commitment. Without $\pi_j^{\mathcal{J}}$, an adversary, given r may calculate $G^s = C_j \cdot H^{-r}$ and reuse it in one of their commitments, convincing a verifier that the original message and s are present. In conclusion, s_j is a hidden proxy for m_j. Knowledge of s_j is required to change the message while keeping $G^{h_j} \cdot C_j$ constant.
For a single message, the privacy is irrelevant, as there was exactly one party involved. With multiple messages however, we define a secret sum $\sum_{j=1}^{|\mathcal{J}|}(h_j + s_j)$ and a public sum $r = \sum_{j=1}^{|\mathcal{J}|} r_j$. Given the value of r, the messages m_j and commitments C_j, which imply h_j, it is infeasible for the verifier to calculate an individual s_j. It is also infeasible to change a message m_j^* and adapt some s_j^* such that $\sum_{j=1}^{|\mathcal{J}|}(h_j + s_j)$ stays constant without knowing s_j. Original signers can always replace their part in a merged signature and replace it with a different part.
We use this property to distribute the value of $\sum_{j=1}^{|\mathcal{J}|}(h_j + s_j)$ randomly over the messages for $\mathsf{SoK}[\mathcal{L}]\mathsf{Sign}$. For each signature, we create a simple commitment $D_i = G^{x_i}$ with the constraint that $\sum_{i=1}^{|\mathcal{S}|} x_i = \sum_{j=1}^{|\mathcal{J}|}(h_j + s_j)$. A pragmatic approach is to use $|\mathcal{S}| - 1$ random values and calculate the last as $x_{|\mathcal{S}|} := \sum_{j=1}^{|\mathcal{J}|}(h_j + s_j) - \sum_{i=1}^{|\mathcal{S}|-1} x_i$. To assure the honest creation of these commitments, a valid argument of knowledge ($\mathsf{AoK}[\mathcal{L}^{\mathsf{com}}]$) proof $\pi_i^{\mathcal{S}}$ as defined in Section 2.6 must be attached. The language is defined as $\mathcal{L}^{\mathsf{com}} := \{D : \exists x \text{ s.t. } D = G^x\}$. We see that the product of signing side commitments D_i are equal to the product of message commitments C_j with $\prod_{i=1}^{|\mathcal{S}|} D_i = H^{-r} \cdot \prod_{j=1}^{|\mathcal{J}|}(G^{h_j} \cdot C_j)$ up to the randomness H^r which is known to the verifier, as r is published. Finally each $\mathsf{SoK}[\mathcal{L}]$ $\mathfrak{r}_i$ is generated with the supplied statements stmt_i and witnesses wit_i and the messages D_i. The aggregatable signature then consists of the commitments C_j, D_i with their SoK/AoK signatures $\pi_j^{\mathcal{J}}, \pi_i^{\mathcal{S}}$, the $\mathsf{SoK}[\mathcal{L}]$ signatures $\mathfrak{r}_i$ and the sum of blinding factors r.

Algorithm 3.3 AS.Merge($\mathfrak{a}_1, \mathfrak{a}_2$)

parse $\mathfrak{a}_1$ as $(\{(\pi^{\mathcal{S}}_{i,1}, D_{i,1})\}^{|\mathcal{S}_1|}_{i=1}, \{(\pi^{\mathcal{T}}_{j,1}, C_{j,1})\}^{|\mathcal{T}_1|}_{j=1}, r_1)$
parse $\mathfrak{a}_2$ as $(\{(\pi^{\mathcal{S}}_{i,2}, D_{i,2})\}^{|\mathcal{S}_2|}_{i=1}, \{(\pi^{\mathcal{T}}_{j,2}, C_{j,2})\}^{|\mathcal{T}_2|}_{j=1}, r_2)$
$\mathfrak{a} = (\{(\pi^{\mathcal{S}}_{i,1}, D_{i,1})\}^{|\mathcal{S}_1|}_{i=1} \cup \{(\pi^{\mathcal{S}}_{i,2}, D_{i,2})\}^{|\mathcal{S}_2|}_{i=1},$
$\{(\pi^{\mathcal{T}}_{j,1}, C_{j,1})\}^{|\mathcal{T}_1|}_{j=1} \cup \{(\pi^{\mathcal{T}}_{j,2}, C_{j,2})\}^{|\mathcal{T}_2|}_{j=1}, r_1 + r_2)$
return $\mathfrak{a}$

3.5.2 *Merging (Algorithm 3.3)*

To merge two valid signatures $\mathfrak{a}_1, \mathfrak{a}_2$, it is sufficient to add their randomness $r_1 + r_2 = r$ and use the union of the sets in $\mathfrak{a}$.

Algorithm 3.4 AS.Verify$(\{(\mathfrak{r}_i, \mathsf{stmt}_i)\}^{|\mathcal{S}|}_{i=1}, \mathfrak{a}, \{m_j\}^{|\mathcal{T}|}_{j=1})$

parse $\mathfrak{a}$ as $(\{(\pi^{\mathcal{S}}_i, D_i)\}^{|\mathcal{S}|}_{i=1}, \{(\pi^{\mathcal{T}}_j, C_j)\}^{|\mathcal{T}|}_{j=1}, r)$
$\Gamma := H^{-r}, \Delta = I$
for all $j \in [|\mathcal{T}|]$ **do**
  if $\mathsf{SoK}[\mathcal{L}^{\mathsf{ped}}]\mathsf{Verify}(\mathsf{stmt} := C_j, \pi^{\mathcal{T}}_j, m_j) = 0$ **then return** o
  $\Gamma := \Gamma C_j G^{\mathfrak{h}(m_j \| C_j)}$
for all $i \in [|\mathcal{S}|]$ **do**
  if $\mathsf{SoK}[\mathcal{L}^{\mathsf{com}}]\mathsf{Verify}(\mathsf{stmt} := D_i, \pi^{\mathcal{S}}_i, 42) = 0$ **then return** o
  if $\mathsf{SoK}[\mathcal{L}]\mathsf{Verify}(\mathfrak{r}_i, \mathsf{stmt}_i, D_i) = 0$ **then return** o
  $\Delta := \Delta D_i$
return $b := \Gamma = \Delta$

3.5.3 *Verification (Algorithm 3.4)*

If the publicly calculable product equality

$$\prod_{i=1}^{|\mathcal{S}|} D_i = H^{-r} \cdot \prod_{j=1}^{|\mathcal{T}|} (G^{h_j} \cdot C_j)$$

holds, and all proofs $(\pi^{\mathcal{T}}_j, \pi^{\mathcal{S}}_i, \mathfrak{r}_i)$ are valid, 1 is returned, 0 otherwise.

3.6 ANALYSIS

Our construction fulfills all required definitions.

Theorem 3.1 (Secure)**.** *Given a non-malleable SoK, the construction for the aggregatable signature is secure according to Definition 3.3.*

Proof. To show the security of our AS scheme, we start with the most simple scenario of one signature and one message. Then we use the adversary $\mathcal{A}$ to ef-

ficiently construct an adversary against the discrete logarithm problem. Given a discrete logarithm challenge $\mathsf{chl} = (G, G^\gamma)$ we proceed as follows.

1. Sample a statement and witness $(\mathsf{stmt}_1, \mathsf{wit}_1) \in \mathbf{R}_{\mathcal{L}}$.
2. Sample a message $m_1 \in \{0,1\}^*$ and $r_1 \in \mathbb{Z}_q$.
3. Calculate $C_1 = G^\gamma H^{r_1}$.
4. Simulate $\pi_1^{\mathcal{T}} = \mathsf{SoK}[\mathcal{L}^{\mathsf{ped}}]\mathsf{Sim}(\mathsf{stmt} = C_1, m_1)$.
5. Calculate $h_1 = \mathfrak{h}(m_1 \| C_1)$ and $D_1 = G^\gamma \cdot G^{h_1}$.
6. Simulate $\pi_1^{\mathcal{S}} = \mathsf{AoK}[\mathcal{L}^{\mathsf{com}}]\mathsf{Sim}(\mathsf{stmt} = D_1)$.
7. Sign D_1 with $\mathfrak{r}_1 = \mathsf{SoK}[\mathcal{L}]\mathsf{Sign}(\mathsf{stmt}_1, \mathsf{wit}_1, D_1)$.

This results in a valid aggregated signature

$$\left(\{\mathfrak{r}_1\}, (\{(\pi_1^{\mathcal{S}}, D_1)\}, \{(\pi_1^{\mathcal{T}}, C_1)\}, r)\right)$$

for stmt_1 and m_1. The adversary $\mathcal{A}$, given the signature above, is able to output a new valid signature for

$$\left(\{\mathfrak{r}_1\} \cup \{\mathfrak{r}_i\}_{i=2}^{|\mathcal{S}|}, (\{(\pi'^{\mathcal{S}}_i, D'_i)\}_{i=1}^{|\mathcal{S}|}, \{(\pi'^{\mathcal{T}}_j, C'_j)\}_{j=1}^{|\mathcal{T}|}, r')\right)$$

which uses the same $\mathfrak{r}_1$ along with possible other signatures $\{\mathfrak{r}_i\}_{i=2}^{|\mathcal{S}|}$ but a set of messages $M = \{m'_j\}_{j=1}^{|\mathcal{T}|}$ which does not include m_1 ($m_1 \notin M$). As $\mathfrak{r}_1$ is a non-malleable signature, it follows that $D'_1 = D_1 = G^\gamma \cdot G^{h_1}$. For all other $\{D'_i\}_{i=2}^{|\mathcal{S}|}$ created by $\mathcal{A}$, we use the efficient extractor $\mathsf{AoK}[\mathcal{L}^{\mathsf{com}}]\mathcal{E}$ which exists due to $\mathsf{AoK}[\mathcal{L}^{\mathsf{com}}]\mathsf{Verify}(\pi'^{\mathcal{S}}_i, D'_i) = 1$ to extract $\{x'_i\}_{i=2}^{|\mathcal{S}|}$ from $\{\pi'^{\mathcal{S}}_i\}_{i=2}^{|\mathcal{S}|}$ for which $D'_i = G^{x'_i}$ holds. On the message side all proofs are valid $\mathsf{SoK}[\mathcal{L}^{\mathsf{ped}}]\mathsf{Verify}(\pi_i^{\mathcal{S}}, C_i, m_i) = 1$ and are created by $\mathcal{A}$, as our simulated $\pi_1^{\mathcal{T}}$ is invalid for all $m' \in M$. Therefore, we extract $\{(s'_i, r'_i)\}_{i=1}^{|\mathcal{T}|}$ from $\{\pi'^{\mathcal{T}}_i\}_{i=1}^{|\mathcal{T}|}$ with $\mathsf{SoK}[\mathcal{L}^{\mathsf{ped}}]\mathcal{E}$. As the new signature is valid, the products are equal and by comparing exponents of G we calculate $\gamma = \sum_{j=1}^{|\mathcal{T}|} s'_j - h_1 - \sum_{i=2}^{|\mathcal{S}|} x'_i$. □

Theorem 3.2 (Simulatable)**.** *Given a simulatable SoK, the* AS *construction is simulatable according to Definition 3.4.*

Proof. The witnesses wit_i are used only in $\mathsf{SoK}[\mathcal{L}]\mathsf{Sign}$, for which an efficient simulator exists. An efficient simulator AS.Sim is defined by replacing $\mathfrak{r}_i$ with $\mathfrak{r}_i \leftarrow \mathsf{SoK}[\mathcal{L}]\mathsf{Sim}(\mathsf{stmt}, D_i)$. □

Theorem 3.3 (Private)**.** *Given a simulatable SoK, the construction for* AS *is private according to Definition 3.5.*

Proof. The information given to the adversary in the experiment about b is $\Sigma_b, \mathfrak{a}_b$. Let Σ_b be simulated by AS.Sim. Sets are closed under the union operation and thereby reveal nothing about b. For both $b \in \{0,1\}$ it holds that $\mathfrak{a}_b = (\{\pi^{\mathcal{S}}_{b,i}, D_{b,i}\}_{i=1}^{|I|}, \{\pi^{\mathcal{T}}_{b,i}, C_{b,i}\}_{i=1}^{|J|}, r_b)$. Again, the union of the sets does not reveal the initial subsets. The randomness r_b is the sum of random values and thereby itself uniformly random. As none of the values is dependent on the signer's identity $u_{b,i}$, Theorem 3.3 holds. □

3.7 CONCLUSION

Our novel aggregatable signature scheme enables multiple parties to non-interactively create a joint signature over sets of messages. Previous work on set homomorphic signatures [JMSW02] left the union only feature as an open questions. While they proposed a scheme for this setting based on groups with infeasible inversion, our scheme achieves the union only property from standard assumptions.
We will use this signature scheme as a building block of our SwapCT scheme. It allows creating partial transactions which can be merged. However, the general concept of balanced randomness is also applicable in Zcash based transactions as outlined in our future work in Section 8.5.

ASSET CONSERVATION PROOFS

4.1 OVERVIEW

4

This chapter describes our second building block. From a functional perspective, a transaction system has to ensure that any transaction consumes exactly as many tokens as it outputs. Thereby the total supply of tokens remains constant and transactions only change the ownership of tokens. In plaintext transactions, this conservation is directly achievable by summing up the inputs and subtracting the output amounts. If the sum is zero and all outputs are non-negative, the total supply of tokens remains constant. If tokens have a type and a value, the plaintext summation must hold for each type separately.
Plaintext amounts and types provide a lot of insight into transactions and allow tracing of users across transactions. One option to conceal the amounts and types is through commitments as discussed in Section 2.12. Given a set of committed input amounts and output amounts, a verifier can no longer sum the values directly. Instead, the prover has to provide a NIZK proof that the committed values sum up to zero. Additionally, the prover has to show that the output values are all positive. Without types and using homomorphic Pedersen commitments, the sum equality is achieved by a sigma protocol proving an opening of zero for the homomorphic combination of input commitments and inverse of output commitments. Together with a range proof of each output commitment to have an amount in a specified non-negative range, a verifier can be convinced that the total number of tokens is conserved.
To support a notion of type for tokens, we require the sum of amounts to be zero for each type separately. If the type is a public property of inputs and outputs, amount confidentiality is achieved though multiple sigma protocols as explained above. Revealing the type allows for tracking of users, which is why the types should be confidential too. As each transaction output is of exactly one type, a subsequent transaction using this as input is then linked to the same type. Public types effectively reduce the anonymity set to only transactions of the same type. This has similar anonymity guarantees as a separate system per type has. We therefore include the type in the amount commitment. There are multiple options to commit to a type and an amount. Poelstra et al. [PBF+17] proposed a type homomorphic commitment, which is homomorphic within each type, facilitating proof for a valid sum. In Section 4.3, we will present a NIZK proof with reduced size compared to their protocol.
One drawback of the typed hommomorphic commitment scheme is its size. As our second contribution, we propose to use a succinct[1] vector Pedersen commitment to commit to types and amounts. While this achieves an optimal commitment size of a single group element, it is no longer type homomorphic. This requires a novel NIZK proof for the conservation. Instead of

1 The size of a vector Pedersen commitment is a single group element and thereby independent of the opening size.

representing the conservation as generic arithmetic circuit and using the Bulletproof extension for bilinear circuits [LMR19] or compressed Σ-protocols [ARR21], we achieve a more efficient scheme with a fraction of multiplication gates. As a trade-off, we require one additional communication round with a single group element, i.e. an intermediate commitment. Our NIZK proof has approximately the same size as the one required for typed homomorphic commitments, but the total transaction size is reduced due to the smaller commitments.

4.2 ASSET CONSERVATION PROOFS

To preserve amounts per type individually while keeping both confidential in a commitment may be done with a NIZK proof for the language $\mathcal{L}_\emptyset$. Given two sets of commitments for the $|\mathcal{S}|$ inputs $\{\mathsf{com}_i^{\mathcal{S}}\}_{i=1}^{|\mathcal{S}|}$ and $|\mathcal{T}|$ outputs $\{\mathsf{com}_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}$ and the openings as a witness, the language assures that:

① The input and output commitments are well formed.

② The output amounts are in a valid range $a_i^{\mathcal{T}} \in \{0, \dots, 2^\beta - 1\}$ for a specified β with $|\mathcal{T}| \cdot 2^\beta < |\mathbb{M}|$ such that calculations in the message space of the commitment $\mathbb{M}$ are equal to the integers $\mathbb{Z}$.

③ For each type in the set of output types $\{\mathsf{ty}_i\}_{i=1}^{|\mathcal{T}|}$, the sum of input amounts of this type $\sum\{a_i^{\mathcal{S}} | \mathsf{ty}_i^{\mathcal{S}} = \mathsf{ty}\}$ is equal to the sum of output amounts $\sum\{a_i^{\mathcal{T}} | \mathsf{ty}_i^{\mathcal{T}} = \mathsf{ty}\}$.

In summary $\mathcal{L}_\emptyset :=$

$$\begin{cases} \mathsf{stmt} = (\{\mathsf{com}_i^{\mathcal{S}}\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{com}_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}) \\ \exists \mathsf{wit} = (\{\mathsf{ty}_i^{\mathcal{S}}, a_i^{\mathcal{S}}, r_i^{\mathcal{S}}\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{ty}_i^{\mathcal{T}}, a_i^{\mathcal{T}}, r_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}) \text{ s.t.} \\ \forall i \in [|\mathcal{S}|] : ①\mathsf{com}_i^{\mathcal{S}} = \mathsf{Commit}(\mathsf{ty}_i^{\mathcal{S}}, a_i^{\mathcal{S}}; r_i^{\mathcal{S}}) \\ \forall i \in [|\mathcal{T}|], ①\mathsf{com}_i^{\mathcal{T}} = \mathsf{Commit}(\mathsf{ty}_i^{\mathcal{T}}, a_i^{\mathcal{T}}; r_i^{\mathcal{T}}) \wedge ②a_i^{\mathcal{T}} \in \{0, \dots, 2^\beta - 1\} \\ ③\forall \mathsf{ty} \in \{\mathsf{ty}_i\}_{i=1}^{|\mathcal{T}|}, \sum\{a_i^{\mathcal{S}} | \mathsf{ty}_i^{\mathcal{S}} = \mathsf{ty}\} = \sum\{a_i^{\mathcal{T}} | \mathsf{ty}_i^{\mathcal{T}} = \mathsf{ty}\} \end{cases}$$

In the following section, and present an overview of our solutions for $\mathcal{L}_\emptyset$.

4.3 INSTANTIATION WITH HIERARCHIAL PEDERSEN COMMITMENTS

While Poelstra et al. [PBF+17] provided a working instantiation of an asset conservation proof, we present a smaller instantiation based on Bulletproofs. The proof still consists of the three elements. A sigma protocol to show an opening to zero, an asset surjection proof, which assures that the types of the outputs are equal to one of the inputs, and a range proof for the output amounts.

Using the typed homomorphic commitment scheme, we get a concrete language $\mathcal{L}_{\mathsf{THC}}$

$$:= \begin{cases} \mathsf{stmt} = (\{(C_i^{\mathcal{S}}, V_i^{\mathcal{S}})\}_{i=1}^{|\mathcal{S}|}, \{(C_i^{\mathcal{T}}, V_i^{\mathcal{T}})\}_{j=1}^{|\mathcal{T}|}) : \\ \exists \mathsf{wit} = (\{\mathsf{ty}_i^{\mathcal{S}}, a_i^{\mathcal{S}}, (r_i^{\mathcal{S}}, s_i^{\mathcal{S}})\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{ty}_j^{\mathcal{T}}, a_j^{\mathcal{T}}, (r_j^{\mathcal{T}}, s_j^{\mathcal{T}})\}_{j=1}^{|\mathcal{T}|}) : \\ \forall j \in [|\mathcal{T}|] : \begin{cases} \prod C_j^{\mathcal{T}} \cdot \vec{C}^{\mathcal{S}\,\circ -e_{i_j}} = G^{\phi_{1,j}} \\ V_j^{\mathcal{T}} = C_j^{\mathcal{T}\,a_j^{\mathcal{T}}} G^{s_j^{\mathcal{T}}}, a_i^{\mathcal{T}} \in \{0, \dots, 2^\beta - 1\} \end{cases} \\ \prod_{i=1}^{|\mathcal{S}|} V_i^{\mathcal{S}} \cdot \prod_{j=1}^{|\mathcal{T}|} V_j^{\mathcal{T}\,-1} = G^{\phi_2} \end{cases}$$

with ϕ efficiently calculatable.

For the instantiation, we parametrize the bulletproof language described in Section 2.9.1. We compress the public elements $G, \vec{C}_{\mathcal{S}}, \vec{C}_{\mathcal{T}}$ and one publicly computable element $\hat{V}$ in the function Λ. The remaining parameters are constructed as follows:

$$\begin{aligned}
\Lambda &:= (G \| \vec{C}_{\mathcal{S}} \| \vec{C}_{\mathcal{T}}^{\circ u \cdot \vec{v}^{|\mathcal{T}|}} \| \hat{V}) \\
\vec{c}_L &:= (\xi \| \vec{e} \| \vec{a}_{\mathcal{T}} \| 1 \| \mathsf{vec}(\mathbf{E}) \| \mathsf{vec}(\mathbf{B})) \\
\vec{c}_R &= (0 \| \vec{0}^{|\mathcal{S}|} \| \vec{0}^{|\mathcal{T}|} \| 0 \| \mathsf{vec}(\mathbf{E}) - \vec{1}^{|\mathcal{T}|\cdot|\mathcal{S}|} \| \mathsf{vec}(\mathbf{B}) - \vec{1}^{|\mathcal{T}|\beta}) \\
\vec{\mathrm{v}}_0 &:= (0 \| \vec{0}^{|\mathcal{S}|} \| \vec{0}^{|\mathcal{T}|} \| 0 \| \vec{y}^{|\mathcal{T}|\cdot|\mathcal{S}|+|\mathcal{T}|\beta}) \\
\vec{\mathrm{v}}_1 &:= (0 \| \vec{0}^{|\mathcal{S}|} \| \vec{0}^{|\mathcal{T}|} \| 0 \| \vec{y}^{|\mathcal{T}|\cdot|\mathcal{S}|+|\mathcal{T}|\beta}) \\
\vec{\mathrm{v}}_2 &:= (0 \| \vec{0}^{|\mathcal{S}|} \| \vec{0}^{|\mathcal{T}|} \| y^{|\mathcal{T}|} \| \vec{y}^{|\mathcal{T}|} \otimes \vec{1}^{|\mathcal{S}|} \| \vec{0}^{|\mathcal{T}|\beta}) \\
\vec{\mathrm{v}}_3 &:= (0 \| \vec{0}^{|\mathcal{S}|} \| -\vec{y}^{|\mathcal{T}|} \| 0 \| \vec{0}^{|\mathcal{T}|\cdot|\mathcal{S}|} \| \vec{y}^{|\mathcal{T}|} \otimes \vec{2}^{\beta}) \\
\vec{\mathrm{v}}_4 &:= (0 \| -\vec{y}^{|\mathcal{S}|} \| \vec{0}^{|\mathcal{T}|} \| 0 \| \vec{v}^{|\mathcal{T}|} \otimes \vec{y}^{|\mathcal{S}|} \| \vec{0}^{|\mathcal{T}|\beta})
\end{aligned}$$

with

$$\hat{V} = \prod \vec{C}_{\mathcal{T}}^{\circ -\vec{v}^{|\mathcal{T}|}} \cdot \prod \vec{V}_{\mathcal{T}}^{\circ -u\cdot\vec{v}^{|\mathcal{T}|}} \cdot \prod \vec{V}_{\mathcal{S}}^{\circ u^2} \cdot \prod V_{\mathcal{T}}^{\circ -u^2}$$

$$\phi_1(\vec{r}^{\mathcal{T}}, \vec{r}', \mathbf{E}, v) = \langle \vec{v}^{|\mathcal{T}|}, \vec{r}_{\mathcal{T}} + \mathbf{E}\vec{r}' \rangle = \textstyle\sum_j \phi_{1,j}$$

$$\phi_2(\vec{r}', \vec{s}', \vec{a}', \vec{r}_{\mathcal{T}}, \vec{s}_{\mathcal{T}}, \vec{a}_{\mathcal{T}}) = \langle \vec{1}^{|\mathcal{S}|}, \vec{a}' \circ \vec{r}' + \vec{s}' \rangle - \langle \vec{1}^{|\mathcal{T}|}, \vec{a}_{\mathcal{T}} \circ \vec{r}_{\mathcal{T}} + \vec{s}_{\mathcal{T}} \rangle, \xi = -\phi_1 + u\langle \vec{v}^{|\mathcal{T}|}, \vec{s}_{\mathcal{T}} \rangle - u^2\phi_2$$

$$\mathsf{vec}(\mathbf{E}) := (\vec{e}_{1_j} \| \vec{e}_{2_j} \| \dots \| \vec{e}_{|\mathcal{T}|_j}) \text{ and } \vec{\tilde{e}} = \vec{v}^{|\mathcal{T}|}\mathbf{E}$$

$$\mathsf{bin}(a) := \beta \text{ -bit binary representation of } a$$

$$\mathsf{vec}(\mathbf{B}) := (\mathsf{bin}(a_1^{\mathcal{T}}) \| \dots \| \mathsf{bin}(a_{|\mathcal{T}|}^{\mathcal{T}}))$$

and the constraints to

- show that $\mathbf{B}$ and $\mathbf{E}$ are binary: $\langle \vec{c}_L, \vec{c}_R \circ \vec{\mathrm{v}}_0 \rangle = 0, \langle \vec{c}_L - \vec{c}_R - \vec{1}^m, \vec{\mathrm{v}}_1 \rangle = 0$
- show that $\mathbf{E}$ consists of unit vectors and $\vec{c}_L$ is 1 at the position of $\hat{V}$: $\langle \vec{c}_L, \vec{\mathrm{v}}_2 \rangle = \langle \vec{1}^{|\mathcal{T}|+1}, \vec{y}^{|\mathcal{T}|+1} \rangle$
- show that all output commitment values have a binary decomposition in $\mathbf{B}$: $\langle \vec{c}_L, \vec{\mathrm{v}}_3 \rangle = 0$
- show that each output type has a corresponding input type: $\langle \vec{c}_L, \vec{\mathrm{v}}_4 \rangle = 0$.

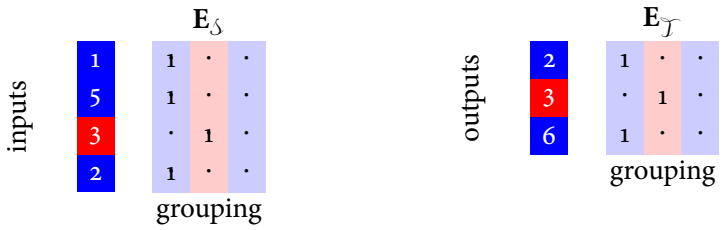


Figure 4.1: Our $\mathcal{L}_{\emptyset}$ proof structure with 4 inputs and 3 outputs. Dots are zeros.

4.4 CONSTRUCTION FOR COMMITTED VECTORS

We create a smaller non-interactive zero-knowledge (NIZK) proof for the language $\mathcal{L}_{\emptyset}$ based on grouping of amounts by type and using succinct type commitments. Similar to confidential assets, we use the Bulletproof [BBB+18] structure presented in Omniring [LRR+19] for the compression to a logarithmic size transcript.

Without the type homomorphic property of a THC scheme, our protocol has to check the amounts and types individually. The idea of our construction is that for each type present in the outputs, we check that the sum of input amounts of this type is equal to the sum of output amounts for the same type. Figure 4.1 shows an example of a transaction with four inputs $\mathcal{S}$ (1 blue, 5 blue, 3 red, 2 blue) and three outputs $\mathcal{T}$ (2 blue, 3 red, 6 blue). To select inputs and outputs of equal type, we construct two binary matrices $\mathbf{E}_{\mathcal{S}}$ and $\mathbf{E}_{\mathcal{T}}$, each with one column per output up to a total of $|\mathcal{T}|$ columns. For ease of readability of the resulting language $\mathcal{L}_{\Pi}$ below, we use circled numbers as references. The binary matrices correspond to ①. The i-th column of $\mathbf{E}_{\mathcal{S}}$ contains a 1 in all positions with an equal type to output i. I.e. the first output is of type blue, so the first column of $\mathbf{E}_{\mathcal{S}}$ is 1 at the position of every blue input. The outputs are grouped the same way in $\mathbf{E}_{\mathcal{T}}$. In our example (cf. Figure 4.1) the first column indicates all outputs with the same type as the first output. In the case that multiple outputs have the same type, in our example, the first and last output is blue, only one output is considered for the grouping of this type, in our case the first one, thus only the first column of $\mathbf{E}_{\mathcal{S}}$ and $\mathbf{E}_{\mathcal{T}}$ are used for type blue. Subsequent outputs of the same type use a zero vector for the corresponding columns of $\mathbf{E}_{\mathcal{S}}$ and $\mathbf{E}_{\mathcal{T}}$. This achieves two objectives. First, each input and output belongs to exactly one group, i.e. each row is a unit vector ②. And second, it hides the true number of unique types in the outputs. A transaction with $|\mathcal{T}|$ outputs may have outputs with up to $|\mathcal{T}|$ different types (assuming $|\mathcal{S}| \geq |\mathcal{T}|$).

There are at most $\min(|\mathcal{S}|, |\mathcal{T}|)$ *types in a transaction but for simplicity we assume* $|\mathcal{S}| \geq |\mathcal{T}|$

With each type grouped into a column, the input amounts belonging to the group are summed up and the sum of outputs in the output group of the same type is subtracted. I.e. in the first column, the blue inputs 1,5,2 are added to 8 and the blue outputs 2,6 referenced by the first column of $\mathbf{E}_{\mathcal{T}}$ subtract to zero. If this holds for every column, we achieve typed conservation.

So far, there is no connection of the matrices $\mathbf{E}_{\mathcal{S}}, \mathbf{E}_{\mathcal{T}}$ to the actual input $\vec{C}_{\mathcal{S}}$ and output $\vec{C}_{\mathcal{T}}$ commitments of the statement. To create this link, we require the concrete instantiation of our typed commitments as described in the prelim-

inaries, specifically $\mathsf{TC.Commit}(\mathsf{ty}, a; r) = G^a F^{\mathsf{ty}} H^r$. The prover first needs to show that the commitments are well formed by proving knowledge of their exponents for the inputs $\vec{\mathsf{ty}}^{\mathcal{S}}, \vec{a}^{\mathcal{S}}, \vec{r}^{\mathcal{S}}$ and outputs $\vec{\mathsf{ty}}^{\mathcal{T}}, \vec{a}^{\mathcal{T}}, \vec{r}^{\mathcal{T}}$ with respect to G, F and H (3).

To check the type equality of inputs and outputs, we use the homomorphic property of the typed commitments. Note that this is *not* the typed homomorphic property from Definition 2.17. A homomorphic commitment $C = G^a F^{\mathsf{ty}} H^r$ can be inverted C^{-1} into a commitment to the values of $-\mathsf{ty}, -a, -r$. Proving that two commitments $C_1 = G^{a_1} F^{\mathsf{ty}} H^{r_1}, C_2 = G^{a_2} F^{\mathsf{ty}} H^{r_2}$ are of the same type ty having different amounts a_1, a_2 and randomness r_1, r_2 is efficiently achieved by proving knowledge of an opening to $C_1 \cdot C_2^{-1} = G^{a_1 - a_2} F^0 H^{r_1 - r_2}$ with a type of *zero*. We construct an or-proof of type equality using a secret bit $b \in \{0, 1\}$. If a prover has knowledge of an opening ϕ, η to the product $C_1^b \cdot C_2^{-b} = G^\phi H^\eta$, either the types are equal and $\phi = a_1 - a_2, \eta = r_1 - r_2$ or $b = 0$ with $\phi = \eta = 0$.

To check if input j has the same type as output i, we use the element $\mathbf{E}_{\mathcal{S}}[j, i]$ as bit in the product of the commitments $\vec{C}_{\mathcal{S}}[i]^{\mathbf{E}_{\mathcal{S}}[j,i]} \cdot \vec{C}_{\mathcal{T}}[i]^{-\mathbf{E}_{\mathcal{S}}[j,i]} = G^{\phi_{\mathsf{ty},\mathcal{S},i,j}} F^0 H^{\eta_{\mathsf{ty},\mathcal{S},i,j}}$. If $\mathbf{E}_{\mathcal{S}}[j, i] = 0$, the opening of the product is always zero and proves nothing about the committed values, however, if $\mathbf{E}_{\mathcal{S}}[j, i] = 1$, the types must be equal to result in an opening with F^0 (4). E.g. the input matrix $\mathbf{E}_{\mathcal{S}}$ indicates that the second input has the same type as the first output $\mathbf{E}_{\mathcal{S}}[2, 1] = 1$. To verify this, we raise the input commitment to this bit and the output commitment to the inverse: $5^{\mathbf{E}_{\mathcal{S}}[2,1]} \cdot 2^{-\mathbf{E}_{\mathcal{S}}[2,1]} = 3$. The type cancels out and the prover knows a valid opening of the result. The output matrix $\mathbf{E}_{\mathcal{T}}$ is linked to the output commitments in the same way (5).

Instead of achieving a type of zero, we apply the same method to achieve amounts of zero. For each group of equal type, represented by a column in $\mathbf{E}_{\mathcal{S}}$, we raise the input commitment vector $\vec{C}_{\mathcal{S}}$ to the elements of the i-th column of $\mathbf{E}_{\mathcal{S}}$ and then multiply the results together. I.e. input group 1: $1^1 \cdot 5^1 \cdot 3^0 \cdot 2^1 = 8$ input group 2: $1^0 \cdot 5^0 \cdot 3^1 \cdot 2^0 = 3$ input group 3: $1^0 \cdot 5^0 \cdot 3^0 \cdot 2^0 = 0$. This results in $|\mathcal{T}|$ commitments, each to the sum of input amounts in the specific type. We repeat this for the output commitments and get the sum of output amounts per type. I.e. output group 1: $2^1 \cdot 3^0 \cdot 6^1 = 8$ output group 2: $2^0 \cdot 3^1 \cdot 6^0 = 3$ output group 3: $2^0 \cdot 3^0 \cdot 6^0 = 0$. The input sums per type have to be equal to the output sum of the same type. Therefore, the prover has to present an opening to the product of the combined input commitments and the inverse of output commitments for each type. I.e. group 1: $8 \cdot 8^{-1} = 0$ group 2: $3 \cdot 3^{-1} = 0$ group 3: $0 \cdot 0^{-1} = 0$ The non-zero type and randomness of the openings for group i are $\chi_{a,i}$ and $\eta_{a,i}$ (6).

To ensure that the sum of amounts hold in $\mathbb{Z}$ while the commitment space is $\mathbb{Z}_q$, we add a range proof for all $a^{\mathcal{T}}[i]$. We prove that a value is in a given range $\{0, \ldots, 2^\beta - 1\}$ by proving knowledge of its β-bit binary representation. The representations of all output amounts $\vec{a}^{\mathcal{T}}$ yield the binary matrix $\mathbf{B}$. In most

systems, β equals 64 ⑦. Given our instantiations and efficiently computable χ's and η's from the witness, we get the language $\mathcal{L}_{\Pi} :=$

$$\begin{cases} \mathsf{stmt} = (\vec{C}_{\mathcal{S}}, \vec{C}_{\mathcal{T}}) \exists \mathsf{wit} = (\vec{\mathsf{ty}}^{\mathcal{S}}, \vec{a}^{\mathcal{S}}, \vec{r}^{\mathcal{S}}, \mathbf{E}_{\mathcal{S}}, \mathbf{E}_{\mathcal{T}}, \mathbf{B}, \vec{\mathsf{ty}}^{\mathcal{T}}, \vec{a}^{\mathcal{T}}, \vec{r}^{\mathcal{T}}) \text{ s.t.} \\ ⑦\mathbf{B} \text{ binary, size } \beta \times |\mathcal{T}|; ①\mathbf{E}_{\mathcal{S}} \text{ binary, size } |\mathcal{S}| \times |\mathcal{T}|; \mathbf{E}_{\mathcal{T}} \text{ binary, size } |\mathcal{T}| \times |\mathcal{T}| \\ \forall i \in [|\mathcal{T}|], \begin{cases} ③\vec{C}_{\mathcal{T}}[i] = G^{\vec{a}^{\mathcal{T}}[i]} F^{\vec{\mathsf{ty}}^{\mathcal{T}}[i]} H^{\vec{r}^{\mathcal{T}}[i]}; ⑦\mathbf{B}[i] \text{ binary repr. of } \vec{a}^{\mathcal{T}}[i] \\ \forall j \in [|\mathcal{S}|], ④\vec{C}_{\mathcal{S}}[j]^{\mathbf{E}_{\mathcal{S}}[j,i]} \cdot \vec{C}_{\mathcal{T}}[i]^{-\mathbf{E}_{\mathcal{S}}[j,i]} = G^{\phi_{\mathsf{ty},\mathcal{S},i,j}} F^0 H^{\eta_{\mathsf{ty},\mathcal{S},i,j}} \\ \forall j \in [|\mathcal{T}|], ⑤\vec{C}_{\mathcal{T}}[j]^{\mathbf{E}_{\mathcal{T}}[j,i]} \cdot \vec{C}_{\mathcal{T}}[i]^{-\mathbf{E}_{\mathcal{T}}[j,i]} = G^{\phi_{\mathsf{ty},\mathcal{T},i,j}} F^0 H^{\eta_{\mathsf{ty},\mathcal{T},i,j}} \\ ⑥\prod \vec{C}_{\mathcal{S}}^{\circ \mathbf{E}_{\mathcal{S}}[i]} \cdot \prod \vec{C}_{\mathcal{T}}^{\circ -\mathbf{E}_{\mathcal{T}}[i]} = G^0 F^{\chi_{a,i}} H^{\eta_{a,i}} \end{cases} \\ ②\mathbf{E}_{\mathcal{S}} \cdot \vec{1}^{|\mathcal{T}|} = \vec{1}^{|\mathcal{S}|}; \mathbf{E}_{\mathcal{T}} \cdot \vec{1}^{|\mathcal{T}|} = \vec{1}^{|\mathcal{T}|}; \forall i \in [|\mathcal{S}|] : ③\vec{C}_{\mathcal{S}}[i] = G^{a_i^{\mathcal{S}}} F^{\mathsf{ty}_i^{\mathcal{S}}} H^{r_i^{\mathcal{S}}} \end{cases}$$

4.5 VECTOR PEDERSEN COMMITMENT NIZK INSTANTIATION

To efficiently prove equivalency of a single position in multiple pairs of vector Pedersen commitments, i.e. only amount or type equivalency, we present an extended outer protocol. It enables a prover to deal with the "noise" introduced by unequal positions in the commitments and the compression of multiple comparisons.

Our protocol allows the prover to commit to arbitrary values $\vec{k}'$ at an intermediate stage, where only a partial set of challenge variables is known to the prover. The committed intermediate values are then usable in subsequent constraints and may be compressed with the remaining challenge variables. In our case, the prover is able to commit to the noise introduced by the amounts and blinding factors of the statement commitments $\vec{C}_{\mathcal{S}}, \vec{C}_{\mathcal{T}}$ in the compression of constraints for type equality. The committed values $\vec{k}'$ are then usable in subsequent constraints, e.g. to cancel out noise when proving the knowledge of a discrete logarithm of aggregated commitments where only some generators have an exponent of 0. To maintain the security of the protocol, the commitment K' to the values $\vec{k}'$ with a randomness r_κ is a vector Pedersen commitment with bases $\vec{K}'_G \in \mathbb{G}^{|\vec{k}'|+1}$ generated by the verifier. More concretely, we build a NIZK proof for a language which is a parametrization of the following language: $\mathcal{L}_{\mathsf{ebp}} :=$

$$\begin{cases} \mathsf{stmt} = \vec{K} \in \mathbb{G}^{m'} \exists \mathsf{wit} = (\vec{c}_L := \vec{c}_{L,1} \| \vec{c}_{L,2}, \vec{c}_R) \in ({\mathbb{Z}_q}^{m+n} \times {\mathbb{Z}_q}^{m+n}) \text{ s.t.} \\ \prod \Lambda(\vec{K}, K', \vec{K}'_G, u, v, v', v'')^{\circ c_{L,1}} = I; \\ \forall i \in \{\vec{\mathrm{v}}_i : \mathsf{cls}(i) = \mathfrak{one}\} : \langle \vec{c}_L - \vec{c}_R - \vec{1}^{m+n}, \vec{\mathrm{v}}_i \rangle = \hat{\mathrm{v}}_i \\ \forall i \in \{\vec{\mathrm{v}}_i : \mathsf{cls}(i) = \mathfrak{mul}\} : \langle \vec{c}_L, \vec{c}_R \circ \vec{\mathrm{v}}_i \rangle = \hat{\mathrm{v}}_i \\ \forall i \in \{\vec{\mathrm{v}}_i : \mathsf{cls}(i) = \mathfrak{dir}\} : \langle \vec{c}_L, \vec{\mathrm{v}}_i \rangle = \hat{\mathrm{v}}_i \\ \forall i \in \{\vec{\mathrm{v}}_i : \mathsf{cls}(i) = \mathfrak{sum}\} : \langle \vec{c}_L, \vec{\mathrm{v}}_i \rangle + \langle \vec{c}_R, \vec{\mathrm{v}}'_i \rangle = \hat{\mathrm{v}}_i \end{cases}$$

The parameters for the language $\mathcal{L}_{\mathsf{ebp}}$ are first the public function Λ and the set of constraints $\vec{\mathrm{v}}_i$ with class $\mathsf{cls}(i)$ and value $\hat{\mathrm{v}}_i$. Instead of a single challenge

variable v, we support arbitrary more to achieve better independent compression. In our setting they are v, v', v''. Λ takes $\vec{K}, K', \vec{K}'_G, u, v, v', v''$ as input. The intermediate commitment K' is supplied by the prover after receiving the challenge variables v, v', v'' but importantly *before* the variable u.
Let $m = |\vec{c}_L|$ and $n = |\Lambda(\dots)|$, he present the full protocol with changes to Omniring are marked in boxes.

$\mathcal{V}$: $v, v', v'' \xleftarrow{\$} \mathbb{Z}_q, \vec{K}_G \xleftarrow{\$} \mathbb{G}^5$

$\mathcal{P} \leftarrow \mathcal{V}$: $v, v', v'', \vec{K}_G$

$\mathcal{P}$: $r_\kappa \xleftarrow{\$} \mathbb{Z}_q$, commit $K' = \prod \vec{K}_G^{\circ(\phi_{\mathsf{ty}}, \eta_{\mathsf{ty}}, \chi_a, \eta_a, r_\kappa)}$

$\mathcal{P} \rightarrow \mathcal{V}$: $\vec{K}'$

$\mathcal{V}$: $u \xleftarrow{\$} \mathbb{Z}_q, D \xleftarrow{\$} \mathbb{G}, \vec{P} \xleftarrow{\$} \mathbb{G}^n, \vec{G}' \xleftarrow{\$} \mathbb{G}^{m-n}, \vec{H} \xleftarrow{\$} \mathbb{G}^m$
$\mathcal{P} \leftarrow \mathcal{V}$: $u, D, \vec{P}, \vec{G}', \vec{H}$
$\mathcal{P}, \mathcal{V}$: For $w \in \mathbb{Z}_q$ define $\vec{G}_w := (\boxed{\Lambda(\vec{K}, K', \vec{K}_G, u, v, v', v'')}^{\circ w} \circ \vec{P} \| \vec{G}')$
$\mathcal{P}$: $r_A \xleftarrow{\$} \mathbb{Z}_q, A := D^{r_A} \vec{G}_0^{\vec{c}_L} \vec{H}^{\vec{c}_R}$
$\mathcal{P} \rightarrow \mathcal{V}$: A
$\mathcal{V}$: $w \xleftarrow{\$} \mathbb{Z}_q$
$\mathcal{P} \leftarrow \mathcal{V}$: w
$\mathcal{P}$: 1. $\vec{s}_L \xleftarrow{\$} {\mathbb{Z}_q}^m, \vec{s}_R = (\forall i \in [m] : \begin{cases} 0 \text{ if } \vec{c}_R[i] = 0 \\ s \xleftarrow{\$} \mathbb{Z}_q \text{ else} \end{cases})$

2. $r_S \xleftarrow{\$} \mathbb{Z}_q, S := D^{r_S} \vec{G}_w^{\vec{s}_L} \vec{H}^{\vec{s}_R}$

$\mathcal{P} \rightarrow \mathcal{V}$: S
$\mathcal{V}$: $y, z \xleftarrow{\$} \mathbb{Z}_q$
$\mathcal{P} \leftarrow \mathcal{V}$: y, z

Now the prover and the verifier compress the constraints of the parametrization. Each constraint $\vec{\mathrm{v}}_i$ gets a unique index i and $\mathsf{cls}(\vec{\mathrm{v}}_i)$ returns the class $\{\mathfrak{mul}, \mathfrak{dir}, \mathfrak{sum}, \mathfrak{one}\}$ of the constraint. Define $\boxed{\delta := \langle \vec{\alpha}, \vec{\mu} \rangle + \langle \vec{1}^m, \vec{\nu} \rangle + \sum_i z^i \hat{\mathrm{v}}_i}$ and

$$\vec{\Theta} := \sum_{i:\mathsf{cls}(\vec{\mathrm{v}}_i)=\mathfrak{mul}} z^i \vec{\mathrm{v}}_i \qquad \vec{\mu} := \sum_{i:\mathsf{cls}(\vec{\mathrm{v}}_i)\neq\mathfrak{mul}} z^i \vec{\mathrm{v}}_i \qquad \vec{\nu} := \sum_{i:\mathsf{cls}(\vec{\mathrm{v}}_i)=\mathfrak{one}} z^i \vec{\mathrm{v}}_i$$
$$\vec{\omega} := \sum_{i:\vec{\mathrm{v}}'_i \neq 0} z^i \vec{\mathrm{v}}'_i \qquad \vec{\alpha} := \vec{\Theta}^{\circ -1} \circ (\vec{\omega} - \vec{\nu}) \qquad \vec{\beta} := \vec{\Theta}^{\circ -1} \circ \mu$$

$\mathcal{P}$: Define polynomials in X:

$$l(X) := \vec{c}_L + \vec{\alpha} + \vec{s}_L \cdot X$$
$$r(X) := \vec{\Theta} \cdot (\vec{c}_R + \vec{s}_R \cdot X) + \vec{\mu}$$

with $t(X) := \langle l(X), r(X) \rangle = \delta + t_1 X + t_2 X^2$ for some t_1 and t_2, let $\tau_1, \tau_2 \xleftarrow{\$} \mathbb{Z}_q, T_1 := G^{t_1} D^{\tau_1}, T_2 := G^{t_2} D^{\tau_2}$

$\mathcal{P} \to \mathcal{V}$: T_1, T_2
$\mathcal{V}$: $x \xleftarrow{\$} \mathbb{Z}_q, Q \xleftarrow{\$} \mathbb{G}$
$\mathcal{P} \leftarrow \mathcal{V}$: x, Q
$\mathcal{P}$: 1. $\boxed{\tau := \tau_1 x + \tau_2 x^2}$; $r := r_A + r_S x$; $(\vec{l}, \vec{r}, t) := (l(x), r(x), t(x))$
2. padd $\vec{l}$ and $\vec{r}$ with 0 to length $2^{\lceil \log_2(m) \rceil}$
3. $\pi_{\mathsf{IP}} \leftarrow$ **IPprove**$(\vec{l}, \vec{r}, \vec{G}_w, \vec{H}^{\vec{\Theta}^{\circ -1}}, Q)$
$\mathcal{P} \to \mathcal{V}$: $\tau, r, \pi_{\mathsf{IP}}, t$
$\mathcal{V}$: verify $P = AS^x \vec{G}_w^{\vec{\alpha}} \vec{H}^{\vec{\beta}} \cdot D^{-r} \cdot Q^t$
and verify **IPvf**$(\pi_{\mathsf{IP}}, \vec{G}_w, \vec{H}^{\vec{\Theta}^{\circ -1}}, P, Q) = 1 \wedge G^t D^\tau = \boxed{G^\delta T_1^x T_2^{x^2}}$

We use the inner product protocol (**IPprove**, **IPvf**) from Bulletproofs which satisfies the language

$$\left((P, Q, G, H) \in \mathbb{G} : \exists \vec{l}, \vec{r} \in {\mathbb{Z}_q}^m \text{ s.t. } P = \vec{G}^{\circ \vec{l}} \vec{H}^{\circ \vec{r}} Q^{\langle \vec{l}, \vec{r} \rangle} \right)$$

Given this NIZK protocol for $\mathcal{L}_{\text{ebp}}$ in Section 2.9.1, we construct the protocol for $\mathcal{L}_{\Pi}$. Therefore, we split $\mathcal{L}_{\Pi}$ into its four main parts (well formedness of commitments, grouping of types, amount equality and valid amount ranges) and explain how each part contributes partial parameters for a single parametrization of $\mathcal{L}_{\text{ebp}}$.

4.5.1 *Well-formedness of Commitments*

In the first part of parametrization, the prover shows the well-formedness of the input and output commitments (③ in $\mathcal{L}_{\Pi}$). This shows that the commitments have a valid opening and the prover knows the opening. To do this, we use the proof for knowledge of discrete logarithms in $\mathcal{L}_{\text{ebp}}$. For a single commitment $C = G^a F^{\mathsf{ty}} H^r$, we encode $\Lambda(C, \dots) := (G, F, H, C)$ and specify the witness vectors as $\vec{c}_L = (-a, -\mathsf{ty}, -r, 1)$ and $\vec{c}_R = \vec{0}^4 := (0, 0, 0, 0)$. This is sufficient to prove well-formedness of the commitment C, if the last element of $\vec{c}_L$ is exactly one: $\vec{c}_L[4] = 1$. Our outer protocol proves that $\prod \Lambda(\dots)^{\circ c_{L,1}} = I$ and with $\vec{c}_{L,1} = \vec{c}_L$, the first three elements of the witness must be the exponents of G, F, H to cancel out the commitment C and therefore be $\vec{c}_L[1] = -a, \vec{c}_L[2] = -\mathsf{ty}, \vec{c}_L[3] = -r$.
While the outer protocol allows us to prove only one identity, we compress all the identity relations for each input and output commitment by raising each to a unique power of the challenge variable v. For a honestly chosen $v \neq 0$ by the verifier, the resulting polynomial of constraints only results in the identity, if all coefficients are the identity. The exponents of G, F, H then show the knowledge of all commitment openings. We construct the partial parameters of $\mathcal{L}_{\text{ebp}}$ as:

$\Lambda^{\text{wf}} := (\, G \,\|\, F \,\|H\|\vec{C}_{\mathcal{S}}\|\vec{C}_{\mathcal{T}}\,)$

$\vec{c}_L^{\text{wf}} := (\phi_0\|\chi_0\|\eta_0\|\vec{v}^{|\mathcal{S}|+|\mathcal{T}|}\|\vec{a}^{\mathcal{S}}\|\vec{a}^{\mathcal{T}}\|\vec{\text{ty}}^{\mathcal{S}}\|\vec{\text{ty}}^{\mathcal{T}}\|\vec{r}^{\mathcal{S}}\|\vec{r}^{\mathcal{T}}\,);\ \vec{c}_R^{\text{wf}} = \vec{0}^{3+4(|\mathcal{S}|+|\mathcal{T}|)}$

$\vec{\text{v}}_1^{\text{wf}} := (\,1\ \cdot\ \cdot\ \cdot\ \cdot\ \|\vec{v}^{|\mathcal{S}|+|\mathcal{T}|}\|\ \cdot\ \cdot\ \cdot\ \cdot\,)\colon \mathfrak{dir}, \hat{\text{v}}_1^{\text{wf}} = 0$

$\vec{\text{v}}_2^{\text{wf}} := (\,\cdot\ 1\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \|\,\vec{v}^{|\mathcal{S}|+|\mathcal{T}|}\,\|\,\cdot\ \cdot\,)\colon \mathfrak{dir}, \hat{\text{v}}_2^{\text{wf}} = 0$

$\vec{\text{v}}_3^{\text{wf}} := (\,\cdot\ \cdot\ 1\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \|\vec{v}^{|\mathcal{S}|+|\mathcal{T}|}\,)\colon \mathfrak{dir}, \hat{\text{v}}_3^{\text{wf}} = 0$

$\vec{\text{v}}_4^{\text{wf}} := (\,\cdot\ \cdot\ \cdot\ \vec{y}^{|\mathcal{S}|+|\mathcal{T}|}\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\,)\colon \mathfrak{dir}, \hat{\text{v}}_4^{\text{wf}} = \sum_{i=1}^{|\mathcal{S}|+|\mathcal{T}|}(vy)^i$

with $\phi_0 = -\langle\vec{v}^{|\mathcal{S}|+|\mathcal{T}|}, \vec{a}_{\mathcal{S}}\|\vec{a}_{\mathcal{T}}\rangle$

$\chi_0 = -\langle\vec{v}^{|\mathcal{S}|+|\mathcal{T}|}, \vec{\text{ty}}_{\mathcal{S}}\|\vec{\text{ty}}_{\mathcal{T}}\rangle$

$\eta_0 = -\langle\vec{v}^{|\mathcal{S}|+|\mathcal{T}|}, \vec{r}_{\mathcal{S}}\|\vec{r}_{\mathcal{T}}\rangle$. The first three constraints ensure that the vector $\vec{c}_L^{\text{wf}}$, namely the openings $\vec{a}^{\mathcal{S}}, \vec{a}^{\mathcal{T}}, \vec{\text{ty}}^{\mathcal{S}}, \vec{\text{ty}}^{\mathcal{T}}, \vec{r}^{\mathcal{S}}, \vec{r}^{\mathcal{T}}$ and combined exponents ϕ_0, χ_0, η_0 are correctly encoded. This holds, as e.g. the inner product $\langle\vec{c}_L, \vec{\text{v}}_1^{\text{wf}}\rangle$ is only 0, if $\vec{c}_L[1] = -\left\langle\vec{v}^{|\mathcal{S}|+|\mathcal{T}|}, \vec{c}_{L,a}\right\rangle$ with

$$\vec{c}_{L,a} := (\vec{c}_L[3+|\mathcal{S}|+|\mathcal{T}|+1], \ldots, \vec{c}_L[3+2(|\mathcal{S}|+|\mathcal{T}|)])$$

. Combined with the constraint that

$$\prod(\vec{C}_{\mathcal{S}}\|\vec{C}_{\mathcal{T}})^{\circ\vec{v}^{|\mathcal{S}|+|\mathcal{T}|}} \cdot G^{\phi_0} \cdot F^{\chi_0} \cdot H^{\eta_0} = I$$

the verifier is convinced that the vector $\vec{c}_{L,a}$ contains the amounts of the commitments $\vec{C}_{\mathcal{S}}$ and $\vec{C}_{\mathcal{T}}$. Constraints $\vec{\text{v}}_2^{\text{wf}}$ and $\vec{\text{v}}_3^{\text{wf}}$ work equivalently for the type and randomness.

Constraint $\vec{\text{v}}_4^{\text{wf}}$ ensures that $\vec{c}_L$ has consecutive powers of v for elements 4 to $4+|\mathcal{S}|+|\mathcal{T}|$. Instead of a single constraint for each commitment individually, we check them all together with a polynomial of $|\mathcal{S}|+|\mathcal{T}|$ powers of the challenge variable y as $\langle\vec{c}_L, \vec{\text{v}}_4\rangle = \hat{\text{v}}_4^{\text{wf}} \Leftrightarrow (\vec{c}_L[4], \ldots, \vec{c}_L[4+|\mathcal{S}|+|\mathcal{T}|]) = \vec{v}^{|\mathcal{S}|+|\mathcal{T}|}$.

4.5.2 *Grouping of Equal Types*

Given that all commitments are well-formed, we describe the partial parameters for $\mathcal{L}_{\text{ebp}}$ grouping inputs and outputs of equal type (④⑤ in $\mathcal{L}_{\Pi}$) as detailed in Section 4.4. We first group the inputs and show that $\mathbf{E}_{\mathcal{S}}$ is correct. The input commitments $\vec{C}_{\mathcal{S}}$ at the positions of ones in $\mathbf{E}_{\mathcal{S}}[i]$ have to commit to the same type as the i-th output commitment $C_{\mathcal{T}}[i]$. We achieve this by proving knowledge of exponents for G and H in $G^{\phi_{\text{ty},S,i,j}} H^{\eta_{\text{ty},S,i,j}} = \vec{C}_{\mathcal{S}}[j]^{\mathbf{E}_{\mathcal{S}}[j,i]} \cdot \vec{C}_{\mathcal{T}}[i]^{-\mathbf{E}_{\mathcal{S}}[j,i]}$ for each position $j \in [|\mathcal{S}|]$. If $\mathbf{E}_{\mathcal{S}}[j,i] = 1$, the openings $\text{ty}^{\mathcal{S}}[j]$ and $\text{ty}^{\mathcal{T}}[i]$ of the input and output commitment must be equal. Otherwise, there will be a non-zero type in the commitment product, which is not satisfiable without an exponent of F. As all products of commitments must result in an exponent of zero for F, we compress all comparisons into a polynomial with two variables v, v'. Each row of $\mathbf{E}_{\mathcal{S}}$ is separated by the challenge variable v and each column by v'. The product of $\vec{C}_{\mathcal{S}}^{\circ\vec{\tilde{e}}_{\text{ty},\mathcal{S}}} \cdot \vec{C}_{\mathcal{T}}^{\circ\vec{\tilde{e}}'_{\text{ty},\mathcal{T}}}$ with $\vec{\tilde{e}}_{\text{ty},\mathcal{S}} = \mathbf{E}_{\mathcal{S}} \circ (\vec{v}^{|\mathcal{S}|} \cdot (\vec{v'}^{|\mathcal{T}|})^\top) \cdot \vec{1}^{|\mathcal{T}|}$ and $\vec{\tilde{e}}'_{\text{ty},\mathcal{T}} = \vec{1}^{|\mathcal{S}|} \cdot (-\mathbf{E}_{\mathcal{S}} \circ (\vec{v}^{|\mathcal{S}|} \cdot (\vec{v'}^{|\mathcal{T}|})^\top)$

has to result in $G^{\phi'_{\mathsf{ty}}} F^0 H^{\eta'_{\mathsf{ty}}}$. The exponents of the remaining elements G and H are $\phi'_{\mathsf{ty}} = \Upsilon'_{\mathsf{ty}}(a), \eta'_{\mathsf{ty}} = \Upsilon'_{\mathsf{ty}}(r)$ with

$$\begin{aligned}\Upsilon'_{\mathsf{ty}}(\psi) = &- \vec{1}^{|\mathcal{S}|} \cdot \left(-\mathbf{E}_{\mathcal{S}} \circ ((\vec{v}^{|\mathcal{S}|} \circ \vec{\psi}^{\mathcal{S}}) \cdot (\vec{v'}^{|\mathcal{T}|})^\top) \right) \\ &+ \vec{1}^{|\mathcal{S}|} \cdot (-\mathbf{E}_{\mathcal{S}} \circ (\vec{v}^{|\mathcal{S}|} \cdot (\vec{v'}^{|\mathcal{T}|} \circ \vec{\psi}^{\mathcal{T}})^\top).\end{aligned}$$

After proving that $\mathbf{E}_{\mathcal{S}}$ is constructed correctly, the proof needs to show correctness of the output grouping specified by $\mathbf{E}_{\mathcal{T}}$. This is done analogous to the input grouping. The prover calculates a vector of exponents $\vec{\tilde{e}}''_{\mathsf{ty},\mathcal{T}}$ for $\vec{C}^{\mathcal{T}}$ which compresses all comparisons into a second polynomial in u, v'', where instead of v', the columns of $\mathbf{E}_{\mathcal{T}}$ are separated by the challenge v'' resulting in

$$\vec{\tilde{e}}''_{\mathsf{ty},\mathcal{T}} = \mathbf{E}_{\mathcal{T}} \circ (\vec{v}^{|\mathcal{T}|} \cdot (\vec{v''}^{|\mathcal{T}|})^\top) \cdot \vec{1}^{|\mathcal{T}|} + \vec{1}^{|\mathcal{T}|} \cdot (-\mathbf{E}_{\mathcal{T}} \circ (\vec{v}^{|\mathcal{T}|} \cdot (\vec{v''}^{|\mathcal{T}|})^\top).$$

The polynomial has to result in an exponent of zero for F and the openings for G and H are $\phi''_{\mathsf{ty}} = \Upsilon''_{\mathsf{ty}}(a), \eta''_{\mathsf{ty}} = \Upsilon''_{\mathsf{ty}}(r)$ with

$$\begin{aligned}\Upsilon''_{\mathsf{ty}}(\psi) = &\vec{1}^{|\mathcal{T}|} \cdot (-\mathbf{E}_{\mathcal{T}} \circ (\vec{v}^{|\mathcal{T}|} \cdot (\vec{v''}^{|\mathcal{T}|} \circ \vec{\psi}^{\mathcal{T}})^\top) \\ &- \vec{1}^{|\mathcal{T}|} \cdot \left(-\mathbf{E}_{\mathcal{T}} \circ ((\vec{v}^{|\mathcal{T}|} \circ \vec{\psi}^{\mathcal{T}}) \cdot (\vec{v''}^{|\mathcal{T}|})^\top) \right).\end{aligned}$$

In total the "noise" introduced by amounts and hiding factors of the commitments in the comparison operations sums up to $\phi_{\mathsf{ty}} = \phi'_{\mathsf{ty}} + \phi''_{\mathsf{ty}}$ and $\eta_{\mathsf{ty}} = \eta'_{\mathsf{ty}} + \eta''_{\mathsf{ty}}$.

To capture this as parameters of $\mathcal{L}_{\mathsf{ebp}}$, we use the preliminary commitment phase which allows the prover to provide $\phi_{\mathsf{ty}}, \eta_{\mathsf{ty}}$ after receiving the challenge variables v, v', v'' but *before* u. Even if the values $\phi_{\mathsf{ty}}, \eta_{\mathsf{ty}}$ are maliciously crafted, they do not interfere with the comparison of types as the exponent of F is fixed to 0 and we assume that the pairwise discrete logarithm between G, F, H holds. As the preliminary commitment is used for the amounts too, we will discuss it in detail when combining all parts in Section 4.5.5.

$$\begin{aligned}
\Lambda^{\mathsf{gr}} &:= (\ G \,\|\, H \,\|\, \vec{C}_{\mathcal{S}} \ \|\ \vec{C}_{\mathcal{T}}\ );\ \vec{c}_R = \vec{0}^{2+|\mathcal{S}|+|\mathcal{T}|+|\mathcal{S}||\mathcal{T}|+|\mathcal{T}||\mathcal{T}|} \\
\vec{c}^{\mathsf{gr}}_L &:= (\phi_{\mathsf{ty}} \| \eta_{\mathsf{ty}} \| \vec{\tilde{e}}_{\mathsf{ty},\mathcal{S}} \| \vec{\tilde{e}}'_{\mathsf{ty},\mathcal{T}} + \vec{\tilde{e}}''_{\mathsf{ty},\mathcal{T}} \| \quad \mathsf{vec}(\mathbf{E}_{\mathcal{S}}) \quad \| \quad \mathsf{vec}(\mathbf{E}_{\mathcal{T}}) \quad) \\
\vec{c}^{\mathsf{gr}}_R &:= (\ \cdot \quad \cdot \quad \cdot \quad \cdot \ \| \ \mathsf{vec}(\mathbf{E}_{\mathcal{S}}) - \vec{1}^{|\mathcal{S}||\mathcal{T}|} \ \| \mathsf{vec}(\mathbf{E}_{\mathcal{T}}) - \vec{1}^{|\mathcal{T}||\mathcal{T}|}) \\
\vec{\mathrm{v}}^{\mathsf{gr}}_1 &:= (\ \cdot \quad \cdot \quad -\vec{y}^{|\mathcal{S}|} \quad \cdot \quad \vec{v'}^{|\mathcal{T}|} \otimes (\vec{v}^{|\mathcal{S}|} \circ \vec{y}^{|\mathcal{S}|}) \quad \cdot \quad) \\
\vec{\mathrm{v}}^{\mathsf{gr}}_2 &:= (\ \cdot \quad \cdot \quad \cdot \quad \vec{y}^{|\mathcal{T}|} \quad \|(\vec{v'}^{|\mathcal{T}|} \circ \vec{y}^{|\mathcal{T}|}) \otimes \vec{v}^{|\mathcal{S}|}\| \quad \vec{v}^* \quad) \\
\vec{\mathrm{v}}^{\mathsf{gr}}_3 &:= (\ \cdot \quad \cdot \quad \cdot \quad \cdot \quad \vec{1}^{\mathcal{T}} \otimes \vec{y}^{|\mathcal{S}|} \quad \cdot \quad) \\
\vec{\mathrm{v}}^{\mathsf{gr}}_4 &:= (\ \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \vec{1}^{\mathcal{T}} \otimes \vec{y}^{|\mathcal{T}|} \quad) \\
\vec{\mathrm{v}}^{\mathsf{gr}}_5 &:= (\ \cdot \quad \cdot \quad \cdot \quad \cdot \quad \vec{y}^{|\mathcal{S}|\cdot|\mathcal{T}|+|\mathcal{T}|^2} \quad)
\end{aligned}$$

with $\vec{v}^* := \vec{v''}^{|\mathcal{T}|} \otimes (\vec{v}^{|\mathcal{T}|} \circ -\vec{y}^{|\mathcal{T}|}) + (\vec{v''}^{|\mathcal{T}|} \circ \vec{y}^{|\mathcal{T}|}) \otimes \vec{v}^{|\mathcal{T}|}$. Similar to the $\vec{\mathrm{v}}^{\mathsf{wf}}_1, \vec{\mathrm{v}}^{\mathsf{wf}}_2$ in the previous section, the constraints $\vec{\mathrm{v}}^{\mathsf{gr}}_1$ and $\vec{\mathrm{v}}^{\mathsf{gr}}_2$ are of class $\mathfrak{dir}$ with $\hat{\mathrm{v}}^{\mathsf{gr}}_0 = \hat{\mathrm{v}}^{\mathsf{gr}}_1 = 0$ to show the correctness of the $\vec{\tilde{e}}_{\mathsf{ty}}$ vectors in the witness at positions 3 to $2 + |\mathcal{S}| + |\mathcal{T}|$. To show that $\mathbf{E}_{\mathcal{S}}$ and $\mathbf{E}_{\mathcal{T}}$ contain a single 1

in each row, the following two constraints $\vec{v}_3^{\text{gr}}, \vec{v}_4^{\text{gr}}$ are of class $\mathfrak{dir}$ with $\hat{v}_3^{\text{gr}} = \langle \vec{1}^{|\mathcal{S}|}, \vec{y}^{|\mathcal{S}|} \rangle$ and $\hat{v}_4^{\text{gr}} = \langle \vec{1}^{|\mathcal{T}|}, \vec{y}^{|\mathcal{T}|} \rangle$. Additionally we require the matrices to be binary, i.e. contain only 0 and 1. We show that equivalently to the original Bulletproof protocol with the following two constraints $\vec{v}_6^{\text{gr}} := \vec{v}_5^{\text{gr}}$, one $\mathfrak{mul}$ and the other $\mathfrak{one}$ with both evaluating to zero $\hat{v}_5^{\text{gr}} = \hat{v}_6^{\text{gr}} = 0$.

4.5.3 *Equality of Amounts*

With the matrices $\mathbf{E}_{\mathcal{S}}, \mathbf{E}_{\mathcal{T}}$ of grouping by type established, we show the amount equality of inputs to outputs within each group of equal type. This is done with the same method as the grouping. The product of all input commitments in a group times the inverse of the output commitments in the same group require an opening to the amount zero, i.e. prove knowledge of exponents for F, H. With a challenge v, all $|\mathcal{T}|$ amount equalities are compressed into the exponent vectors $\vec{e}_{a,\mathcal{S}} = \mathbf{E}_{\mathcal{S}} \circ (\vec{1}^{|\mathcal{S}|} \cdot (\vec{v}^{|\mathcal{T}|})^\top) \cdot \vec{1}^{|\mathcal{T}|}$ and $\vec{e}_{a,\mathcal{T}} = -\mathbf{E}_{\mathcal{T}} \circ (\vec{1}^{|\mathcal{T}|} \cdot (\vec{v}^{|\mathcal{T}|})^\top) \cdot \vec{1}^{|\mathcal{T}|}$ for the input and output commitments.

$$\Lambda^{\text{am}} := (\, F \| H \| \vec{C}_{\mathcal{S}} \| \vec{C}_{\mathcal{T}} \,); \vec{c}_L^{\text{am}} = \vec{0}^{2+|\mathcal{S}|+|\mathcal{T}|+|\mathcal{S}||\mathcal{T}|+|\mathcal{T}||\mathcal{T}|}$$

$$\vec{c}_L^{\text{am}} := (\chi_a \| \eta_a \| \vec{e}_{a,\mathcal{S}} \| \vec{e}_{a,\mathcal{T}} \| \quad \mathsf{vec}(\mathbf{E}_{\mathcal{S}}) \quad \| \quad \mathsf{vec}(\mathbf{E}_{\mathcal{T}}) \quad)$$

$$\vec{v}_1^{\text{am}} := (\, \cdot \quad \cdot \quad -\vec{y}^{|\mathcal{S}|} \quad \cdot \quad \vec{v}^{|\mathcal{T}|} \otimes (\vec{1}^{|\mathcal{S}|} \circ \vec{y}^{|\mathcal{S}|}) \quad \cdot \quad) : \mathfrak{dir}$$

$$\vec{v}_2^{\text{am}} := (\, \cdot \quad \cdot \quad \cdot \quad \vec{y}^{|\mathcal{T}|} \quad \cdot \quad \vec{v}^{|\mathcal{T}|} \otimes (\vec{1}^{|\mathcal{T}|} \circ \vec{y}^{|\mathcal{T}|})) : \mathfrak{dir}$$

with $\hat{v}_0^{\text{am}} = 0$ and $\hat{v}_1^{\text{am}} = 0$. Note that the constraint for the matrices to be binary is already captured by $\vec{v}_5^{\text{gr}}, \vec{v}_6^{\text{gr}}$. The types and randomness for the exponents of $\vec{C}_{\mathcal{S}}$ and $\vec{C}_{\mathcal{T}}$ do not add to zero, so the prover compensates with some exponents of F and H which are fixed by the preliminary commitment with $\chi_a = \Upsilon_a(\mathsf{ty}), \eta_a = \Upsilon_a(r)$ and

$$\begin{aligned} \Upsilon_a(\psi) = & \vec{1}^{|\mathcal{S}|} \cdot -\mathbf{E}_{\mathcal{S}} \circ (\vec{\psi}^{\mathcal{S}} \cdot (\vec{v}^{|\mathcal{T}|})^\top) \cdot \vec{1}^{|\mathcal{T}|} \\ & + \vec{1}^{|\mathcal{T}|} \cdot \mathbf{E}_{\mathcal{T}} \circ (\vec{\psi}^{\mathcal{T}} \cdot (\vec{v}^{|\mathcal{T}|})^\top) \cdot \vec{1}^{|\mathcal{T}|} \end{aligned}$$

4.5.4 *Range proofs*

The amount equivalence above only holds, if all outputs are positive. Therefore, the prover has to show that the output amounts $\vec{a}^{\mathcal{T}}$ lie in the specified range $\{0, \dots, 2^\beta - 1\}$. As in the Bulletproof protocol, we show the range by proving a binary decomposition into β bits. All binary decompositions of the outputs form the matrix $\mathbf{B}$. We specify the partial parameters of $\mathcal{L}_{\text{ebp}}$ for the range:

$$\vec{c}_L^{\text{ra}} := (\, \vec{a}^{\mathcal{T}} \quad \| \quad \mathsf{vec}(\mathbf{B}) \quad); \Lambda^{\text{ra}} = ()$$

$$\vec{c}_R^{\text{ra}} := (\, \vec{0}^{|\mathcal{T}|} \, \| \mathsf{vec}(\mathbf{B}) - \vec{1}^{\beta|\mathcal{T}|})$$

$$\vec{v}_1^{\text{ra}} := (\quad \cdot \quad \vec{y}^{\beta|\mathcal{T}|} \quad) : \mathfrak{mul}, \hat{v}_1^{\text{ra}} = 0$$

$$\vec{v}_2^{\text{ra}} := (\quad \cdot \quad \vec{y}^{\beta|\mathcal{T}|} \quad) : \mathfrak{one}, \hat{v}_2^{\text{ra}} = 0$$

$$\vec{v}_3^{\text{ra}} := (-\vec{y}^{|\mathcal{T}|} \quad \vec{y}^{|\mathcal{T}|} \otimes \vec{2}^\beta \quad) : \mathfrak{dir}, \hat{v}_3^{\text{ra}} = 0$$

Constraints $\vec{\mathrm{v}}_1^{\mathrm{ra}}$ and $\vec{\mathrm{v}}_2^{\mathrm{ra}}$ serve the equivalent purpose for $\mathbf{B}$ as $\vec{\mathrm{v}}_5^{\mathrm{gr}}, \vec{\mathrm{v}}_6^{\mathrm{gr}}$ do for $\mathbf{E}_{\mathcal{S}}$ and $\mathbf{E}_{\mathcal{T}}$, i.e. that $\mathbf{B}$ is a binary matrix. The last constraint $\vec{\mathrm{v}}_3^{\mathrm{ra}}$ shows that $\mathbf{B}$ is the binary decomposition of all values $\vec{a}^{\mathcal{T}}$. In combination with the well-formedness of the output commitments, all output amounts have a valid range.

4.5.5 *Compression of Parts*

In this section, we compress the four sets of parameters of $\mathcal{L}_{\mathrm{ebp}}$ described above and the preliminary commitment into a single instance of $\mathcal{L}_{\mathrm{ebp}}$.
To compensate for the noise in the equality and amount parts, we need to commit to the four values $\phi_{\mathsf{ty}}, \eta_{\mathsf{ty}}, \chi_a, \eta_a$. Before receiving the challenge u, the prover gets five bases $\vec{K}'_G$ from the verifier and creates the commitment $K' = \prod \vec{K}'^{\circ(\phi_{\mathsf{ty}},\eta_{\mathsf{ty}},\chi_a,\eta_a,r_\kappa)}_G$ with randomness r_κ. The well formedness of K' is captured by more $\mathcal{L}_{\mathrm{ebp}}$ parameters: $\Lambda^{\mathrm{pc}} = (K' \| \vec{K}_G^{-1}), \vec{c}_L^{\mathrm{pc}} = (1 \| \phi_{\mathsf{ty}}, \eta_{\mathsf{ty}}, \chi_a, \eta_a, r_\kappa), \vec{c}_R^{\mathrm{pc}} = \vec{0}^6$ and one $\mathfrak{dir}$ constraint $\vec{\mathrm{v}}_1^{\mathrm{pc}} = (1 \| \vec{0}^5)$ with $\hat{\mathrm{v}}_1^{\mathrm{pc}} = 1$ assuring that $\vec{c}_L^{\mathrm{pc}}[1] = 1$.
Once the commitment K' is submitted to the verifier, the prover gets the challenge u to compress all five public functions Λ. We build a polynomial in u where well-formedness is u^0, type grouping u^1, amount equality u^2, range proofs u^3, although the range proof has no elements ($\Lambda^{\mathrm{ra}} = ()$), and well-formedness of K' is u^4. We start by creating a common Λ as the union of all elements required. We raise K' and the bases directly to u^4 to reduce the complexity of the encoded witness resulting in:

$$\Lambda := (G \| F \| H \| K'^{u^4} \| \quad \vec{G}_K^{-u^4} \quad \| \vec{C}_{\mathcal{S}} \| \vec{C}_{\mathcal{T}})$$

$$\vec{c}_L := (\phi \| \chi \| \eta \| \ 1 \ \| \phi_{\mathsf{ty}} \| \eta_{\mathsf{ty}} \| \chi_a \| \eta_a \| r_\kappa \| \vec{e}_{\mathcal{S}} \| \vec{e}_{\mathcal{T}} \| \vec{a}^{\mathcal{S}} \| \vec{a}^{\mathcal{T}} \| \vec{\mathsf{ty}}^{\mathcal{S}} \| \vec{\mathsf{ty}}^{\mathcal{T}} \| \vec{r}^{\mathcal{S}} \| \vec{r}^{\mathcal{T}} \| \vec{c}_{L,2})$$

$$\vec{c}_R := (\cdot \ \cdot \ \cdot \quad \cdot \quad \cdot \quad \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \| \vec{c}_{R,2})$$

$$\vec{c}_{L,2} := (\mathsf{vec}(\mathbf{E}_{\mathcal{S}}) \| \mathsf{vec}(\mathbf{E}_{\mathcal{T}}) \| \mathsf{vec}(\mathbf{B}))$$

$$\vec{c}_{R,2} := (\ \vec{c}_{L,2} - \vec{1}^{|\mathcal{S}|\cdot|\mathcal{T}|+|\mathcal{T}|^2+|\mathcal{T}|\beta}\ )$$

with

$$\begin{aligned}
\phi &= \phi_0 + u\phi_{\mathsf{ty}} \\
\chi &= \chi_0 \quad\quad + u^2\chi_a \\
\eta &= \eta_0 + u\eta_{\mathsf{ty}} + u^2\eta_a \\
\vec{e}_{\mathcal{S}} &= \quad u\vec{e}_{\mathsf{ty},\mathcal{S}} + u^2\vec{e}_{a,\mathcal{S}} \\
\vec{e}_{\mathcal{T}} &= \quad u\vec{e}_{\mathsf{ty},\mathcal{T}} + u^2\vec{e}_{a,\mathcal{T}}
\end{aligned}$$

The constraint vectors of the five parts are adapted by repositioning elements to match the new witness structure and, where necessary, multiplied with a power of u.

4.6 EVALUATION

To corroborate our claim of a smaller typed conservation NIZK proof, we evaluate our construction and compare it to existing solutions, namely the original

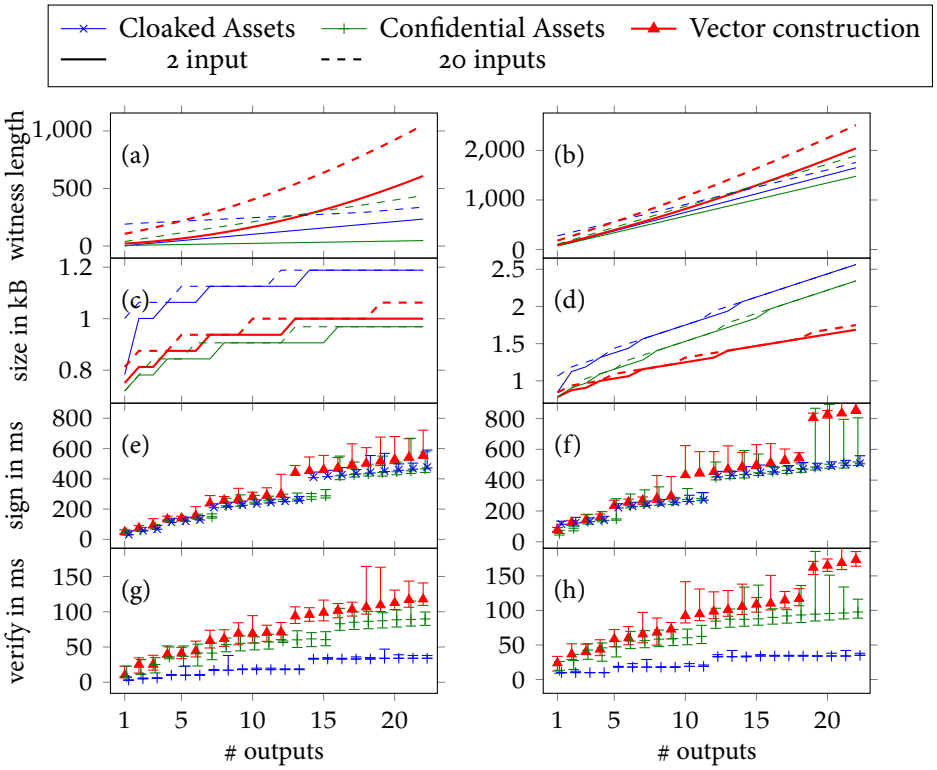


Figure 4.2: Size comparison of type conservation proofs. We compare CL (blue) and our CO (green) and succinct construction (red). We plot dependencies by the number of outputs and show the difference of 2 (solid) and 20 (dashed) inputs with the line type. While we plot lines, the sizes are only defined at integer positions. (a) shows the witness length without the range proof, (b) includes the range proof for 64 bit. (c) shows the size in Bytes of the resulting proof transcript and (d) the transcript plus the size of the outputs. The following plots show the generation time of the implementations, (e) for 2 inputs, (f) for 20 inputs. Similarly (g) and (h) show the verification time for 2 and 20 inputs.

confidential assets and cloaked assets. We assume group and field elements are of the same size, i.e. 32 Bytes. For m inputs and n outputs, let

$$\begin{cases} s = 2, c = 11 \text{ if } m \cdot n = 1 \wedge m + n > 2 \\ s = 5, c = 11 \text{ if } m = n = 1 \\ s = 0, c = 14 \text{ otherwise.} \end{cases}$$

Our typed homomorphic commitment proof transcript is $7 + 2 + 2\lceil \log_2(2 + m + n + mn + 64n) \rceil$ elements and a cloaked asset one is $2 + 2\lceil \log_2(11(n + m - 2) - 5(\min(m, n) - 2) - s + 64n) \rceil + c$ elements in size. Both require output commitments with a total size of $2n$. With vector Pedersen commitments, we achieve a transcript with $7 + 2 + 1 + 2\lceil \log_2(4 + 5 + 4(n + m) + n \cdot m + n^2 + 64n) \rceil$ elements and merely n elements for our succinct commitments.

As all three approaches follow the general Bulletproof structure, we compare their internal structure. All witness vectors include a 64-bit range proof for each output. Figure (4.2a) shows the witness length without the range proof, which in our construction is quadratic in m and n. The impact of the quadratic length on the total length of the witness is minimal as most transactions have few inputs and outputs and the length is dominated by the range proofs included in Figure (4.2b). Applying the logarithmic compression results in very similar transcript sizes shown in Figure (4.2c). Padding to powers of two cause the steps in size. The offsets between the systems stem from the different number of constant communication required, i.e. CL requires 14 elements and CO 9. In our protocol, the additional intermediate commitment K' requires an additional element to a total of 10. Regarding the transcript itself, our approach is slightly larger than the CO proof as it utilizes the type homomorphic property. However, conservation proofs are usually part of a transaction where inputs are references to previous transaction outputs and only the outputs are stored with the transaction. Including the size of the outputs in our consideration in Figure (4.2d), as most transaction systems do, we achieve an improved scaling due to the 32 B succinct commitments.
While our focus is on the size, we provide a proof of concept implementation to show the performance of our systems. Implementations are available for our improved confidential asset proof[2] for cloaked assets[3] and for our vector Pedersen scheme[4]. All use rust and were measured on a ThinkPad T460p running kubuntu 21.04 with kernel 5.11 and rust 1.48 on an i7-6820HQ processor. The signature generation (4.2e for 2 inputs, 4.2f for 20 inputs) is similar in all systems is linear in the length of the witness (4.2b) including padding to a power of two. The verification time (4.2g for 2 inputs, 4.2h for 20 inputs) behaves similarly. We notice that the verification of our vector and confidential asset scheme is significantly slower than cloaked assets. This is due to the structure of the extended Bulletproof protocol where the generators to commit the witness vectors are dynamically generated instead of relying on reusable elements.

4.7 ANALYSIS

Both constructions are parametrizations of the bulletproof structure. For both construction, it holds that:

Theorem 4.1. *The protocol has a public coin verifier $\mathcal{V}$ with logarithmic rounds and is perfectly complete and special honest verifier zero knowledge.*

Theorem 4.2. *Given the discrete logarithm assumption holds in $\mathcal{G}$, the protocol has computational witness-extended emulation. Additionally, given computationally unique responses, it is transformable to perfectly complete, extractable,*

2 https://github.com/SwapCT/SwapCT
3 https://github.com/stellar/slingshot/tree/main/spacesuit/
4 https://anonymous.4open.science/r/coloring-EEC3

perfectly simulatable signature of knowledge for any message $m \in \{0,1\}^*$ using Fiat-Shamir [FS] which holds for logarithmic rounds [DFM20, Thm. 23].

4.8 SECURITY PROOFS

As the THC construction did not provide a proof for the updatability, we provide an explicit one.

Proof of Theorem 2.1 (Update). We show that THC is binding according to Definition 2.15. Given a discrete log challenge $\mathsf{chl} = (G, G^\gamma) \in \mathbb{G}^2$, we define an oracle H^O for the adversary to use. Sample a secret $\mathsf{sk} \xleftarrow{\$} \{0,1\}^\lambda$ and define a new hash function $\mathfrak{h} : \{0,1\}^* \to \mathbb{Z}_q$. On input of i, calculate a bit $b = \mathfrak{h}(i \| \mathsf{sk}) \mod 2$ and return $(G^\gamma)^{\mathfrak{h}(i\|\mathsf{sk})}$ if $b = 0$ and $G^{\mathfrak{h}(i\|\mathsf{sk})}$ otherwise. The output is indistinguishable from a uniformly random distribution over $\mathbb{G}$. Assume that an efficient adversary $\mathcal{A}$ exists, which returns

$$i, i', v, v', (r,s), (r',s') \leftarrow \mathcal{A}_{\mathsf{H}^O}(\mathsf{pp})$$

for which

$$\mathsf{Commit}(\mathsf{ty}, v; r, s) = \mathsf{Commit}(\mathsf{ty}', v'; r', s') \wedge (i \neq i' \vee (i = i' \wedge v \neq v'))$$

holds with $\mathsf{ty} = \mathsf{ComTypeGen}(i)$ and $\mathsf{ty}' = \mathsf{ComTypeGen}(i')$.
This is equal to $\mathsf{H}^O(i)G^r = \mathsf{H}^O(i')G^{r'}$ and $\mathsf{H}^O(i)^v G^{rv+s} = \mathsf{H}^O(i')^{v'} G^{r'v'+s'}$.
We have two cases:

FOR $i = i'$ the pre-image is equal, so $v \neq v'$ is true and $v - v' \neq 0$. Then $H^O(i)^{v-v'} = G^{r'v'+s'-rv-s}$. In $\frac{1}{2}$ of the executions, $\mathfrak{h}(i\|\mathsf{sk}) \equiv 0 \mod 2$, and thereby $(G^\gamma)^{v-v'} = G^{r'v'+s'-rv-s}$ from which we return $\gamma = \frac{r'v'-rv}{v-v'}$ to with the challenge.

FOR $i \neq i'$ with different identifiers, it holds with $\frac{1}{2}$ probability that $\mathfrak{h}(i\|\mathsf{sk}) \not\equiv \mathfrak{h}(i'\|\mathsf{sk}) \mod 2$. Without loss of generality, assume $\mathsf{H}^O(i) = G^{\mathfrak{h}(i\|\mathsf{sk})}$ and $\mathsf{H}^O(i') = G^{\gamma\mathfrak{h}(i'\|\mathsf{sk})}$. From

$$\left(G^{\mathfrak{h}(i\|\mathsf{sk})}\right)^v G^{rv+s} = \left(G^{\gamma\mathfrak{h}(i'\|\mathsf{sk})}\right)^{v'} G^{r'v'+s'}$$

we calculate $\gamma = \frac{v\mathfrak{h}(i\|\mathsf{sk})+rv+s-r'v'-s'}{v'\mathfrak{h}(i'\|\mathsf{sk})}$ and return γ to solve the challenge.

In both cases, independent of the adversary's choice of i, i', we have $\frac{1}{2} > \mathsf{negl}(\lambda)$ chance to solve the discrete logarithm challenge. Therefore we conclude that no efficient adversary against the binding property exists.
As THC is binding, it is also securely updatable according to Theorem 2.1. We show that from an efficient adversary $\mathcal{A}$ against the update Theorem 2.1, we can derive an efficient Adversary for the binding property of Def. 2.15 which proceeds as follows: Sample $i \xleftarrow{\$} \{0,1\}^*$ and $v, r, s, v', r', s' \xleftarrow{\$} \mathbb{Z}_q$ with $v \neq v'$. Then commit $(T, V) = \mathsf{Commit}(\mathsf{ComTypeGen}(i), v; r, s)$ and

$(T', V') = \mathsf{Commit}(\mathsf{ComTypeGen}(i), v'; r', s')$. Invoke the adversary to get a ϕ_2 for which $V \cdot V'^{-1} = G^{\phi_2}$ holds. From this calculate the discrete logarithm of $\mathsf{H}(i)$ to base G as $\frac{\phi_2 - vr - s + v'r' + s'}{v - v'}$ from which an efficient adversary against the binding property is easily constructed. □

Proof of Theorem 4.1 (Simulatability). By inspection, the protocol is public coin, has logarithmic rounds and is complete. It is special honest verifier zero knowledge with an efficient simulator Sim similar to the one of Omniring. Given a statement $\vec{C}_{\mathcal{S}}, \vec{C}_{\mathcal{T}}$ and the public coins

$$(\vec{G}_K, D, \vec{P}, \vec{G}', \vec{H}, v, v', v'', u, w, x, y, z)$$

- Sim samples $K' \xleftarrow{\$} \mathbb{G}$
- computes $\vec{G}_w$ as the verifier
- samples random $A, T_2, \tau, r, \vec{l}, \vec{r}$
- sets $t = \langle \vec{l}, \vec{r} \rangle$.

To satisfy the verification equations, Sim calculates

$$S = (D^{-r} A \vec{G}_w^{\vec{\alpha} - \vec{l}} \vec{H}^{\vec{\beta} - \vec{\theta}^{\circ -1} \circ \vec{r}})^{-1/x}$$

and

$$T_1 = (G^{\delta - t} D^{-\tau} T_2^{x^2})^{-1/x}$$

and outputs $(K', A, S, T_1, T_2, \tau, r, \vec{l}, \vec{r}, t)$ □

The extractability of the Omniring protocol holds as we have not changed the protocol after $\vec{G}_w$ is calculated. Thereby we are sure to get an extracted witness $\vec{c}'_L, \vec{c}'_R$ which given the constraints is correctly encoded. It remains to show for Theorem 4.2 that the addition of the preliminary commitment does not impact the security. For a grid of subvector comparisons of vector Pedersen commitment pairs, either the subvector equality holds or there is a reduction to an adversary against the discrete logarithm assumption. We exemplarily show the proof for type grouping but it holds equivalently for amount comparisons.

Lemma 4.1 (Type Equality)**.** *Two typed commitments are of equal type, if for any PPT adversary $\mathcal{A}$ and all* $\mathsf{pp} \leftarrow \mathsf{TC.Setup}(\lambda)$*, it holds that*

$$\Pr\left[\begin{array}{l|r} \forall i \in \{0,1\} : C_b = G^{a_b} F^{\mathsf{ty}_b} H^{r_b} & (\{a_b, \mathsf{ty}_b, r_b\}_{b=0}^1, \phi, \eta) \\ C_0 \cdot C_1^{-1} = G^{\phi} H^{\eta} \wedge \mathsf{ty}_0 \neq \mathsf{ty}_1 & \leftarrow \mathcal{A}(\mathsf{pp}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Proof. Given a discrete logarithm challenge (B, B^{γ}), we prepare the public parameters such that $F := B^{\gamma}, G := B$ and for a uniformly random $x \xleftarrow{\$} \mathbb{Z}_q$ set $H := B^x$. From an efficient adversary $\mathcal{A}$, we receive $(\{a_b, \mathsf{ty}_b, r_b\}_{b=0}^1, \phi, \eta)$ for which

$$G^{a_0} F^{\mathsf{ty}_0} H^{r_0} \cdot G^{-a_1} F^{-\mathsf{ty}_1} H^{-r_1} = G^{\phi} H^{\eta}$$

and $\mathsf{ty}_0 \neq \mathsf{ty}_1$ holds. With our knowledge of the setup, this equation translates to

$$B^{a_0+\gamma \mathsf{ty}_0+xr_0-a_1-\gamma \mathsf{ty}_1-xr_1} = B^{\phi-x\eta}$$

and $\gamma = \frac{\phi-x\eta-a_0-xr_0+a_1+xr_1}{\mathsf{ty}_0-\mathsf{ty}_1}$. This equation is well defined as $\mathsf{ty}_0 - \mathsf{ty}_1 \neq 0$ because $\mathsf{ty}_0 \neq \mathsf{ty}_1$, solving the discrete logarithm challenge. □

Lemma 4.2 (Type Group Equality)**.** *Type equality holds for all input output commitment relations indicated by* 1 *in a binary matrix* **E***, if for any PPT adversaries* $\mathcal{A}_1.\mathcal{A}_2$ *and all* $\mathsf{pp} \leftarrow \mathsf{TC.Setup}(\lambda)$*, it holds that*

$$\Pr\left[\begin{array}{rr} \forall i \in [|\mathcal{S}|] : C_{\mathcal{S}}[i] = G^{\vec{a}^{\mathcal{S}}[i]} F^{\vec{\mathsf{ty}}^{\mathcal{S}}[i]} H^{\vec{r}^{\mathcal{S}}[i]} & (\mathsf{aux}, \mathcal{S}, \mathcal{T}) \leftarrow \mathcal{A}_1(\mathsf{pp}) \\ \forall i \in [|\mathcal{T}|] : C_{\mathcal{T}}[i] = G^{\vec{a}^{\mathcal{T}}[i]} F^{\vec{\mathsf{ty}}^{\mathcal{T}}[i]} H^{\vec{r}^{\mathcal{T}}[i]} & v, v' \xleftarrow{\$} \mathbb{Z}_q \\ \prod_{i=1}^{|\mathcal{T}|} \prod_{j=1}^{|\mathcal{S}|} {v'}^i v^j (C_{\mathcal{S}}[j] C_{\mathcal{T}}[i]^{-1})^{\mathbf{E}[j,i]} = G^{\phi} H^{\eta} & : (\phi, \eta, \mathbf{E}) \leftarrow \mathcal{A}_2(\mathsf{aux}, v, v') \\ & (\vec{a}^{\mathcal{S}}, \vec{\mathsf{ty}}^{\mathcal{S}}, \vec{r}^{\mathcal{S}}) := \mathcal{S} \\ \mathbf{E}[j,i] = 1 \nRightarrow \mathsf{ty}^{\mathcal{S}}[j] = \mathsf{ty}^{\mathcal{T}}[i] & (\vec{a}^{\mathcal{T}}, \vec{\mathsf{ty}}^{\mathcal{T}}, \vec{r}^{\mathcal{T}}) := \mathcal{T} \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Proof. For an adversary $\mathcal{A}$ to have a non-negligible advantage, at least one element of **E** needs to be 1. Different pairs of (i, j) cannot influence each other as they are separated by different powers of the challenge variables ${v'}^i v^j$. Given efficient adversaries $\mathcal{A}_1, \mathcal{A}_2$, we construct an efficient adversary against the single comparison of Lemma 4.1 by outputting a pair $(\vec{a}^{\mathcal{S}}[j], \vec{\mathsf{ty}}^{\mathcal{S}}[j], \vec{r}^{\mathcal{S}}[j])$, $(\vec{a}^{\mathcal{T}}[i], \vec{\mathsf{ty}}^{\mathcal{T}}[i], \vec{r}^{\mathcal{T}}[i])$ where $\mathbf{E}[j,i] = 1$ and $\mathsf{ty}^{\mathcal{S}}[j] = \mathsf{ty}^{\mathcal{T}}[i]$. □

Given the lemmata above, the extracted witness actually satisfies the conservation language.

4.9 CONCLUSION

In this section we presented two asset conservation NIZK protocols based on the Bulletproof structure. For this we adapted the existing protocol such that it supports efficient equality checks of individual positions of vector Pedersen commitments. This saves on expensive zero-knowledge multiplications compared to using a blackbox arithmetic circuit protocol. Compared to the existing asset conservation proofs for type homomorphic commitments, we reduce the proof size from linear to logarithmic in the number of inputs. The resulting small asset conservation proofs provide the central NIZK for the full transaction systems we describe in the next chapters.

II

MULTI-TYPE TRANSACTIONS

COLORING

5.1 OVERVIEW

5

Having the building blocks for a multi-type system, we finally join them together to form an actual system. The goal of the system is to support RingCT-style transactions while hiding an additional type for each output. The conservation rules for this type in a transaction are the same as defined by the asset conservation where now the amount *per type* is equal in the transaction inputs and its outputs. We name this system Coloring. As in Omniring, we use a combined transaction signature, which authorizes the spending of the inputs as well as proves the correct conservation. The authorization part of the signature assures that every input selected from a common, large anonymity set corresponds to a tag published along with the signature. Without a notion of types, the conservation in Omniring just calculates the sum of all selected inputs and proves its equivalency to the sum of all outputs. Our conservation perform this equivalency check for each type involved. Additionally, the union of all used inputs in any of the types need to have matching tags.
To check the equality for each type individually, we use technique from our asset conservation proof and group the equal types first and then check for the sum equality. To hide the number of different types transferred in a transaction, we use the upper bound of possible groups and assume that each output is of a different type, creating its own group. In the example of Figure 5.1, the left type equality group captures all blue inputs and the two blue outputs. The middle equality compares the red top input and red bottom output. The last comparison group is empty, as all types present in the transaction are already handled because two outputs were of the same color (blue).
We cannot have three separated conservation proofs, as the authorization signature reveals the tags for the used input. This discloses the number of inputs of the type in consideration. To keep the composition of input and output types confidential, we first aggregate the used inputs from each type and then provide the authorization signature for the subset of all used inputs together.

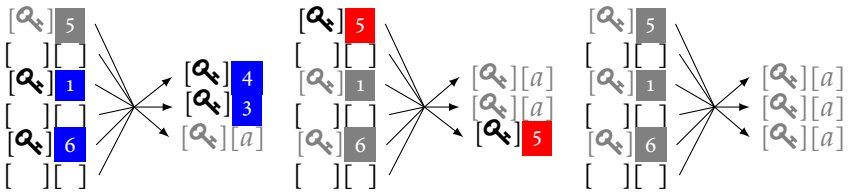


Figure 5.1: Multi-Type RingCT construction showing the conservation for all three possible types involved. The left equality is for all blue inputs and outputs, the middle for all red inputs and outputs and the right one is empty as there is no third type.

Thereby the tags can no longer be matched to a type or leak the number of inputs per type.
Using succinct vector Pedersen commitments, we enable storing additional attributed for our transaction outputs. We showcase this by including a time lock. An output is only spendable after the time, specified in the time lock, passed. This time lock remains confidential and a zero knowledge proof ensures that the time lock has passed compared to a publicly provided input date. Using the efficient Bulletproof protocol from our asset conservation arguments to compare the inputs, our multi-type extension adds minimal overhead to Omniring transactions. For typical numbers of inputs and outputs, the size overhead is less than 10% and the runtime is approximately twice as slow to manage all constraints on the additional two attributes per output instead of a simple sum of all real inputs.

5.2 FORMALIZATION

Regarding the formalization, a multi-type RingCT system is a standard RingCT system, where each amount is replaced by a tuple (a, ty).

Definition 5.1. *A privacy-preserving multi-type RingCT system consists of the following algorithms which are the extension of Omniring:*

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\alpha, 1^\beta)$ *takes the security parameter λ and integers α for a maximum of 2^α outputs of a transaction where each has an amount maximum of $2^\beta - 1$. Then it outputs public parameters* pp *which are implicitly given to all the following algorithms. Setup is called once when a Multi-type RingCT system is initialized.*

$(\mathsf{ltp}, \mathsf{lts}) \leftarrow \mathsf{KeyGen}()$ *generates a long term secret key* lts *with the corresponding long term public key* ltp *for participants to initially join the system. The* ltp *is distributed and serves as a recipient address.*

$\mathsf{ty} \leftarrow \mathsf{TypeGen}(\mathsf{name})$ *generates a type* ty *given a* name.

$\mathsf{acc}, \mathsf{ck} \leftarrow \mathsf{OTGen}(\mathsf{ltp}, a, \mathsf{ty})$ *creates a one-time account* acc *with coin key* ck *from a long term public key* ltp *and an amount a of a type* ty *to then use this account as an output in an offer or a new type registration.*

$\mathsf{t} \leftarrow \mathsf{Spend}(\mathcal{R}, \mathcal{S}, \mathcal{T})$: *takes as input*

- *a set of ring inputs* $\mathcal{R} = \{\mathsf{acc}_i^{\mathcal{R}}\}_{i=1}^{|\mathcal{R}|}$.
- *a set of inputs* $\mathcal{S} = \{a_i^{\mathcal{S}}, \mathsf{ty}_i^{\mathcal{S}}, \mathsf{ck}_i^{\mathcal{S}}, \mathsf{sk}_i, j_i\}_{i=1}^{|\mathcal{S}|}$ *with amounts* $a_i^{\mathcal{S}}$ *of typ* $\mathsf{ty}_i^{\mathcal{S}}$ *with coin key* $\mathsf{ck}_i^{\mathcal{S}}$ *and secret key* sk_i *for the account* $\mathsf{acc}_{j_i}^{\mathcal{R}}$
- *a set of outputs* $\mathcal{T} = (\mathsf{ck}_i^{\mathcal{T}}, a_i^{\mathcal{T}}, \mathsf{ty}_i^{\mathcal{T}}, \mathsf{acc}_i^{\mathcal{T}})$, *defined by their amount* $a_i^{\mathcal{T}}$ *of type* $\mathsf{ty}_i^{\mathcal{T}}$ *hidden by coin key* $\mathsf{ck}_i^{\mathcal{T}}$ *in the account* $\mathsf{acc}_i^{\mathcal{T}}$

The algorithm outputs a signature t *for the transaction.*

$b \leftarrow \mathsf{VfTx}(\mathsf{tx}, \mathsf{t})$ *takes a transaction defined as*

$$\mathsf{tx}(\mathcal{R}, \mathcal{S}, \mathcal{T}) := \left(\{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{acc}_i^{\mathcal{R}}\}_{i=1}^{|\mathcal{R}|}, \{\mathsf{acc}_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}\right)$$

and the signature t *and returns a bit* b *representing the validity.*

$(\mathsf{tag}, \mathsf{sk}, a, \mathsf{ty}, \mathsf{ck}) \leftarrow \mathsf{Receive}(\mathsf{acc}, \mathsf{lts})$ *gets an account* acc *and a long term secret* lts *and returns the matching* tag*, secret key* sk*, amount* a*, type* ty *and coin key* ck *for* acc*.*

Further, we require the following two auxiliary algorithms to define the security properties.

$b \leftarrow \mathsf{ChkAcc}(\mathsf{acc}, a, \mathsf{ty}, \mathsf{ck})$ *takes an account* acc*, an amount* a *with type* ty *and a coin key* ck *and checks if they are consistent.*

$b \leftarrow \mathsf{ChkTag}(\mathsf{acc}, \mathsf{tag}, \mathsf{sk})$ *takes an account* acc*, a* tag *and a secret key* sk *and returns* 1 *if consistent,* o *otherwise.*

The system has to fulfill the following correctness criteria:

Definition 5.2 (Correctness)**.** *A Multi Color RingCT scheme is correct, if for all* $\lambda, \alpha, \beta \in \mathbb{N}$ *and all* $\mathsf{pp} \in \mathsf{Setup}(1^\lambda, 1^\alpha, 1^\beta)$*:*

- *Honestly generated payments are received correctly: For any* $\mathsf{ltp}, \mathsf{lts} \in \mathsf{KeyGen}()$*, any* $\mathsf{name} \in (0,1)^*, \mathsf{ty} = \mathsf{TypeGen}(\mathsf{name})$*, any amount* $a \in \{0, \dots, 2^\beta - 1\}$*, any* $(\mathsf{acc}, \mathsf{ck}) \in \mathsf{OTGen}(\mathsf{ltp}, a, \mathsf{ty})$*, and any* $(\cdot, a', \mathsf{ty}', \mathsf{ck}') \in \mathsf{Receive}(\mathsf{acc}, \mathsf{lts})$*, it holds that* $(a, \mathsf{ty}, \mathsf{ck}) = (a', \mathsf{ty}', \mathsf{ck}')$*.*
- *Honestly received payments have a valid amount, type and tag: For any* $(\mathsf{tag}, \mathsf{sk}, a, \mathsf{ty}, \mathsf{ck}) \in \mathsf{Receive}(\mathsf{acc}, \mathsf{lts})$*, it holds that* $\mathsf{ChkAcc}(\mathsf{acc}, a, \mathsf{ty}, \mathsf{ck}) = 1$ *and* $\mathsf{ChkTag}(\mathsf{acc}, \mathsf{tag}, \mathsf{sk}) = 1$*.*
- *Honestly generated transactions are valid: For each* $\mathcal{S}, \mathcal{R}, \mathcal{T}$*, defined as above, that satisfy*
  - $\forall i \in [|\mathcal{S}|], \mathsf{ChkTag}(\mathsf{acc}_{j_i}^{\mathcal{R}}, \mathsf{tag}_i, \mathsf{sk}_i) = 1$
  - $\forall i \in [|\mathcal{S}|], \mathsf{ChkAcc}(\mathsf{acc}_{j_i}^{\mathcal{R}}, a_i^{\mathcal{S}}, \mathsf{ty}_i^{\mathcal{S}}, \mathsf{ck}_i^{\mathcal{S}}) = 1$
  - $\forall i \in [|\mathcal{T}|], \mathsf{ChkAcc}(\mathsf{acc}_i^{\mathcal{T}}, a_i^{\mathcal{T}}, \mathsf{ty}_i^{\mathcal{T}}, \mathsf{ck}_i^{\mathcal{T}}) = 1$
  - $|\mathcal{T}| \leq 2^\alpha$
  - $\forall i \in [|\mathcal{T}|] : a_i^{\mathcal{T}} \in \{0, \dots, 2^\beta - 1\}$
  - $\forall \mathsf{ty} \in \{\mathsf{ty}_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|} : \sum\{a_i^{\mathcal{S}} | \mathsf{ty}_i^{\mathcal{S}} = \mathsf{ty}\}_{i=1}^{|\mathcal{S}|} = \sum\{a_i^{\mathcal{T}} | \mathsf{ty}_i^{\mathcal{T}} = \mathsf{ty}\}_{i=1}^{|\mathcal{T}|}$

  and for any signature $\mathsf{t} \in \mathsf{Spend}(\mathcal{S}, \mathcal{R}, \mathcal{T})$*, it holds that* $\mathsf{VfTx}(\mathsf{tx}, \mathsf{t}) = 1$ *with* $\mathsf{tx} = \mathsf{tx}(\mathcal{S}, \mathcal{R}, \mathcal{T})$*.*

5.3 SECURITY & PRIVACY

We require at least the same security and privacy guarantees in a type aware system as Omniring does. The security and privacy experiments are equivalent to the ones described by Omniring up to replacing any amount with an (amount, type, lock time) tuple. Balance is the most basic property a transaction system must satisfy, i.e., no participant must be able to steal tokens from the system or other users.

Algorithm 5.1 $\text{Balance}^{\text{rct}}_{\mathcal{A},\mathcal{E}_\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\alpha, 1^\beta)$
$(\mathsf{tx}, \mathsf{t}) \leftarrow \mathcal{A}(\mathsf{pp})$
$(\mathcal{S}, \mathcal{R}, \mathcal{T}) \leftarrow \mathcal{E}_\mathcal{A}(\mathsf{pp}, \mathsf{tx}, \mathsf{t})$
parse $\mathcal{S}$ as $\left\{\mathsf{tag}_i, j_i, \mathsf{sk}_i, a_i^\mathcal{S}, \mathsf{ty}_i^\mathcal{S}, \mathsf{ck}_i^\mathcal{S}\right\}_{i=1}^{|\mathcal{S}|}$
parse $\mathcal{R}$ as $\{\mathsf{acc}_i^\mathcal{R}\}_{j=1}^{|\mathcal{R}|}$
parse $\mathcal{T}$ as $\left\{\mathsf{acc}_i^\mathcal{T}, a_i^\mathcal{T}, \mathsf{ty}_i^\mathcal{T}, \mathsf{ck}_i^\mathcal{T}\right\}_{i=1}^{|\mathcal{T}|}$
$b_1 := \mathsf{VfTx}(\mathsf{tx}, \mathsf{t})$
$b_2 := \mathsf{tx} = \mathsf{tx}(\mathcal{S}, \mathcal{R}, \mathcal{T})$
$b_3 := \forall i \in [|\mathcal{S}|], \mathsf{ChkTag}(\mathsf{acc}_{j_i}^\mathcal{R}, \mathsf{sk}_i, \mathsf{tag}_i) = 1$
$b_4 := \forall i \in [|\mathcal{S}|], \mathsf{ChkAcc}(\mathsf{acc}_{j_i}^\mathcal{R}, a_i^\mathcal{S}, \mathsf{ty}_i^\mathcal{S}, \mathsf{ck}_i^\mathcal{S}) = 1$
$b_5 := \forall i \in [|\mathcal{T}|], \mathsf{ChkAcc}(\mathsf{acc}_i^\mathcal{T}, a_i^\mathcal{T}, \mathsf{ty}_i^\mathcal{T}, \mathsf{ck}_i^\mathcal{T}) = 1$
$\boxed{b_6 := \forall \mathsf{ty} \in \{\mathsf{ty}_i^\mathcal{T}\}_{i=1}^{|\mathcal{T}|} : \sum\{a_i^\mathcal{S} | \mathsf{ty}_i^\mathcal{S} = \mathsf{ty}\}_{i=1}^{|\mathcal{S}|} = \sum\{a_i^\mathcal{T} | \mathsf{ty}_i^\mathcal{T} = \mathsf{ty}\}_{i=1}^{|\mathcal{T}|}}$
return $b_1 \wedge b_2 \wedge \neg(b_3 \wedge b_4 \wedge b_5 \wedge b_6)$

Definition 5.3 (Balance). *A multi-type RingCT scheme is balanced, if the functions* ChkAcc *and* ChkTag *are binding to the amount and type similar to the definition in Omniring. Additionally, the Omniring balance experiment per type involved in the transaction holds. Specifically, instead of* $b_5 := \sum\{a_i^\mathcal{S}\}_{i=1}^{|\mathcal{S}|} = \sum\{a_i^\mathcal{T}\}_{i=1}^{|\mathcal{T}|}$ *in the Omniring* $\mathsf{Balance}_{\Omega,\mathcal{A},\mathcal{E}_\mathcal{A}}$ *experiment, the equality has to hold per type:*

$$b_5 := \forall \mathsf{ty} \in \{\mathsf{ty}_i\}_{i=1}^{|\mathcal{T}|}, \sum\{a_i^\mathcal{S} | \mathsf{ty}_i^\mathcal{S} = \mathsf{ty}\} = \sum\{a_i^\mathcal{T} | \mathsf{ty}_i^\mathcal{T} = \mathsf{ty}\}$$

More formally, the scheme is balanced, if for all PPT adversaries $\mathcal{A}$ *and all positive integers* $\alpha, \beta \in \mathsf{poly}(\lambda)$*, there exists a PPT extractor* $\mathcal{E}_\mathcal{A}$ *such that*

$$\Pr[\text{Balance}^{rct}_{\mathcal{A},\mathcal{E}_\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta) = 1] \leq \mathsf{negl}(\lambda)$$

with $\text{Balance}^{rct}_{\mathcal{A},\mathcal{E}_\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta)$ *defined in Algorithm 5.1.*

As our resulting protocol has the same privacy and non-slanderability requirements as Omniring, the security experiments are identical up to handling a type, amount tuple wherever an amount is considered. For the time lock, our

security stems directly from the intuition. A PPT adversary wins the time lock experiment, if they are able to spend an output before the block number is reached.
The privacy definition lets an adversary interact with a spending oracle which generates valid transactions for accounts to which the adversary has no secret keys. After the interaction with the oracle, the adversary presents a partial transaction with two possible subsets of uncompromized inputs to spend from and two sets of uncompromized recipients. The security experiment will create one of these two possible transactions and the adversary wins the game if they successfully distinguish which one was created. This captures sender and receiver anonymity as well as amount and type confidentiality. Knowledge of any of them easily distinguishes the two versions of the experiment.

Definition 5.4 (Privacy)**.** *A multi-type RingCT scheme is private, if any PPT adversary has a negligible advantage in the Omniring privacy experiment.*

In a transaction scheme, where tokens are marked as spent, it is important that only the owner can spend the tokens and that no one else can prevent the rightful owner to spend the tokens. This means publishing a tag for someone else must be prevented. The non-slanderability property captures this as an adversary wins the experiment, if they present a valid tag for an account which is uncompromized and was not used by the oracle in a transaction.

Definition 5.5 (Non-Slanderability)**.** *A multi-type RingCT scheme is non-slanderable, if any PPT adversary has a negligible chance of winning the Omniring Non-slanderability experiment.*

5.4 CONSTRUCTION

In this section we describe how to integrate our efficient multi-type conservation NIZK proof into Omniring to support confidential types.
The Omniring transaction signature which satisfies all security and privacy requirements of a privacy-preserving transaction system is similar to a threshold linkable ring signature which authorizes the spending of a hidden subset of all inputs. To prevent double spending and assure that each output is referenced as real spending input exactly once, a signature is only valid, if it is unlinkable to all previous transaction signatures. The linkability is explicitly denoted by a tagging scheme consisting of $\mathsf{TagKGen}$ to get the public key from a secret key and $\mathsf{TagEval}$ to get the tag from the secret key. To maintain key anonymity, the tag is not linkable to the public key without knowledge of the secret key. This double spend protection is reduced to verifying uniqueness of tags, given that the transaction signature assures correct tag to secret key relations.
Formally, the Omniring transaction signature fulfils the language $\mathcal{L}_{\text{omni}}$. The statement consists of the anonymity set $\mathcal{R}$ of inputs referenced with a public key $\mathsf{pk}_i^{\mathcal{R}}$ and a commitment $\mathsf{com}_i^{\mathcal{R}}$. For each used input, there is a corresponding tag_i and regarding the outputs $\mathcal{T}$, only the commitments $\mathsf{com}_i^{\mathcal{T}}$ are relevant, as the recipients public keys are not verified. The witness is a set of

openings $a_i^{\mathcal{S}}, r_i^{\mathcal{S}}$ to each real input commitment $\mathsf{com}_{j_i}^{\mathcal{R}}$ at secret position j_i. The secret keys for the public key pk_{j_i} of the hidden positions are x_i. The output commitment openings $a_i^{\mathcal{T}}, r_i^{\mathcal{T}}$ form the last part of the witness. The language assures that the relevant commitments are well formed, tags correspond to the public keys and tokens are conserved: $\mathcal{L}_{\text{omni}} :=$

$$\begin{cases} \mathsf{stmt} = (\{\mathsf{pk}_i^{\mathcal{R}}, \mathsf{com}_i^{\mathcal{R}}\}_{i=1}^{|\mathcal{R}|}, \{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{com}_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}) \\ \exists \mathsf{wit} = (\{(j_i, x_i, a_i^{\mathcal{S}}, r_i^{\mathcal{S}})\}_{i=1}^{|\mathcal{S}|}, \{a_i^{\mathcal{T}}, r_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|})\}) \text{ s.t.} \\ \sum_{i\in[|\mathcal{S}|]} a_i^{\mathcal{S}} = \sum_{i\in[|\mathcal{T}|]} a_i^{\mathcal{T}} \\ \forall i \in [|\mathcal{S}|] : \begin{cases} \mathsf{pk}_{j_i}^{\mathcal{R}} = \mathsf{TagKGen}(x_i) \\ \mathsf{com}_{j_i}^{\mathcal{R}} = \mathsf{PC.Commit}(a_i^{\mathcal{S}}, r_i^{\mathcal{S}}) \\ \mathsf{tag}_i = \mathsf{TagEval}(x_i) \end{cases} \\ \forall i \in [|\mathcal{T}|] : \begin{cases} \mathsf{com}_i^{\mathcal{T}} = \mathsf{PC.Commit}(a_i^{\mathcal{T}}, r_i^{\mathcal{T}}) \\ a_i^{\mathcal{T}} \in \{0, \dots, 2^\beta - 1\} \end{cases} \end{cases}$$

Our goal is to integrate confidential types into the Omniring transaction system. Given the three asset conservation protocols (CO, CL and ours), our protocol is best suited due to its small transaction sizes: We are not aware of an efficient type homomorphic commitment scheme which requires only one element, thereby we outperform CO by a scaling factor of 2 with a similar transcript size. A succinct type homomorphic commitment scheme can be built with a zero-knowledge hash function, however due to the non-algebraic properties, this is prohibitively expensive in a transparent setup. The CL structure heavily depends on the possibility to shuffle all inputs. To achieve sender set anonymity, CL requires a preliminary step with a threshold ring signature adding unnecessary size in the form of additional rerandomized commitments. In addition, CL depends on two separate commitments which results in the same scaling as CO. Our protocol structure however is naturally adapted to the threshold setting. The fact that our commitments have the same size as the non-type aware Pedersen commitments of Omniring leads to a marginal (≈5-10%) size increase when integrating confidential types compared to the original Omniring.

The vector Pedersen commitment and our subvector comparison enable integrating additional attributes of outputs without increasing the commitment size, e.g. requiring knowledge of an additional secret key or time locks. A time locked output is only usable as input to a transaction after a specific point in time, usually measured in blocks. E.g. a sender creates an output which is only usable by the recipient after block 42000. Using the output as a real input before block 42000 cannot yield a valid transaction signature. Due to the confidentiality, the output may be used as mixin earlier. If there is no constraint on spending, the time lock is set to block 0 and thereby is always spendable. Such time locks are an important step towards enabling layer two systems, such as payment channels [MBL+20; TMSS20] on privacy-preserving ledgers. Pay-

ment channels allow for high frequency low cost transactions while preserving a decentralized network [EKK+17].
Given vector Pedersen commitments (VC) for vectors of length 3, we specify the language for our type aware confidential Omniring extension and mark the changes in boxes. The proof requires the current block height h and the openings to the commitments (a, ty, h). This results in $\mathcal{L}_{\text{coloring}} :=$.

$$
\begin{cases}
\mathsf{stmt} = (\{(\mathsf{pk}_i^{\mathcal{R}}, \mathsf{com}_i^{\mathcal{R}})\}_{i=1}^{|\mathcal{R}|}, \{\mathsf{tag}\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{com}_i^{\mathcal{T}}\}_{i=1}^{\mathcal{T}}, \boxed{h}) : \\
\exists \mathsf{wit} = (\{j_i, \mathsf{sk}_i, a_i^{\mathcal{S}}, \boxed{\mathsf{ty}_i^{\mathcal{S}}, h_i^{\mathcal{S}}}, r_i^{\mathcal{S}}\}_{i=1}^{|\mathcal{S}|}, \{a_i^{\mathcal{T}}, \boxed{\mathsf{ty}_i^{\mathcal{T}}, h_i^{\mathcal{T}}}, r_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}) \text{ s.t.} \\
\forall i \in [|\mathcal{S}|], \begin{cases} \mathsf{pk}_{j_i} = \mathsf{TagKGen}(\mathsf{sk}_i) \\ \mathsf{tag}_i = \mathsf{TagEval}(\mathsf{sk}_i) \\ \boxed{\mathsf{com}_{j_i}^{\mathcal{R}} = \mathsf{VC.Commit}(a_i^{\mathcal{S}}, \mathsf{ty}_i^{\mathcal{S}}, h_i^{\mathcal{S}}; r_i^{\mathcal{S}}); h_i^{\mathcal{S}} < h} \end{cases} \\
\forall i \in [|\mathcal{T}|], \begin{cases} \boxed{\mathsf{com}_i^{\mathcal{T}} = \mathsf{VC.Commit}(a_i^{\mathcal{T}}, \mathsf{ty}_i^{\mathcal{T}}, h_i^{\mathcal{T}}; r_i^{\mathcal{T}})} \\ a_i^{\mathcal{T}} \in \{0, \dots, 2^{\beta} - 1\} \end{cases} \\
\boxed{\forall \mathsf{ty} \in \{\mathsf{ty}_i\}_{i=1}^{|\mathcal{T}|}, \sum \{a_i^{\mathcal{S}} | \mathsf{ty}_i^{\mathcal{S}} = \mathsf{ty}\} = \sum \{a_i^{\mathcal{T}} | \mathsf{ty}_i^{\mathcal{T}} = \mathsf{ty}\}}
\end{cases}
$$

5.5 INSTANTIATION

In this section, we extend our NIZK protocol for $\mathcal{L}_{\emptyset}$ (Section 4.2) to support the additional requirements from $\mathcal{L}_{\text{coloring}}$. In addition to the type conservation, the transactions provide means to keep track of the used inputs. Therefore, we introduce a new $|\mathcal{R}| \times |\mathcal{S}|$ matrix $\mathbf{T}$ which contains $|\mathcal{S}|$ unique unit vectors of length $|\mathcal{R}|$, each indicating a used position in the ring (marked by ① in $\mathcal{L}_{\omega}$ below). E.g., the red input in our example is grouped into the second column of $\mathbf{E}_{\mathcal{S}}$ and the second column of $\mathbf{T}$ specifies that this input is used. The well-formedness for the commitment raises the input ring to the columns of $\mathbf{T}$ and shows a valid opening: $\boxed{1}^0 \cdot \boxed{5}^0 \cdot \boxed{}^0 \cdot \boxed{}^0 \cdot \boxed{3}^1 \cdot \boxed{2}^0 \cdot \boxed{}^0 = \mathsf{Commit}(\mathsf{red}, 3, r)$. We extend the matrix $\mathbf{E}_{\mathcal{S}}$ to a total of $|\mathcal{R}|$ rows to accommodate all input commitments (② in $\mathcal{L}_{\omega}$). Instead of requiring a single 1 in each row of $\mathbf{E}_{\mathcal{S}}$, we require that the rows containing a 1 are represented in $\mathbf{T}$ with a unit vector row too (③ in $\mathcal{L}_{\omega}$). This assured that any input belonging to a type group has a corresponding unit vector in $\mathbf{T}$. Figure 5.2 shows our construction for the same example as in Figure 4.1 but with three additional mixins at positions 3,4 and 7.
We now use the matrix $\mathbf{T}$ to verify the correct tags for each used input. The set of public keys for each input is defined as $\vec{R} = \{\mathsf{pk}_i\}_{i=1}^{|\mathcal{R}|}$ and each used input has a corresponding tag $\vec{T} = \{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|}$ (④ in $\mathcal{L}_{\omega}$). The secret keys x_i of the used inputs are part of the witness as $\vec{x}$ in the same order as the inputs are specified in $\mathbf{T}$ (⑤ in $\mathcal{L}_{\omega}$). Given the matrix $\mathbf{T}$, the well-formedness of the input commitments $\vec{C}_{\mathcal{R}}$ is only required for used inputs (⑥ in $\mathcal{L}_{\omega}$). This is achieved by using the unit vectors of $\mathbf{T}$, selecting each commitment

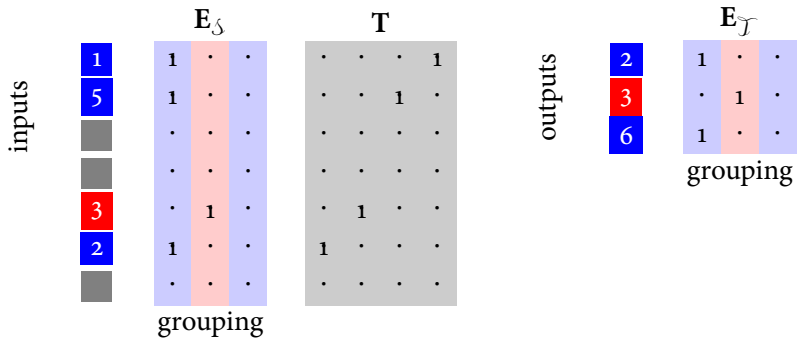


Figure 5.2: Our NIZK proof structure for $\mathcal{L}_{\text{coloring}}$ with 4 inputs, hidden in a ring of 7, and 3 outputs. In the matrices, 0 is substituted with a dot.

separately. Additionally, the private key of the inputs have to match the tag. The tagging scheme instantiation of Omniring uses $\mathsf{TagKGen}(x) := H^x$ and $\mathsf{TagEval}(x) := G^{x^{-1}}$ and for each used input the corresponding tag must be correct (⑦ in $\mathcal{L}_\omega$). We integrate additional, confidential input attributes, like lock times, by extending the vector Pedersen commitment by a position with a new generator $T \in \mathbb{G}$. The well-formedness shows knowledge of a valid opening, including the lock time (⑥ and ⑧ in $\mathcal{L}_\omega$). The type grouping and amount conservation makes use of intermediate committed values τ_{ty} and τ_a respectively to compensate the noise. The lock time check is performed by showing that the difference between the time of the inputs and the current time h ⑨ from a public input is a positive number. The binary representations of the differences for each input are the columns of the matrix $\mathbf{B}'$ (⓪ in $\mathcal{L}_\omega$). The

constraints are analog to the range proof. All properties are summarized by our concrete language $\mathcal{L}_\omega :=$

$$\begin{cases}
\mathsf{stmt} = (④\vec{R}, \vec{T}, \vec{C}_{\mathcal{R}}, \vec{C}_{\mathcal{T}}, ⑨h) \\
\exists \mathsf{wit} = (⑤\vec{x}, \vec{\mathsf{ty}}^{\mathcal{S}}, \vec{a}^{\mathcal{S}}, \vec{r}^{\mathcal{S}}, \vec{h}^{\mathcal{S}} \mathbf{E}_{\mathcal{S}}, \mathbf{E}_{\mathcal{T}}, \mathbf{B}, ①\mathbf{T}, \vec{\mathsf{ty}}^{\mathcal{T}}, \vec{a}^{\mathcal{T}}, \vec{r}^{\mathcal{T}}, \vec{h}^{\mathcal{T}}) \text{ s.t.} \\
\mathbf{B} \text{ binary, size } \beta \times |\mathcal{T}|; \mathbf{B}' \text{ binary, size } 24 \times |\mathcal{S}|; \\
\mathbf{E}_{\mathcal{S}} \text{ binary, size } ②|\mathcal{R}| \times |\mathcal{T}|; \mathbf{E}_{\mathcal{T}} \text{ binary, size } |\mathcal{T}| \times |\mathcal{T}| \\
\quad \forall i \in [|\mathcal{T}|], \begin{cases}
⑧C_{\mathcal{T}}[i] = G^{\vec{a}^{\mathcal{T}}[i]} F^{\vec{\mathsf{ty}}^{\mathcal{T}}[i]} T^{\vec{h}^{\mathcal{T}}[i]} H^{\vec{r}^{\mathcal{T}}[i]} \\
\mathbf{B}[i] \text{ binary repr. of } \vec{a}^{\mathcal{T}}[i] \\
\forall j \in [|\mathcal{R}|] : \vec{C}_{\mathcal{R}}[j]^{\mathbf{E}_{\mathcal{S}}[j,i]} \cdot \vec{C}_{\mathcal{T}}[i]^{-\mathbf{E}_{\mathcal{S}}[j,i]} = G^{\phi_{\mathsf{ty},\mathcal{S},i,j}} F^{0} T^{\tau_{\mathsf{ty},\mathcal{S},i,j}} H^{\eta_{\mathsf{ty},\mathcal{S},i,j}} \\
\forall j \in [|\mathcal{T}|], \vec{C}_{\mathcal{T}}[j]^{\mathbf{E}_{\mathcal{T}}[j,i]} \cdot \vec{C}_{\mathcal{T}}[i]^{-\mathbf{E}_{\mathcal{S}}[j,i]} = G^{\phi_{\mathsf{ty},\mathcal{T},i,j}} F^{0} T^{\tau_{\mathsf{ty},\mathcal{T},i,j}} H^{\eta_{\mathsf{ty},\mathcal{T},i,j}} \\
\prod \vec{C}_{\mathcal{S}}^{\circ \mathbf{E}_{\mathcal{S}}[i]} \cdot \prod \vec{C}_{\mathcal{T}}^{\circ -\mathbf{E}_{\mathcal{T}}[i]} = G^{0} F^{\chi_{a,i}} T^{\tau_{a,i}} H^{\eta_{a,i}}
\end{cases} \\
\mathbf{E}_{\mathcal{S}} \cdot \vec{1}^{|\mathcal{T}|} = ③\mathbf{T} \cdot \vec{1}^{|\mathcal{S}|}; \mathbf{E}_{\mathcal{T}} \cdot \vec{1}^{|\mathcal{T}|} = \vec{1}^{|\mathcal{T}|} \\
\quad \forall i \in [|\mathcal{S}|] : \begin{cases}
⑥\vec{C}_{\mathcal{R}}^{\circ \mathbf{T}[i]} = G^{\vec{a}^{\mathcal{S}}[i]} F^{\vec{\mathsf{ty}}^{\mathcal{S}}[i]} T^{\vec{h}^{\mathcal{S}}[i]} H^{\vec{r}^{\mathcal{S}}[i]} \\
⑦\vec{R}^{\circ \mathbf{T}[i]} = H^{\mathsf{sk}_i}; \vec{T}[i] = G^{\mathsf{sk}_i^{-1}} \\
①\mathbf{T}[i] \text{ unit vector, length } |\mathcal{R}| \\
⓪\mathbf{B}'[i] \text{ bin. repr. of } h - \vec{h}^{\mathcal{S}}[i]
\end{cases}
\end{cases}$$

We proceed to highlight the difference to our construction without an anonymity set. Generally, wherever we used all inputs, we now have to select the used ones with the help of the matrix $\mathbf{T}$.

5.5.1 *Well-formedness of Commitments*

Instead of raising each input commitment in $\vec{C}_{\mathcal{S}}$ to increasing powers of the challenge variable v, we only raise the used commitments of $\vec{C}_{\mathcal{R}}$ to consecutive powers of v with the help of the unit vectors in $\mathbf{T}$.

$$\begin{aligned}
\Lambda^{\mathrm{wf}} &:= (\, G \| F \| T \| H \| \ \vec{C}_{\mathcal{R}} \ \| \ \vec{C}_{\mathcal{T}} \,) \\
\vec{c}_L^{\,\mathrm{wf}} &:= (\phi_0 \| \chi_0 \| \tau_0 \| \eta_0 \| \ \vec{\tilde{v}} \ \| v^{|\mathcal{S}|} \vec{v}^{|\mathcal{T}|} \| \vec{a}^{\mathcal{T}} \| \vec{\mathsf{ty}}^{\mathcal{S}} \| \vec{\mathsf{ty}}^{\mathcal{T}} \| \vec{t}^{\mathcal{S}} \| \vec{t}^{\mathcal{T}} \| \vec{r}^{\mathcal{S}} \| \vec{r}^{\mathcal{T}} \ \| \ \mathsf{vec}(\mathbf{T}) \,) \\
\vec{\mathrm{v}}_4^{\mathrm{wf}} &:= (\, \cdot \ \cdot \ \cdot \ \cdot \ -\vec{y}^{|\mathcal{R}|} \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \vec{v}^{|\mathcal{S}|} \otimes \vec{y}^{|\mathcal{R}|} \,) \\
\vec{\mathrm{v}}_5^{\mathrm{wf}} &:= (\, \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \vec{y}^{|\mathcal{T}|} \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \,) \\
\vec{\mathrm{v}}_6^{\mathrm{wf}} &:= (\, \cdot \ \cdot \ 1 \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \cdot \ \vec{v}^{|\mathcal{S}|+|\mathcal{T}|} \ \cdot \,)
\end{aligned}$$

with $\vec{\tilde{v}} := \mathbf{T}\vec{v}^{|\mathcal{S}|}$ and $\vec{\mathrm{v}}_1^{\mathrm{wf}}$ to $\vec{\mathrm{v}}_3^{\mathrm{wf}}$ equivalent to Section 4.5.1. All constraints are $\mathfrak{dir}$ and $\hat{\mathrm{v}}_i^{\mathrm{wf}} = 0$ for $i \in \{1,2,3,4,6\}$ and $\hat{\mathrm{v}}_5^{\mathrm{wf}} = \sum_{i=1}^{|\mathcal{T}|} v^{|\mathcal{S}|+i} y^i$.
We notice that $\mathbf{T}$ has to consist of unit vectors. We add the constraints for this in Section 5.5.3 with the correctness for tags and keys.

5.5.2 *Grouping of Equal Types and Equality of Amounts*

In our grouping of types and equality of amounts, we select inputs and outputs according to the positions of ones in the matrix $\mathbf{E}_{\mathcal{S}}$. This is unaffected by additional rows of zeros and it suffices to adapt the constraints which iterate over all inputs from length $|\mathcal{S}|$ to $|\mathcal{R}|$. To compensate the noise introduced by the time locks, the prover commits to additional values $\tau_{\mathsf{ty}}, \tau_a$ in the preliminary commitment K'. The range proofs are completely unaffected by the input commitments.

5.5.3 *Correctness of Tags and Keys*

As the tags included in a transaction specify the used inputs and prevent double spending, it is important to prove that the tags belong to the used input secret keys x_i as $\mathsf{tag}_i = G^{\frac{1}{x_i}}$. With a challenge v, all tags are compressed into $\hat{T} = \prod \vec{T}^{\circ \vec{v}^{|\mathcal{S}|}}$ and $\vec{\hat{t}} = \mathbf{T}\vec{v}^{|\mathcal{S}|}$. In the compression of all parameters, the element $\hat{T}$ is compressed with K' separated by a power of u, therefore not increasing the length of the encoded witness. With $\phi_2 = -\langle \vec{v}^{|\mathcal{S}|}, \vec{x}^{\circ -1} \rangle$ and $\eta_2 = -\langle \vec{v}^{|\mathcal{S}|}, \vec{x} \rangle$ the constraints are

$$
\begin{array}{lccccccccccl}
\Lambda^{\mathsf{tk}} := & (& G \| H \| \hat{T} \| & \vec{R} &) & & & & & & & \\
\vec{c}_L^{\,\mathsf{tk}} := & (& \phi_2 \| \eta_2 \| 1 \| & \vec{\hat{t}} & \| & \vec{x} & \| & \mathsf{vec}(\mathbf{E}_{\mathcal{S}}) & \| & \mathsf{vec}(\mathbf{T}) &) & \\
\vec{c}_R^{\,\mathsf{tk}} := & (& \cdot \quad \cdot \quad \cdot \quad \cdot & & & \vec{x}^{\circ -1} & & \mathsf{vec}(\mathbf{E}_{\mathcal{S}}) - \vec{1}^{|\mathcal{R}\|\mathcal{T}|} & \| & \mathsf{vec}(\mathbf{T}) - \vec{1}^{|\mathcal{R}\|\mathcal{S}|} &) & \\
\vec{\mathrm{v}}_1^{\mathsf{tk}} := & (& \cdot \quad 1 \quad \cdot \quad \cdot & & & \vec{v}^{|\mathcal{S}|} & & \cdot & & \cdot &) & : \mathfrak{dir}, \hat{\mathrm{v}}_1^{\mathsf{tk}} = 0 \\
\vec{\mathrm{v}}_2^{\mathsf{tk}} := & (& \cdot \quad \cdot \quad \cdot \quad \cdot & & & \vec{y}^{|\mathcal{S}|} & & \cdot & & \cdot &) & : \mathfrak{mul}, \hat{\mathrm{v}}_2^{\mathsf{tk}} = y^* \\
\vec{\mathrm{v}}_3^{\mathsf{tk}} := & (& 1 \quad \cdot \quad \cdot \quad \cdot & & & \cdot & & \cdot & & \cdot &) & : \mathfrak{sum}, \hat{\mathrm{v}}_3^{\mathsf{tk}} = 0 \\
\vec{\mathrm{v}}'^{\mathsf{tk}}_3 := & (& \cdot \quad \cdot \quad \cdot \quad \cdot & & & \vec{v}^{|\mathcal{S}|} & & \cdot & & \cdot &) & \\
\vec{\mathrm{v}}_4^{\mathsf{tk}} := & (& \cdot \quad \cdot \quad \cdot \quad \cdot & & & \cdot & & \vec{1}^{|\mathcal{T}|} \otimes \vec{y}^{|\mathcal{R}|} & \| & -\vec{1}^{|\mathcal{S}|} \otimes \vec{y}^{|\mathcal{R}|} &) & : \mathfrak{dir}, \hat{\mathrm{v}}_4^{\mathsf{tk}} = 0 \\
\vec{\mathrm{v}}_5^{\mathsf{tk}} := & (& \cdot \quad \cdot \quad \cdot \quad \cdot & & & \cdot & & \cdot & & \vec{y}^{|\mathcal{S}|} \otimes \vec{1}^{|\mathcal{R}|} &) & : \mathfrak{dir}, \hat{\mathrm{v}}_5^{\mathsf{tk}} = y^* \\
\vec{\mathrm{v}}_6^{\mathsf{tk}} := & (& \cdot \quad \cdot \quad \cdot & -\vec{y}^{|\mathcal{R}|} & & \cdot & & \cdot & & \vec{v}^{|\mathcal{S}|} \otimes \vec{y}^{|\mathcal{R}|} &) & : \mathfrak{dir}, \hat{\mathrm{v}}_6^{\mathsf{tk}} = 0
\end{array}
$$

with $y^* = \langle \vec{1}^{|\mathcal{S}|}, \vec{y}^{|\mathcal{S}|} \rangle$. Constraint $\vec{\mathrm{v}}_2^{\mathsf{tk}}$ assures the correct inverse of $\vec{x}$ and $\vec{\mathrm{v}}_5^{\mathsf{tk}}$ checks that $\mathbf{T}$ consists of unit vectors. Additionally $\mathbf{T}$ must be binary, which is accounted for by extending the constraints $\vec{\mathrm{v}}_5^{\mathsf{gr}}$ and $\vec{\mathrm{v}}_6^{\mathsf{gr}}$.

5.6 SECURITY ANALYSIS

The security of the resulting multi-type system is captured by the following theorem.

Theorem 5.1 (Multi-type Omniring)**.** *If the typed commitments are binding, and our extended* $\mathsf{SoK}[\mathcal{L}_{coloring}]$ *is extractable, our construction is balanced according to Definition 5.3. If the Omniring construction is private and the commitments are binding and hiding and* $\mathsf{SoK}[\mathcal{L}_{coloring}]$ *is simulatable, our construction is private. If Omniring is non-slanderable, and* $\mathsf{SoK}[\mathcal{L}_{coloring}]$ *is extractable and simulatable, our construction is non-slanderable.*

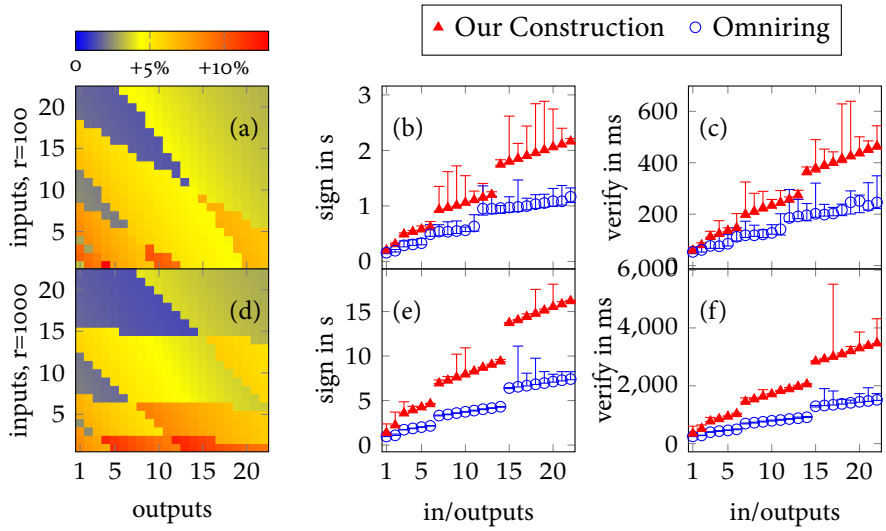


Figure 5.3: Size of out multi-type construction relative to Omniring for anonymity set sizes (a) $r = 100$ and (d) $r = 1000$. The signing and verification times are for equal numbers of inputs and outputs and 100 ring members for (b)(c) and 1000 in (e)(f).

As we do not change the Omniring construction and the vector Pedersen commitments are binding and hiding, the security holds analogous to the proofs of Omniring. Then it remains to show that $\mathsf{SoK}[\mathcal{L}_{\text{coloring}}]$ is a secure NIZK SoK which is simulatable and extractable. As the protocol is only a different parametrization including set anonymity and time locks of the proof for $\mathcal{L}_{ebp}$, $\mathsf{SoK}[\mathcal{L}_{\text{coloring}}]$ is simulatable and extractable according to the Theorems 4.1 and 4.2.

5.7 EVALUATION

To show the change in transaction size and computation time of extending Omniring with confidential types, we adapted our prototype type conservation implementation to integrate the additional requirements. Our transactions require $2n+m+7+2+1+2\lceil\log_2(5+7+r+r+n+m+4(n+m)+r\cdot n+r\cdot m+n^2+64n+24m)\rceil$ elements for m inputs and n outputs with anonymity set size r. We chose to compare our scheme to the original Omniring which has a transaction size of $2n+m+7+2+2\lceil\log_2(3+r+r\cdot m+64n+3m)\rceil$ elements. Figure 5.3a presents the overhead of our construction in relation to Omniring for an anonymity set of 100. Even for large anonymity set sizes of 1000 inputs (Figure 5.3d), our typed construction has only up to 10% overhead while providing confidential types and time locks. The runtime differences of Omniring and our construction are promising for a proof of concept implementation as seen in Figures 5.3b,5.3c for $r = 100$ and 5.3e,5.3f for $r = 1000$.

5.8 CONCLUSION

In this chapter we formalized a multi-type RingCT system on the basis of the Omniring RingCT formalization. The similarity to the Omniring model allowed us to use their existing boilerplate and only change the core transaction signature to achieve a construction. Our new transaction signature nearly achieves the performance of the Omniring counterpart while supporting confidential types. Our presented multi-type RingCT scheme is useful for applications which require many separate types of tokens but have little interaction between them. Still, all types benefit from the common anonymity set and transactions do not leak the types involved. We present an example of such an application in Chapter 7.

6

6.1 OVERVIEW

The privacy-preserving multi-type transaction system of the previous section allows for efficient transfer of differently typed tokens between participants. However to exchange tokens of different types, this system has to rely on external, trusted exchanges. We propose an integrated exchange mechanism which enables participants to anonymously and fairly trade tokens in so-called swap transactions. An earlier version of the results presented in this chapter were published as [EMP+21]. Compared to other exchange mechanism, such as OMAP [GXK+19] or cross chain swaps, we rely on partial transactions, so called offers which can be merged without interaction of the creator. This allows us to perform an atomic swap in a single persisted transaction.
In a swap transaction, e.g. Alice starts off with owning an UTXO of green tokens and Bob one with red tokens. They both want some of the other's type. Assuming a fair exchange rate of e.g. 2 greens are worth 5 reds, both authorize to spend a portion of their tokens on the condition that they get the equivalent amount in the opposite type. Seen as transactions, neither Alice's nor Bob's authorizations and outputs are balanced and thereby cannot be published as transactions. We call these unbalanced transactions *offers*. However taking the union of both authorizations and outputs, we get a balanced transaction. This enabled merging of offers and once published to a ledger, both parties spend their inputs and get the outputs in the new type. Importantly, neither party can steal the funds by aborting the protocol early, thereby we achieve an atomic swap.
Monero as well as Omniring require the final set of outputs to be fixed at the time of signing the input ring(s), which is not available in non-interactive swaps since the swap partner might not be known in advance. To support non-interactive swap transactions, we propose a new transaction structure which slightly resembles Monero. Figure 6.1 a shows two independent parties (Alice and Bob), each with one input $[\mathfrak{o}_\downarrow][42, \mathrm{ty}_g]$ and $[\mathfrak{o}_\downarrow][100, \mathrm{ty}_r]$. However, instead of directly signing inputs with ring signatures, using all outputs concatenated as signing message, we use our new anonymously aggregatable signature (AS) (Chapter 3) in combination with an efficient linkable ring signature for each input. We use a separate ring signature for each input to hide if multiple inputs are contributed by the same signer. Each partial transaction, called offer, has its own outputs $[\mathfrak{o}_\downarrow]\ a,\mathrm{ty}$ which are joined by merging offers. We use the intermediate copies for amounts and types a,ty as Monero does for amounts only. The openings of our intermediate and output commitments are part of the offer and therefore offers must not be published globally. We envision multiple off-chain dissemination scenarios:

1. The offer is not shared with anyone. The outputs are then designated recipients, resulting in a simple transaction.

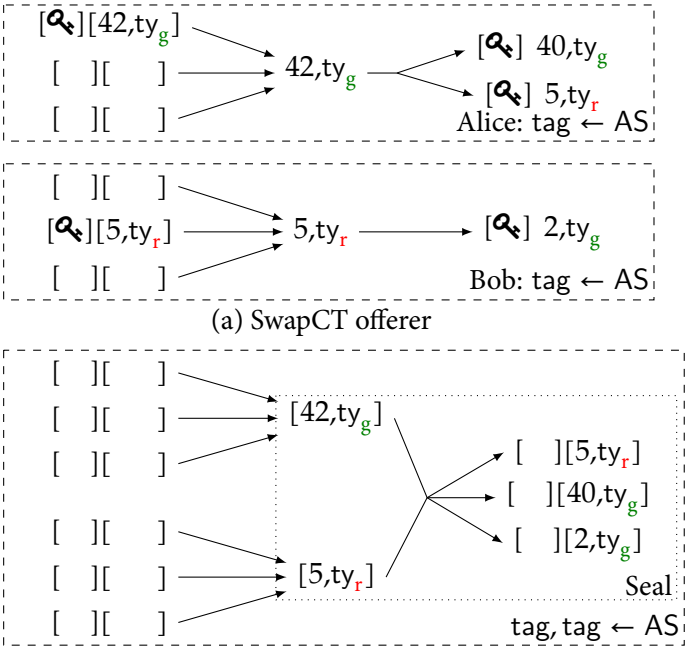


Figure 6.1: Transaction structures for (a) two SwapCT offers and (b) merged into one SwapCT transaction. Dashed rectangles enclose the message for the signatures to authorize spending and dotted rectangles enclose the message for the asset conservation SoK. $[\cdot]$ denotes a commitment to the amount, type, or a one-time key. Empty [] denote that the opening is not known to the signer or merger.[EMP+21]

2. The offer is sent to a peer who created a matching offer. The offers are merged by the peer for an atomic swap transaction.

3. The offer is shared with a small group of participants who may be interested to fulfill the offer.

4. The offer is sent to a mostly untrusted exchange. Exchanges in our SwapCT system are entrusted with publishing an order-book and merging of offers. Due to our AS, ring signature and one-time accounts, SwapCT exchanges cannot steal tokens or deanonymize their users.

A merged transaction is sealed to assure amount and type conservation (Seal). The real senders are hidden from the merger who only seals a transaction with access to the opening of the output commitments and the intermediate commitments. However a merger does not know the witness of the ring signature keeping senders anonymous and the merger only gets a uniformly random output one-time public key. Miners without insight into transactions have no advantage reordering them to front-run offers.
To support confidential types, the wealth conservation requires a zero-knowledge proof that the committed types are valid. The ledger, similar to the UTXO

set, maintains a list of valid types. New types are registered by proposing a one-time account which holds the initial supply of tokens. If the new type is unique, it is persisted in the DLT and future transactions can reference it as input.
For efficient ring signatures and conservation proofs in SwapCT, our contribution is to propose two efficient NIZK proofs based on the Bulletproof inner product protocol. With an anonymity set size of r and m inputs, this results in transaction sizes of SwapCT between the logarithmic efficiency of Omniring $O(\log(m \cdot r))$ and the linear efficiency of Monero $O(m \cdot r)$. We achieve a transaction size logarithmic in the size of the anonymity set per input $O(m \log(r))$ because of the non-interactive ring signature creation. Depending on the use-case, the SwapCT system is easily adapted to use vector Pedersen commitments to store types and then relies on the conservation proof of Chapter 4. For simplicity, we describe SwapCT with Type Homomorphic Commitments only.
An efficient proof for large ring sizes is important to provide sender anonymity. As SwapCT ring signatures provide the same anonymity as Monero ring signatures, we can use the Binned Mixin Sampling from Möser et al. [MSH+18] to select the ring members. Binned Mixin Sampling references temporally local groups of previous outputs to counter timing attacks and protect against an adversary who controls many outputs. Sampling a proper ring is important, as transaction graph analysis with external information may trace an input to a real sender with high probability. While Monero users are financially incentivized to keep the ring size small due to linear transaction fees per mixin, our SwapCT ring signature sizes only grow logarithmically in the ring size. The default Monero ring size of 11 may provide a worst-case effective untraceability set of 4 to 6 possible inputs. With suggested 123 ring members in SwapCT, the worst-case effective untraceability is between 40 and 60 accoring to Möser et al. [MSH+18]. The effective untraceability is the anonymity set expected after running statistical deanonymization attacks on the transaction graph.

6.2 EXAMPLE

To show the usual interaction of the algorithms of our SwapCT system, we describe a swap between Alice and Bob in Figure 6.2. The system is transparently set up and to participate both users generate a long term key with KeyGen serving as their identity (line 1). As there are no types registered yet, both create their own type with TypeGen from type identifiers (line 2) and generate one-time accounts with themselves as recipients (line 3). The total supply of each type is fixed in this operation. The consensus accepts the registration of a new token type as the identifiers (`green`, `red`) are unique regarding all previous registrations (line 4). Alice now possesses all green tokens and Bob all red tokens. Alice wants 5 red tokens and proceeds to generate outputs $\mathsf{acc}_{A'}$ and $\mathsf{acc}_{A''}$ which deduct 2 green tokens from her account acc_A and give her 5 red tokens (lines 5,6). She authorizes the tentative spending of acc_A by creating an offer off_A with inputs $\mathcal{S}_A$ (line 7), outputs $\widetilde{\mathcal{T}_A}$ (line 8) and ring members

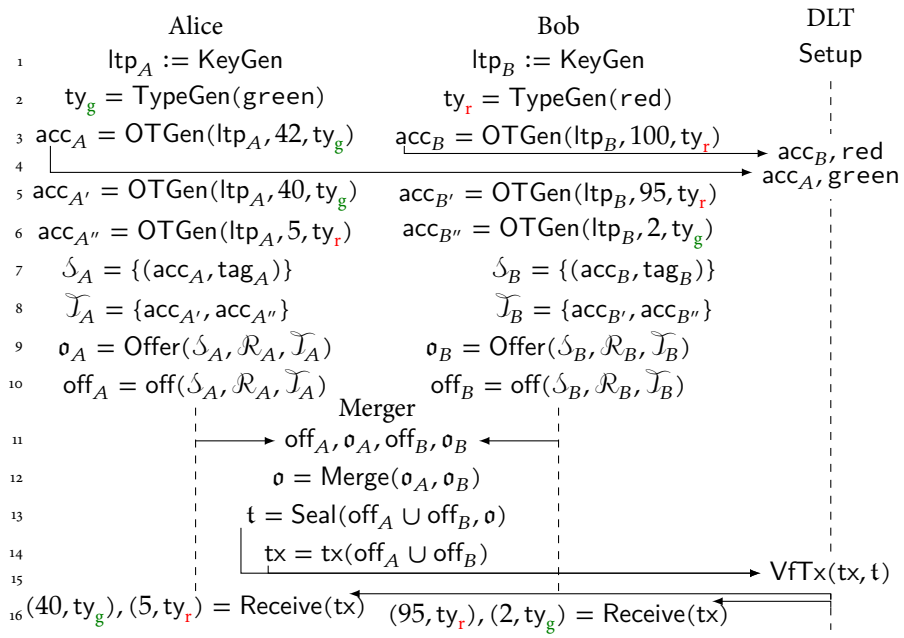


Figure 6.2: Example of a full SwapCT system with setup, type registration and an atomic swap of 2 green for 5 red tokens between Alice and Bob with an untrusted Merger and DLT. Sampling of the rings $\mathcal{R}_a$ and $\mathcal{R}_b$ is covered by related research [MSH+18]. [EMP+21]

$\mathcal{R}_A$ (sampled from all previous outputs) signed by Offer (line 9). Bob creates a complementary offer, which trades 5 red tokens for 2 green tokens. The spending authorization includes publishing of a tag corresponding to the input account. The offers (line 11) and osser signatures are then merged with Merge by either an independent Merger, Alice or Bob (line 12). Merging does not require any secret input. However, each input in an offer contains a unique tag to prevent double-spending. Anyone with access to an offer will recognize its tag in a persisted transaction. Still, access to an offer does not deanonymize the sender or receiver and only reveals the tokens transferred.
A balanced offer, where the input tokens and output tokens match, is then sealed with Seal (line 13) and submitted along with the public elements of the transaction (line 14) to the consensus for verification (VfTx) (line 15). The transaction is persisted, if the signatures are valid and all tags are unique regarding all previous transactions. After persisting the transaction, the designated outputs $\{\mathsf{acc}'_A, \mathsf{acc}''_A\}$ and $\{\mathsf{acc}'_B, \mathsf{acc}''_B\}$ generated in the offer process are received with Receive (line 16) and are usable as future transaction inputs.

6.3 FORMALIZATION

The security of single-type set anonymous systems is well formalized by Omniring [LRR+19]. We extend this formalization to encompass type support and swap transactions.

Definition 6.1. *A Swap Confidential Transaction (SwapCT) scheme consists of a tuple of PPT algorithms* (Setup, KeyGen, TypeGen, OTGen, Offer, VfOffer, Merge, Seal, VfTx, Receive) *defined as follows:*

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\alpha, 1^\beta)$ *takes the security parameter* λ *and integers* α *for a maximum of* 2^α *outputs of a transaction where each has an amount maximum of* $2^\beta - 1$. *Then it outputs public parameters* pp *which are implicitly given to all the following algorithms. Setup is called once when a SwapCT system is initialized.*

$(\mathsf{ltp}, \mathsf{lts}) \leftarrow \mathsf{KeyGen}()$ *generates a long term secret key* lts *with the corresponding long term public key* ltp *for participants to initially join the system. The* ltp *is distributed and serves as a recipient address.*

$\mathsf{ty} \leftarrow \mathsf{TypeGen}(\mathsf{name})$ *generates a type* ty *given a* name.

$\mathsf{acc}, \mathsf{ck} \leftarrow \mathsf{OTGen}(\mathsf{ltp}, a, \mathsf{ty})$ *creates a one-time account* acc *with coin key* ck *from a long term public key* ltp *and an amount* a *of a type* ty *to then use this account as an output in an offer or a new type registration.*

$\mathsf{o} \leftarrow \mathsf{Offer}(\mathcal{S}, \mathcal{R}, \mathcal{T})$ *takes the inputs*

- $\mathcal{S} = \{(\mathsf{tag}_i, j_i, \mathsf{sk}_i, a_i^{\mathcal{S}}, \mathsf{ty}_i^{\mathcal{S}}, \mathsf{ck}_i^{\mathcal{S}})\}_{i=1}^{|\mathcal{S}|}$ *is a set of inputs with a* tag_i *corresponding to* $\mathsf{acc}_{i,j_i}^{\mathcal{R}}$ *at index* $j_i \in [|\mathcal{R}_i|]$, *secret key* sk_i, *amount* $a_i^{\mathcal{S}}$ *and type* $\mathsf{ty}_i^{\mathcal{S}}$ *with coin key* $\mathsf{ck}_i^{\mathcal{S}}$.
- $\mathcal{R} = \{\{\mathsf{acc}_{i,j}^{\mathcal{R}}\}_{j=1}^{|\mathcal{R}_i|}\}_{i=1}^{|\mathcal{S}|}$ *is a set of ring account sets, one set per input to hide the real account.*
- $\mathcal{T} = \{(\mathsf{acc}_i^{\mathcal{T}}, a_i^{\mathcal{T}}, \mathsf{ty}_i^{\mathcal{T}}, \mathsf{ck}_i^{\mathcal{T}})\}_{i=1}^{|\mathcal{T}|}$ *is a set of accounts* $\mathsf{acc}_i^{\mathcal{T}}$ *with amount* $a_i^{\mathcal{T}}$, *type* $\mathsf{ty}_i^{\mathcal{T}}$ *and coin key* $\mathsf{ck}_i^{\mathcal{T}}$.

It outputs a signature o *as authorization to spend the inputs.*

$b \leftarrow \mathsf{VfOffer}(\mathsf{off}, \mathsf{o})$ *takes a signature* o *and an offer*

$$\mathsf{off}(\mathcal{S}, \mathcal{R}, \mathcal{T}) := \Big(\{\mathsf{tag}_i, a_i^{\mathcal{S}}, \mathsf{ty}_i^{\mathcal{S}}\}_{i=1}^{|\mathcal{S}|}, \{\{\mathsf{acc}_{i,j}^{\mathcal{R}}\}_{j=1}^{|\mathcal{R}_i|}\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{acc}_i^{\mathcal{T}}, a_i^{\mathcal{T}}, \mathsf{ty}_i^{\mathcal{T}}, \mathsf{ck}_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}\Big) \tag{6.1}$$

and returns a bit b *specifying if the offer is valid.*

$\mathsf{o} \leftarrow \mathsf{Merge}(\mathsf{o}_0, \mathsf{o}_1)$ *takes two offer signatures and generates a combined one* o *valid for the union of both offers.*

$\mathsf{t} \leftarrow \mathsf{Seal}(\mathsf{off}, \mathsf{o})$ *Takes a valid balanced offer defined as above and its signature* o *and outputs a seal proof* t.

$b \leftarrow \mathsf{VfTx}(\mathsf{tx}, \mathsf{t})$ *takes a transaction defined as*

$$\mathsf{tx}(\mathsf{off}) := \Big(\{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|}, \{\{\mathsf{acc}_{i,j}^{\mathcal{R}}\}_{j=1}^{|\mathcal{R}_i|}\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{acc}_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}\Big)$$

and the signature t *and returns a bit* b *representing the validity.*

$(\mathsf{tag}, \mathsf{sk}, a, \mathsf{ty}, \mathsf{ck}) \leftarrow \mathsf{Receive}(\mathsf{acc}, \mathsf{lts})$ *gets an account* acc *and a long term secret* lts *and returns the matching* tag*, secret key* sk*, amount* a*, type* ty *and coin key* ck *for* acc*.*

Further, we require the following two auxiliary algorithms to define the security properties.

$b \leftarrow \mathsf{ChkAcc}(\mathsf{acc}, a, \mathsf{ty}, \mathsf{ck})$ *takes an account* acc*, an amount* a *with type* ty *and a coin key* ck *and checks if they are consistent.*

$b \leftarrow \mathsf{ChkTag}(\mathsf{acc}, \mathsf{tag}, \mathsf{sk})$ *takes an account* acc*, a* tag *and a secret key* sk *and returns* 1 *if consistent,* o *otherwise.*

Definition 6.2 (Correctness)**.** *A SwapCT scheme is correct, if for all* $\lambda, \alpha, \beta \in \mathbb{N}$ *and all* $\mathsf{pp} \in \mathsf{Setup}(1^\lambda, 1^\alpha, 1^\beta)$*:*

- *Honestly generated payments are received correctly: For any* $\mathsf{ltp}, \mathsf{lts} \in \mathsf{KeyGen}()$*, any* $\mathsf{name} \in (0,1)^*, \mathsf{ty} = \mathsf{TypeGen}(\mathsf{name})$*, any amount* $a \in \{0, \dots, 2^\beta - 1\}$*, any* $(\mathsf{acc}, \mathsf{ck}) \in \mathsf{OTGen}(\mathsf{ltp}, a, \mathsf{ty})$*, and any* $(\cdot, a', \mathsf{ty}', \mathsf{ck}') \in \mathsf{Receive}(\mathsf{acc}, \mathsf{lts})$*, it holds that* $(a, \mathsf{ty}, \mathsf{ck}) = (a', \mathsf{ty}', \mathsf{ck}')$*.*
- *Honestly received payments have a valid amount, type and tag: For any* $(\mathsf{tag}, \mathsf{sk}, a, \mathsf{ty}, \mathsf{ck}) \in \mathsf{Receive}(\mathsf{acc}, \mathsf{lts}), \mathsf{ChkAcc}(\mathsf{acc}, a, \mathsf{ty}, \mathsf{ck}) = 1$ *and* $\mathsf{ChkTag}(\mathsf{acc}, \mathsf{tag}, \mathsf{sk}) = 1$ *hold.*
- *Honestly generated offers are valid: For each* $\mathcal{S}, \mathcal{R}, \mathcal{T}$*, defined as above, that satisfy*
  - $\forall i \in [|\mathcal{S}|], \mathsf{ChkTag}(\mathsf{acc}^{\mathcal{R}}_{i,j_i}, \mathsf{tag}_i, \mathsf{sk}_i) = 1$
  - $\forall i \in [|\mathcal{S}|], \mathsf{ChkAcc}(\mathsf{acc}^{\mathcal{R}}_{i,j_i}, a^{\mathcal{S}}_i, \mathsf{ty}^{\mathcal{S}}_i, \mathsf{ck}^{\mathcal{S}}_i) = 1$
  - $\forall i \in [|\mathcal{T}|], \mathsf{ChkAcc}(\mathsf{acc}^{\mathcal{T}}_i, a^{\mathcal{T}}_i, \mathsf{ty}^{\mathcal{T}}_i, \mathsf{ck}^{\mathcal{T}}_i) = 1$

  and for any signature $\mathfrak{o} \in \mathsf{Offer}(\mathcal{S}, \mathcal{R}, \mathcal{T})$*, it holds that* $\mathsf{VfOffer}(\mathsf{off}, \mathfrak{o}) = 1$ *with* $\mathsf{off} = \mathsf{off}(\mathcal{S}, \mathcal{R}, \mathcal{T})$*.*
- *Honestly merged valid offers are again valid: For each pair of valid* $\mathcal{S}_k, \mathcal{R}_k, \mathcal{T}_k$ *with* $k \in \{0,1\}$*, each* $\mathfrak{o}_k \in \mathsf{Offer}(\mathcal{S}_k, \mathcal{R}_k, \mathcal{T}_k)$ *and* $\mathfrak{o} = \mathsf{Merge}(\mathfrak{o}_0, \mathfrak{o}_1)$*, it holds that* $\mathsf{VfOffer}(\mathsf{off}, \mathfrak{o}) = 1$ *with* $\mathsf{off} = \mathsf{off}(\mathcal{S}_0 \cup \mathcal{S}_1, \mathcal{R}_0 \cup \mathcal{R}_1, \mathcal{T}_0 \cup \mathcal{T}_1)$*.*
- *Honestly sealed transactions are valid: For any* $\mathcal{S}, \mathcal{R}, \mathcal{T}$ *as above that satisfies all offer criteria and:*
  - $|\mathcal{T}| \leq 2^\alpha, \forall i \in [|\mathcal{T}|] : a^{\mathcal{T}}_i \in \{0, \dots, 2^\beta - 1\}$
  - $\forall \mathsf{ty} \in \{\mathsf{ty}^{\mathcal{T}}_i\}^{|\mathcal{T}|}_{i=1} : \sum\{a^{\mathcal{S}}_i | \mathsf{ty}^{\mathcal{S}}_i = \mathsf{ty}\}^{|\mathcal{S}|}_{i=1} = \sum\{a^{\mathcal{T}}_i | \mathsf{ty}^{\mathcal{T}}_i = \mathsf{ty}\}^{|\mathcal{T}|}_{i=1}$

  and for any proof $\mathsf{t} \in \mathsf{Seal}(\mathsf{off}, \mathfrak{o})$ *it holds that* $\mathsf{VfTx}(\mathsf{tx}, \mathsf{t}) = 1$ *with* $\mathsf{off} = \mathsf{off}(\mathcal{S}, \mathcal{R}, \mathcal{T})$ *and* $\mathsf{tx} = \mathsf{tx}(\mathsf{off})$*.*

6.4 SECURITY

To formalize the security of a SwapCT scheme, we borrow components from other RingCT schemes, namely Omniring, as transactions of a SwapCT scheme should have comparable properties to their single type RingCT counterparts. For the non-slanderability, we make use of the definitions from Omniring [LRR+19], which allows an attacker to make arbitrary transactions through oracles and who must then output a valid transaction which uses a tag of the honest, oracle controlled accounts that was not previously authorized. Theft prevention and privacy of SwapCT require additional constraints to ensure these properties in the presence of offers, types and multiple distrusting parties jointly transacting.

6.4.1 *Theft*

The core of any transaction system is to assure that no value is created out of thin air. Moreover, for individual participants it is of paramount importance that all outgoing transactions from their wealth are authorized.
The authorization in the case of a single transactor is simple: Signing the transaction with a set of inputs for a final set of outputs. This is not directly applicable to offers that may not have a final output set at the time of authorizing the first inputs. The authorization given to an offer translates to a condition that the signer accepts the spending of the inputs if and only if the specified outputs are fulfilled. As offers have to be authorized before some untrusted party uses them to seal a transaction and without further interaction, the authorization must be conditioned that all designated outputs are included in the final transaction without modifications. As long as at least one input authorization, identified by a tag, is used from an offer, the transaction outputs must be a super-set of the original offer outputs. The tag is anonymously bound to the hidden input.
To achieve theft prevention, we require ChkAcc and ChkTag to be binding. Then, if a tag is bound to a source account, double-spend detection is reduced to checking for duplicate tags. In addition, the binding property prevents opening an account to a different amount or type.

Definition 6.3 (Theft)**.** *A SwapCT scheme is theft protecting, if for any* $\lambda \in \mathbb{N}$ *and all* $\alpha, \beta \in \mathsf{poly}(\lambda)$ *with* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\alpha, 1^\beta)$

1. ChkTag *and* ChkAcc *are binding such that for any adversary* $\mathcal{A}$ *it holds that*

$$\Pr\left[\begin{array}{c} \mathsf{ChkTag}(\mathsf{acc}, \mathsf{sk}, \mathsf{tag}) = 1 \\ \mathsf{ChkTag}(\mathsf{acc}, \mathsf{sk}', \mathsf{tag}') = 1, (\mathsf{sk}, \mathsf{tag}) \neq (\mathsf{sk}', \mathsf{tag}') \\ (\mathsf{acc}, \mathsf{sk}, \mathsf{tag}, \mathsf{sk}', \mathsf{tag}') \leftarrow \mathcal{A}(\mathsf{pp}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

and

$$\Pr\left[\begin{array}{c|c} \mathsf{ty} = \mathsf{TypeGen}(n), \mathsf{ty}' = \mathsf{TypeGen}(n') & \\ \mathsf{ChkAcc}(\mathsf{acc}, \mathsf{ck}, a, \mathsf{ty}) = 1 & \\ \mathsf{ChkAcc}(\mathsf{acc}, \mathsf{ck}', a', \mathsf{ty}') = 1 & \\ (a, n, \mathsf{ck}) \neq (a', n', \mathsf{ck}') & \\ (\mathsf{acc}, a, n, \mathsf{ck}, a', n', \mathsf{ck}') \leftarrow \mathcal{A}(\mathsf{pp}) & \end{array}\right] \leq \mathsf{negl}(\lambda)$$

2. *for all PPT adversaries* $\mathcal{A}$ *and all* $\mathcal{S}, \mathcal{R}, \mathcal{T}$ *defined as above, it holds that*

$$\Pr\left[\begin{array}{c|c} \{\mathsf{tag}'_i\}_{i=1}^{|\mathcal{S}'|} \cap \{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|} \neq \emptyset & \\ \{\mathsf{acc}'^{\mathcal{T}}_i\}_{i=1}^{|\mathcal{T}'|} \not\supseteq \{\mathsf{acc}_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}, \mathsf{VfOffer}(\mathsf{off}', \mathfrak{o}') = 1 & \\ \mathfrak{o} \leftarrow \mathsf{Offer}(\mathcal{S}, \mathcal{R}, \mathcal{T}) & \\ (\mathfrak{o}', \mathsf{off}') \leftarrow \mathcal{A}(\mathsf{pp}, \mathfrak{o}, \mathsf{off}(\mathcal{S}, \mathcal{R}, \mathcal{T})) & \end{array}\right] \leq \mathsf{negl}(\lambda).$$

6.4.2 *Balance*

To prevent users from spending more value than they have as input or spending the same value twice, increasing the total supply, a transaction must be balanced. Our balance property only differs from a type unaware RingCT system in the fact that the balance has to hold for each type individually.
To achieve the balance property, we rely on theft prevention, as a prerequisite. The balance experiment in Figure 6.1 states that for any efficient adversary $\mathcal{A}$ which generates a valid transaction $\mathsf{tx}, \mathfrak{t}$, there exists an extractor $\mathcal{E}_{\mathcal{A}}$ which extracts the witness $\mathcal{S}, \mathcal{R}, \mathcal{T}$ of this transaction. The witness must satisfy that the tags match the inputs $\mathsf{acc}^{\mathcal{R}}_{i,j_i}$. In addition, the amounts and types must match the input and output accounts with ChkAcc. Unlike single type transaction systems, we additionally require that each output type is present in the input. Then, the sum of amounts in the inputs must be equal to the sum of outputs per type.
Assume an efficient adversary creating a valid transaction signature for an unbalanced transaction. The signature ensures balance and thereby the adversary can be used to create an efficient adversary against one of the binding properties. This means being able to spend the same account under a different tag or change the amount or type.

Definition 6.4 (Balance)**.** *A SwapCT scheme is balanced if it prevents theft (Def. 2) and for all PPT adversaries* $\mathcal{A}$ *and all positive integers* $\alpha, \beta \in \mathsf{poly}(\lambda)$*, there exists a PPT extractor* $\mathcal{E}_{\mathcal{A}}$ *such that*

$$\Pr[\mathsf{Balance}_{\mathcal{A},\mathcal{E}_{\mathcal{A}}}(1^\lambda, 1^\alpha, 1^\beta) = 1] \leq \mathsf{negl}(\lambda)$$

with $\mathsf{Balance}_{\mathcal{A},\mathcal{E}_{\mathcal{A}}}(1^\lambda, 1^\alpha, 1^\beta)$ *defined in Algorithm 6.1.*

Algorithm 6.1 $\text{Balance}_{\mathcal{A},\mathcal{E}_{\mathcal{A}}}(1^\lambda, 1^\alpha, 1^\beta)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\alpha, 1^\beta)$
$(\mathsf{tx}, \mathsf{t}) \leftarrow \mathcal{A}(\mathsf{pp})$
$(\mathcal{S}, \mathcal{R}, \overline{\mathcal{T}}) \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{pp}, \mathsf{tx}, \mathsf{t})$
parse $\mathcal{S}$ as $\left\{\mathsf{tag}_i, j_i, \mathsf{sk}_i, a_i^{\mathcal{S}}, \mathsf{ty}_i^{\mathcal{S}}, \mathsf{ck}_i^{\mathcal{S}}\right\}_{i=1}^{|\mathcal{S}|}$
parse $\mathcal{R}$ as $\left\{\{\mathsf{acc}_{i,j}^{\mathcal{R}}\}_{j=1}^{|\mathcal{R}_i|}\right\}_{i=1}^{|\mathcal{R}|}$
parse $\overline{\mathcal{T}}$ as $\left\{\mathsf{acc}_i^{\overline{\mathcal{T}}}, a_i^{\overline{\mathcal{T}}}, \mathsf{ty}_i^{\overline{\mathcal{T}}}, \mathsf{ck}_i^{\overline{\mathcal{T}}}\right\}_{i=1}^{|\overline{\mathcal{T}}|}$
$b_1 := \mathsf{VfTx}(\mathsf{tx}, \mathsf{t}), b_2 := \mathsf{tx} = \mathsf{tx}(\mathcal{S}, \mathcal{R}, \overline{\mathcal{T}})$
$b_3 := \forall i \in [|\mathcal{S}|], \mathsf{ChkTag}(\mathsf{acc}_{i,j_i}^{\mathcal{R}}, \mathsf{sk}_i, \mathsf{tag}_i) = 1$
$b_4 := \forall i \in [|\mathcal{S}|], \mathsf{ChkAcc}(\mathsf{acc}_{i,j_i}^{\mathcal{R}}, a_i^{\mathcal{S}}, \mathsf{ty}_i^{\mathcal{S}}, \mathsf{ck}_i^{\mathcal{S}}) = 1$
$b_5 := \forall i \in [|\overline{\mathcal{T}}|], \mathsf{ChkAcc}(\mathsf{acc}_i^{\overline{\mathcal{T}}}, a_i^{\overline{\mathcal{T}}}, \mathsf{ty}_i^{\overline{\mathcal{T}}}, \mathsf{ck}_i^{\overline{\mathcal{T}}}) = 1$
$b_6 := \forall \mathsf{ty} \in \{\mathsf{ty}_i^{\overline{\mathcal{T}}}\}_{i=1}^{|\overline{\mathcal{T}}|} : \sum\{a_i^{\mathcal{S}} | \mathsf{ty}_i^{\mathcal{S}} = \mathsf{ty}\}_{i=1}^{|\mathcal{S}|} = \sum\{a_i^{\overline{\mathcal{T}}} | \mathsf{ty}_i^{\overline{\mathcal{T}}} = \mathsf{ty}\}_{i=1}^{|\overline{\mathcal{T}}|}$
return $b_1 \wedge b_2 \wedge \neg(b_3 \wedge b_4 \wedge b_5 \wedge b_6)$

6.4.3 *Privacy*

The privacy of a SwapCT scheme consists of two different settings. Offers require sender and receiver anonymity, while for sealed transactions the regular RingCT privacy must hold, which consists of sender and receiver anonymity as well as value confidentiality. To provide value confidentiality in a SwapCT, we have to extend the RingCT model by also hiding the type.
The transaction creation process may be distributed and offers are passed to possibly malicious parties. Therefore, we require sender and receiver anonymity for offers, too. Value and type confidentiality are not desired for offers, as other parties must be able to access the offered assets and decide if they want to merge the offer. To ensure that swap transactions are indistinguishable from single-user transactions, the number of offers merged together must remain hidden, making merged offers appear identical to single transactions.
Sender and receiver anonymity is defined by an adversary interacting with a set of oracles and then presenting a maliciously crafted offer together with instructions for the security experiment on how to construct two sets of offers. The instructions contain input ring accounts with two possible senders and recipients together with amounts and types. In addition, they contain an identifier of the party which should use the input or output, thereby showing that the offers do not reveal the participants. The adversary should not be able to distinguish which set of offers is created and merged.
More formally, we specified a security experiment OffPv^b in Figure 6.2 with a bit $b \in \{0, 1\}$. An adversary $\mathcal{A}$ queries the available oracles (Algorithms 6.3,6.4,6.5,6.6) and then returns a valid offer $(\mathsf{off}, \mathfrak{o})$ and instructions I and J. I contains $\{(\{u_{t,i}^{\mathcal{S}}, j_{t,i}\}_{t=0}^{1}, \{\mathsf{acc}_{i,j}^{\mathcal{R}}\}_{j=1}^{|\mathcal{R}_i|})\}_{i=1}^{|I|}$ where $u_{b,i}^{\mathcal{S}}$ is the identifier of the

Algorithm 6.2 $\mathsf{OffPv}^{b}_{\mathcal{A}}(1^{\lambda}, 1^{\alpha}, 1^{\beta})$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^{\lambda}, 1^{\alpha}, 1^{\beta}), \mathsf{InitOracles}()$
$\mathbb{O} = \{\mathsf{KeyGen}O, \mathsf{Offer}O\}$
$(I, J, \mathsf{off}, \mathfrak{o}) \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{pp})$
$\mathfrak{o}_0 := \mathfrak{o}_1 := \mathfrak{o}, \mathsf{off}_0 := \mathsf{off}_1 := \mathsf{off}, \mathcal{S}_0 := \mathcal{S}_1 := \mathcal{T}_0 := \mathcal{T}_1 := \emptyset$
if $\mathsf{VfOffer}(\mathsf{off}, \mathfrak{o}) = 0$ **then return** o
parse I as $\{(\{(u^{\mathcal{S}}_{t,i}, j_{t,i})\}^{1}_{t=0}, \{\mathsf{acc}^{\mathcal{R}}_{i,j}\}^{|\mathcal{R}_i|}_{j=1})\}^{|I|}_{i=1}$
for all $i \in [|I|]$ **do**
  for all $t \in \{0, 1\}$ **do**
    $(\mathsf{tag}_{t,i}, \mathsf{sk}^{\mathcal{S}}_{t,i}, a^{\mathcal{S}}_{t,i}, \mathsf{ty}^{\mathcal{S}}_{t,i}, \mathsf{ck}^{\mathcal{S}}_{t,i}) := \mathsf{TryReceive}(\mathsf{acc}^{\mathcal{R}}_{i,j_{t,i}})$
    $\mathcal{S}_t[i] = (\mathsf{tag}_{t,i}, j_{t,i}, \mathsf{sk}^{\mathcal{S}}_{t,i}, a^{\mathcal{S}}_{t,i}, \mathsf{ty}^{\mathcal{S}}_{t,i}, \mathsf{ck}^{\mathcal{S}}_{t,i})$
    if $\mathsf{tag}_{t,i} \in \mathsf{Offrd} \wedge (\mathsf{tag}_{t,i}, \{\mathsf{acc}^{\mathcal{R}}_{i,j}\}^{|\mathcal{R}_i|}_{j=1}) \notin \mathsf{Offrd}$ **then return** o
  if $a_{0,i} \neq a_{1,i} \vee \mathsf{ty}_{0,i} \neq \mathsf{ty}_{1,i}$ **then return** o
parse J as $\{(\{(u^{\mathcal{T}}_{t,i}, k^{\mathcal{T}}_{t,i})\}^{1}_{t=0}, a^{\mathcal{T}}_{i}, \mathsf{ty}^{\mathcal{T}}_{i})\}^{|J|}_{i=1}$
for all $j \in [|J|]$ **do**
  for all $t \in \{0, 1\}$ **do**
    $(\mathsf{acc}^{\mathcal{T}}_{t,j}, \mathsf{ck}^{\mathcal{T}}_{t,j}) \leftarrow \mathsf{OTGen}(\mathsf{LTP}[k_{t,j}], a^{\mathcal{T}}_{j}, \mathsf{ty}^{\mathcal{T}}_{j})$
    $\mathcal{T}_t[j] = (\mathsf{acc}^{\mathcal{T}}_{t,j}, a^{\mathcal{T}}_{j}, \mathsf{ty}^{\mathcal{T}}_{j}, \mathsf{ck}^{\mathcal{T}}_{t,j})$
for all $t \in \{0, 1\}$ **do**
  $\mathfrak{U}^{I}_{t} := \{u^{\mathcal{S}}_{t,i}\}^{|I|}_{i=1}, \mathfrak{U}^{J}_{t} := \{u^{\mathcal{T}}_{t,j}\}^{|J|}_{j=1}$
  if $\mathfrak{U}^{I}_{t} \neq \mathfrak{U}^{J}_{t}$ **then return** o
  for all $k \in \mathfrak{U}^{I}_{t}$ **do**
    $\mathcal{S}^{k}_{t} := \{\mathcal{S}_t[i] | u^{\mathcal{S}}_{t,i} = k\}^{|I|}_{i=1}$
    $\mathcal{R}^{k}_{t} := \{\{\mathsf{acc}_{i,j}\}^{|\mathcal{R}|}_{j=1} | u^{\mathcal{S}}_{t,i} = k\}^{|I|}_{i=1}$
    $\mathcal{T}^{k}_{t} := \{\mathcal{T}_t[j] | u^{\mathcal{T}}_{t,i} = k\}^{|J|}_{j=1}$
    $\mathsf{off}^{k}_{t} \leftarrow \mathsf{off}(\mathcal{S}^{k}_{t}, \mathcal{R}^{k}_{t}, \mathcal{T}^{k}_{t})$
    $\mathfrak{o}^{k}_{t} \leftarrow \mathsf{Offer}(\mathcal{S}^{k}_{t}, \mathcal{R}^{k}_{t}, \mathcal{T}^{k}_{t})$
    if $\mathsf{VfOffer}(\mathsf{off}^{k}_{t}, \mathfrak{o}^{k}_{t}) = 0$ **then return** o
    $\mathsf{off}_t := \mathsf{off}_t \cup \mathsf{off}(\mathcal{S}^{k}_{t}, \mathcal{R}^{k}_{t}, \mathcal{T}^{k}_{t})$
    $\mathfrak{o}_t \leftarrow \mathsf{Merge}(\mathfrak{o}_t, \mathfrak{o}^{k}_{t})$
$b' \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{off}_b, \mathfrak{o}_b)$ **return** b'

Algorithm 6.3 $\mathsf{InitOracles}()$

$\mathsf{LTP} := \mathsf{LTS} := \emptyset$
$\mathsf{Offrd} := \emptyset$

Algorithm 6.4 $\mathsf{KeyGen}O()$

$(\mathsf{ltp}, \mathsf{lts}) \leftarrow \mathsf{KeyGen}()$
$\mathsf{LTP} := \mathsf{LTP} \| \mathsf{ltp}$
$\mathsf{LTS} := \mathsf{LTS} \| \mathsf{lts}$
return ltp

Algorithm 6.5 TryReceive(acc)

for all $i \in [|\mathsf{LTS}|]$ **do**
  $(\mathsf{tag}, \mathsf{sk}, a, \mathsf{ty}, \mathsf{ck}) \leftarrow \mathsf{Receive}(\mathsf{acc}, \mathsf{LTS}[i])$
  if $(\mathsf{tag}, \mathsf{sk}, a, \mathsf{ty}, \mathsf{ck}) \neq \bot$ **then**
    return $(\mathsf{tag}, \mathsf{sk}, a, \mathsf{ty}, \mathsf{ck})$
return $\bot$

Algorithm 6.6 Offer$O(I, \mathcal{T})$

$\mathcal{S} := \emptyset$
parse I as $\{j_i, \{\mathsf{acc}^{\mathcal{R}}_{i,j}\}_{j=1}^{|\mathcal{R}_i|}\}_{i=1}^{|I|}$
for all $i \in [|I|]$ **do**
  $\mathcal{S}_i := (\mathsf{tag}_i, \mathsf{sk}_i, a_i, \mathsf{ty}_i, \mathsf{ck}_i) := \mathsf{TryReceive}(\mathsf{acc}^{\mathcal{R}}_{i,j_i})$
  // Check that the same tag was not used with another ring
  if $(\mathsf{tag}_i, \cdot) \in \mathsf{Offrd} \wedge (\mathsf{tag}_i, \{\mathsf{acc}^{\mathcal{R}}_{i,j}\}_{j=1}^{|\mathcal{R}_i|}) \notin \mathsf{Offrd}$ **then return** $\bot$
$\mathcal{R} = \{\{\mathsf{acc}^{\mathcal{R}}_{i,j}\}_{j=1}^{|\mathcal{R}_i|}\}_{i=1}^{|I|}, \mathsf{off} = \mathsf{off}(\mathcal{S}, \mathcal{R}, \mathcal{T}), \mathfrak{o} \leftarrow \mathsf{Offer}(\mathcal{S}, \mathcal{R}, \mathcal{T})$
if $\mathsf{VfOffer}(\mathsf{off}, \mathfrak{o}) = 0$ **then return** $\bot$
$\mathsf{Offrd} := \mathsf{Offrd} \cup \{(\mathsf{tag}_i, \{\mathsf{acc}^{\mathcal{R}}_{i,j}\}_{j=1}^{|\mathcal{R}_i|})\}_{i=1}^{|I|}$
return $\mathfrak{o}$

party who should use input i depending on the selected bit b. The set of all input identifiers $\{u^{\mathcal{S}}_{t,i}\}_{i=1}^{|I|}$ form the unordered set $\mathfrak{U}^I_t$ for $t \in \{0, 1\}$. $j_{t,i}$ specifies the index in the set of ring accounts $\{\mathsf{acc}^{\mathcal{R}}_{i,j}\}_{j=1}^{|\mathcal{R}_i|}$ on which the experiment calls $(\mathsf{tag}_{t,i}, a^{\mathcal{S}}_{t,i}, \mathsf{ty}^{\mathcal{S}}_{t,i}, \mathsf{sk}^{\mathcal{S}}_{t,i}) \leftarrow \mathsf{TryReceive}(\mathsf{acc}^{\mathcal{R}}_{i,j_{t,i}})$ to recover the account secrets. Some trivial cases which are easy to distinguish are excluded. An efficient adversary exists, if the amount and type of the two ring accounts may be different, as these values will be published in the merged offer. Therefore, we require $a^{\mathcal{S}}_{0,i} = a^{\mathcal{S}}_{1,i}$ and $\mathsf{ty}^{\mathcal{S}}_{0,i} = \mathsf{ty}^{\mathcal{S}}_{1,i}$ for all i. We also abort if one of the tags was already disclosed in another offer: $(\mathsf{tag}_{t,i}, \cdot) \notin \mathsf{Offrd}$. This implies that when a participant decides to create multiple offers with the identical input, it is important to the sender anonymity that the input uses the same ring in each offer as otherwise, the real sender is an account of the intersection of all rings. The output instructions J are similar to I, as they contain

$$\{(\{u^{\mathcal{T}}_{t,i}, k^{\mathcal{T}}_{t,i}\}_{t=0}^{1}, a^{\mathcal{T}}_i, \mathsf{ty}^{\mathcal{T}}_i)\}_{i=1}^{|J|}$$

where $u^{\mathcal{T}}_{t,i}$ specifies the party to use this output and all output identifiers form the unordered set $\mathfrak{U}^J_t$, equal to $\mathfrak{U}^I_t$. The element $k_{t,i}$ references an uncompromized long term public key $\mathsf{LTP}[k_{t,i}]$ from which a one-time account $\mathsf{acc}^{\mathcal{T}}_{t,j}$ is derived. As the amounts $a^{\mathcal{T}}_i$ and types $\mathsf{ty}^{\mathcal{T}}_i$ of an offer are public, they are equivalent in both values of b. The experiment proceeds by distributing the inputs and outputs to each identifier in the set $\mathfrak{U}_t = \mathfrak{U}^I_t = \mathfrak{U}^J_t$, then creates a set of offers and authorizes each of them. All offers off^b_k, along with the malicious off are merged. The adversary wins by correctly guessing the bit b.

Definition 6.5 (Offer Privacy). *A SwapCT scheme has private swaps, if for all PPT adversaries $\mathcal{A}$ and all positive integers $\alpha, \beta \in \mathsf{poly}(\lambda)$ it holds that*

$$\left|\Pr[\mathsf{OffPv}^0_{\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta) = 1] - \Pr[\mathsf{OffPv}^1_{\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta) = 1]\right| \leq \mathsf{negl}(\lambda)$$

with $\mathsf{OffPv}^b_{\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta)$ defined in Figure 6.2.

6.4.4 *Transaction Privacy*

The privacy of a sealed transaction extends the offer privacy as follows. As the amounts $a^{\mathcal{S}}_i, a^{\mathcal{T}}_i$ and types $\mathsf{ty}^{\mathcal{S}}_i, \mathsf{ty}^{\mathcal{T}}_i$ of an offer are discarded in the seal operation, the requirement on the instructions from the adversary to have equal amounts and types are lifted. According to Definition 6.5, the number of transactors is hidden. Thereby, it is sufficient to show the case where the adversary provides instructions to just one party. The security experiment in Figure 6.7 then seals the merged transaction at the end.

Algorithm 6.7 $\mathsf{TxPv}^b_{\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\alpha, 1^\beta), \mathsf{InitOracles}()$
$\mathbb{O} = \{\mathsf{KeyGen}O, \mathsf{Offer}O\}$
$(I, J, \mathsf{off}, \mathfrak{o}) \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{pp})$
$\mathcal{S}_0 := \mathcal{T}_0 := \mathcal{S}_1 := \mathcal{T}_1 := \mathcal{R} := \emptyset$
parse I as $\{(\{j_{t,i}\}^1_{t=0}, \{\mathsf{acc}^{\mathcal{R}}_{i,j}\}^{|\mathcal{R}_i|}_{j=1})\}^{|I|}_{i=1}$
for all $i \in [|I|]$ **do**
  for all $t \in \{0,1\}$ **do**
    $(\mathsf{tag}_{t,i}, \mathsf{sk}^{\mathcal{S}}_{t,i}, a^{\mathcal{S}}_{t,i}, \mathsf{ty}^{\mathcal{S}}_{t,i}, \mathsf{ck}^{\mathcal{S}}_{t,i}) := \mathsf{TryReceive}(\mathsf{acc}^{\mathcal{R}}_{t,j_{t,i}})$
    $\mathcal{S}_t[i] = (\mathsf{tag}_{t,i}, j_{t,i}, \mathsf{sk}^{\mathcal{S}}_{t,i}, a^{\mathcal{S}}_{t,i}, \mathsf{ty}^{\mathcal{S}}_{t,i}, \mathsf{ck}^{\mathcal{S}}_{t,i})$
    $\mathcal{R}[i] = \{\mathsf{acc}^{\mathcal{R}}_{i,j}\}^{|\mathcal{R}_i|}_{j=1}$
  if $\mathsf{tag}_{0,i} \neq \mathsf{tag}_{1,i} \wedge \{(\mathsf{tag}_{t,i}, \cdot)\}^1_{t=0} \cap \mathsf{Offrd} \neq \emptyset$ **then return** o
parse J as $\{\{(k^{\mathcal{T}}_{t,j}, a^{\mathcal{T}}_{t,j}, \mathsf{ty}^{\mathcal{T}}_{t,j})\}^1_{t=0}\}^{|J|}_{j=1}$
for all $j \in [|J|]$ **do**
  for all $t \in \{0,1\}$ **do**
    $(\mathsf{acc}^{\mathcal{T}}_{t,j}, \mathsf{ck}^{\mathcal{T}}_{t,j}) := \mathsf{OTGen}(\mathsf{LTP}[k^{\mathcal{T}}_{t,j}], a^{\mathcal{T}}_{t,j}, \mathsf{ty}^{\mathcal{T}}_{t,j})$
    $\mathcal{T}_t[j] := (\mathsf{acc}^{\mathcal{T}}_{t,j}, a^{\mathcal{T}}_{t,j}, \mathsf{ty}^{\mathcal{T}}_{t,j}, \mathsf{ck}^{\mathcal{T}}_{t,j})$
for all $t \in \{0,1\}$ **do**
  $\mathfrak{o}_t \leftarrow \mathsf{Merge}(\mathfrak{o}, \mathsf{Offer}(\mathcal{S}_t, \mathcal{R}, \mathcal{T}_t))$
  $\mathsf{off}_t := \mathsf{off}(\mathcal{S}_t, R, \mathcal{T}_t) \cup \mathsf{off}$
  $\mathsf{tx}_t := \mathsf{tx}(\mathsf{off}_t)$
  $\mathsf{t}_t \leftarrow \mathsf{Seal}(\mathsf{off}_t, \mathfrak{o}_t)$
  if $\mathsf{VfOffer}(\mathsf{off}_t, \mathfrak{o}_t) = 0 \vee \mathsf{VfTx}(\mathsf{tx}_t, \mathsf{t}_t) = 0$ **then return** o
$b' \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{tx}_b, \mathsf{t}_b)$ **return** b'

Definition 6.6 (Transaction Privacy)**.** *A SwapCT has private transactions, if the participants are able to share messages by an anonymous broadcast, and if for all PPT adversaries $\mathcal{A}$ and all positive integers $\alpha, \beta \in \mathsf{poly}(\lambda)$ it holds that*

$$\left|\mathsf{Pr}[\mathsf{TxPv}^0_{\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta) = 1] - \mathsf{Pr}[\mathsf{TxPv}^1_{\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta) = 1]\right| \leq \mathsf{negl}(\lambda)$$

with $\mathsf{TxPv}^b_{\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta)$ *defined in Algorithm 6.7.*

6.5 CONSTRUCTION OF COMPONENTS

We require a protocol to anonymously authorize spending from a set of ring accounts $\{(\mathsf{pk}_i, \mathsf{com}_i)\}_{i=1}^{|\mathcal{R}|}$ without revealing the true source $(\mathsf{pk}_j, \mathsf{com}_j)$ and prevent double-spending. Especially the creator of the authorization does not necessarily know the secret keys of all the ring accounts nor interacts with the parties holding the secret keys. A ring signature solves exactly this problem to sign a message without revealing the true secret key sk used. In addition, we require linkability if the same secret key and thereby the same account was used in two different ring signatures. This is achieved by publishing a tag which is anonymously bound to the public key pk_j. If two signatures have the same tag, they were created by the same secret key. This is known as a tagged ring signature scheme TRS, similar to the MLSAG scheme in Monero [AH18]. The TRS is parameterized with a tagging scheme TAG and a typed homomorphic commitment scheme THC. We define this in the form of a SoK parameterized with the following language $\mathcal{L}^{\mathsf{ring}}$

$$:= \begin{cases} \mathsf{stmt} = (\{(\mathsf{pk}_i, \mathsf{com}_i)\}_{i=1}^{|\mathcal{R}|}, \mathsf{tag}, \mathsf{com}') : \\ \exists \mathsf{wit} = (j, \mathsf{sk}, a, \mathsf{ty}, \mathsf{ck}, \mathsf{ck}') \text{ s.t.} \\ \mathsf{pk}_j = \mathsf{TagKGen}(\mathsf{sk}), \quad \mathsf{com}_j = \mathsf{Commit}(\mathsf{ty}, a; \mathsf{ck}) \\ \mathsf{tag} = \mathsf{TagEval}(\mathsf{sk}), \quad \mathsf{com}' = \mathsf{Commit}(\mathsf{ty}, a; \mathsf{ck}') \end{cases}$$

Thereby, we assure that the signer knows at least one secret key sk of the ring accounts, and the tag matches this account j. The TRS also shows that a commitment com' commits to a type ty and amount a, which is the same as the amount and type in com_j, referenced by tag but has a different coin key ck' to hide the link to com_j. From $\mathcal{L}^{\mathsf{ring}}$, we see that given a binding THC scheme and a secure TAG scheme, the SoK ring signature is set anonymous and assures equal values in com_j and com'.

6.5.1 *Instantiation*

We instantiate the tagged ring signature by specifying the parameters of the previously defined efficient SoK. Using the concrete instantiations for THC

$(\mathsf{com} := (T, V))$ with Theorem 2.1 and TAG, we get the following language $\mathcal{L}^{\mathsf{ring}}{}_{\mathsf{THC,TAG}}$

$$:= \begin{cases} \mathsf{stmt} = (\{(\mathsf{pk}_i, (T_i, V_i))\}_{i=1}^{|\mathcal{R}|}, \mathsf{tag}, (T', V')) : \\ \exists \mathsf{wit} = (j, x, a, \mathsf{ty}, (r, s), (r's')) \text{ s.t.} \\ \vec{\mathsf{pk}}^{\circ \vec{e}_j} = H^x, \mathsf{tag} = G^{x^{-1}}, \prod (\vec{T} \cdot T'^{-1})^{\circ \vec{e}_j} = G^{\phi_1} \\ \prod (\vec{V} \cdot V'^{-1})^{\circ \vec{e}_j} = G^{\phi_2}, \vec{e}_j \text{ unit vector }, |\vec{e}_j| = |\mathcal{R}| \end{cases}$$

Given the challenge variables u, v from the extended Bulletproof system, we compress the conditions into $\vec{K}' := \vec{\mathsf{pk}} \circ (\vec{T} \cdot T'^{-1})^{\circ u} \circ (\vec{V} \cdot V'^{-1})^{\circ u^2}$. To satisfy $\prod \vec{K}^{\circ \vec{c}_{L,1}} = I$ and check for a correct tag, we extend $\vec{K}'$ with tag, G, H. The encoding for $\vec{c}_{L,1}$ is chosen appropriately with $\xi = -u\phi_1 - u^2\phi_2 - u^3 x^{-1}$ with $\phi_1(r, r') = r - r'$ and $\phi_2(a, r, s, r', s') = ar + s - ar' - s'$. It combines to

$$\begin{aligned} K &:= (\mathsf{tag}^{u^3} \| G \| \ H \ \| \quad \vec{K}' \quad) \\ \vec{c}_L &:= (\ \ 1 \ \ \| \xi \| -x \| \quad \vec{e}_j \quad \| \phi_1 \| \phi_2) \\ \vec{c}_R &:= (\ \ 0 \ \ \| 0 \| x^{-1} \| \vec{e}_j - \vec{1}^{|\mathcal{R}|} \| \ 0 \ \| \ 0 \ ) \end{aligned}$$

To enforce correct witness encoding, we define inner product relations. A constraint is parameterized by the variables u and v as well as a new challenge y:

$$\begin{array}{l|l} \vec{\mathsf{v}}_0 := (0 \| 0 \| \ 0 \ \| \vec{y}^{|\mathcal{R}|} \| 0 \| \ 0 \ ) & \langle \vec{c}_L, \vec{c}_R \circ \vec{\mathsf{v}}_0 \rangle = 0 \\ \vec{\mathsf{v}}_1 := (0 \| 0 \| \ 0 \ \| \vec{y}^{|\mathcal{R}|} \| 0 \| \ 0 \ ) & \langle \vec{c}_L - \vec{c}_R - \vec{1}^m, \vec{\mathsf{v}}_1 \rangle = 0 \\ \vec{\mathsf{v}}_2 := (y \| 0 \| \ 0 \ \| 1^{|\mathcal{R}|} \| 0 \| \ 0 \ ) & \langle \vec{c}_L, \vec{\mathsf{v}}_2 \rangle = \langle \vec{1}^2, \vec{y}^2 \rangle \\ \vec{\mathsf{v}}_3 := (0 \| 1 \| \ 0 \ \| \vec{0}^{|\mathcal{R}|} \| u \| u^2) & \\ \vec{\mathsf{v}}'_3 := (0 \| 0 \| u^3 \| \vec{0}^{|\mathcal{R}|} \| 0 \| \ 0 \ ) & \langle \vec{c}_L, \vec{\mathsf{v}}_3 \rangle + \langle \vec{c}_R, \vec{\mathsf{v}}'_3 \rangle = 0 \\ \vec{\mathsf{v}}_4 := (0 \| 0 \| -y \| \vec{0}^{|\mathcal{R}|} \| 0 \| \ 0 \ ) & \langle \vec{c}_L, \vec{c}_R \circ \vec{\mathsf{v}}_4 \rangle = y \end{array}$$

Using these parameters, we get an efficient SoK for the Tagged Ring Signature which has logarithmic communication size in the members of the ring allowing for large anonymity sets with small proof sizes.

Theorem 6.1 (SoK Signatures)**.** *Given the parameters above, the resulting protocol is perfectly complete, perfectly special honest-verifier zero-knowledge and logarithmic round argument of knowledge scheme for $\mathcal{L}^{\mathsf{ring}}$. Given witness-extended emulation and computationally unique responses, it is transformable to a perfectly complete, extractable, perfectly simulatable signature of knowledge for the language and any message $m \in \{0,1\}^*$ using Fiat-Shamir [FS] which holds for multiple rounds [DFM20, Thm. 23]. As the simulator and an extraction proofs follow the same structure as Omniring proofs we refer the reader to Lai et al. [LRR+19].*

6.5.2 *SwapCT*

With a tagging scheme (Section 2.11.1), a tagged ring signature (TRS), a seal signature for asset conservation (Chapter 4), the SwapCT construction Ξ is the interaction of the following algorithms. Intuitively, we create accounts with the public key of the tagging scheme and store the amount and type in a commitment. For each input of an offer, the real account is hidden in a set of ring accounts and a tagged ring signature assures that the published tag belongs to the account from which the amount and type is spent. To decouple the sender anonymity set from the remainder of the transaction, we create an intermediate, randomized commitment with a copy of the input values. Output accounts are derived from the recipients long term public key. To combine inputs with outputs, we use our anonymously aggregatable signature scheme which enables the simple merging of offers. Finally, an offer is sealed by the seal signature to become a verifiable transaction to be persisted.
For a detailed description, let χ be the key space of TAG and $\mathbb{R}$ the THC randomness space. Let $\mathfrak{h} : \{0,1\}^* \to \chi$ be a random oracle. Setup (Algorithm 6.8) generates the public parameters of each component by calling their setup functions.

Algorithm 6.8 $\mathsf{Setup}(1^\lambda, 1^\alpha, 1^\beta)$

$\mathsf{pp}_{\mathsf{THC}} \leftarrow \mathsf{ComSetup}(1^\lambda)$
$\mathsf{pp}_{\mathsf{PKE}} \leftarrow \mathsf{PKESetup}(1^\lambda)$
$\mathsf{pp}_{\mathsf{TAG}} \leftarrow \mathsf{TagSetup}(1^\lambda)$
$\mathsf{pp}_{\mathsf{AS}} \leftarrow \mathsf{AS.Setup}(1^\lambda, \mathcal{L}^{\mathsf{ring}})$
$\mathsf{pp}_{\mathsf{seal}} \leftarrow \mathsf{SoK}[\mathcal{L}_\emptyset]\mathsf{Setup}(1^\lambda, 1^\alpha, 1^\beta, \mathsf{pp}_{\mathsf{THC}})$
$\mathsf{pp} := (\mathsf{pp}_{\mathsf{THC}}, \mathsf{pp}_{\mathsf{PKE}}, \mathsf{pp}_{\mathsf{TAG}}, \mathsf{pp}_{\mathsf{AS}}, \mathsf{pp}_{\mathsf{seal}})$
return pp

To participate, each entity generates a long term key $(\mathsf{ltp}, \mathsf{lts})$ with KeyGen (Algorithm 6.9) that consists of two key pairs of the PKE scheme: $(\mathsf{vpk}, \mathsf{vsk})$ is used for access to received amounts and $(\mathsf{apk}, \mathsf{ask})$ is used to recover the authorization key to spend the received funds. The long term credentials further include a key pair $(\bar{\mathsf{sk}}, \bar{\mathsf{pk}})$ from the TAG to later calculate tags for each derived account.

Algorithm 6.9 $\mathsf{KeyGen}()$

$(\mathsf{vpk}, \mathsf{vsk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}_{\mathsf{PKE}})$
$(\mathsf{apk}, \mathsf{ask}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}_{\mathsf{PKE}})$
$\bar{\mathsf{sk}} := x \xleftarrow{\$} \chi, \bar{\mathsf{pk}} := X \leftarrow \mathsf{TagKGen}(x)$
$\mathsf{ltp} := (\mathsf{vpk}, \mathsf{apk}, \bar{\mathsf{pk}})$
$\mathsf{lts} := (\mathsf{vsk}, \mathsf{ask}, \bar{\mathsf{sk}})$
return $(\mathsf{ltp}, \mathsf{lts})$

The different token types available in a SwapCT system are not specified a priori but are dynamically added. A new type ty is generated by specifying a

new unique name and using ComTypeGen of the commitment scheme as described in TypeGen (Algorithm 6.10). The mere specification of a type ty is not sufficient to introduce the new type into circulation. To allow trading with this new type, a new account acc is derived from a long term public key ltp by OTGen (Algorithm 6.11). The acc is piggybacked onto a regular transaction to pay the registration fee. The matching lts is then allowed to spend the newly minted tokens of type ty. The uniqueness of the name in such a type registration must be ensured by the consensus mechanism.

Algorithm 6.10 TypeGen(name)

ty := ComTypeGen(name) **return** ty

The one-time account generation is used in subsequent transactions to specify the outputs. OTGen generates a random ephemeral key ek and uses it to generate a public key pk for which only the recipient can recover the secret key sk. A THC com is created to the amount a and type ty with a random coin key $\mathsf{ck} \in \mathbb{R}$. Finally, the secret values ek, ty, a, ck are encrypted under the recipient's keys apk, vpk to be decrypted with ask, vsk recovering the tokens. The structure of the one-time accounts is very similar to the Omniring construction apart from using a typed homomorphic commitment.

Algorithm 6.11 OTGen(ltp, a, ty)

parse ltp as (vpk, apk, $\bar{\mathsf{pk}}$)
$\mathsf{ek} \xleftarrow{\$} \{0,1\}^\lambda$
$\mathsf{ck} \xleftarrow{\$} \mathbb{R}$
$s := \mathfrak{h}(\mathsf{ltp}, \mathsf{ek})$
$\mathsf{pk} := \bar{\mathsf{pk}} \cdot \mathsf{TagKGen}(s)$
$\mathsf{com} := \mathsf{Commit}(\mathsf{ty}, a; \mathsf{ck})$
$\tilde{\mathsf{ek}} \leftarrow \mathsf{Enc}(\mathsf{apk}, (\mathsf{pk}, \mathsf{com}), \mathsf{ek})$
$\tilde{\mathsf{ck}} \leftarrow \mathsf{Enc}(\mathsf{vpk}, (\mathsf{pk}, \mathsf{com}), (\mathsf{ty}, a, \mathsf{ck}))$
$\mathsf{acc} = (\mathsf{pk}, \mathsf{com}, \tilde{\mathsf{ek}}, \tilde{\mathsf{ck}})$
return (acc, ck)

The owner of the long term secret key lts is able to receive an account acc by Receive (Algorithm 6.12). This is again similar to Omniring with the exception of a different commitment. First the recipient decrypts the ciphertexts $\tilde{\mathsf{ek}}, \tilde{\mathsf{ck}}$ with the labeled encryption scheme to get ek, the amount a, type ty and ck and then derives the tag for this account from the tagging scheme.
With all accounts set up, Offer (Algorithm 6.13) ensures sender anonymity by creating a temporary commitment com'_i for each input $i \in [|\mathcal{S}|]$ with fresh randomness $\mathsf{ck}'_i \in \mathbb{R}$. It then calls the aggregatable signature scheme parameterized with the tagged ring signature language $\mathcal{L}^{\mathsf{ring}}$. The SoK for TRS requires the temporary commitment com'_i as well as the ring accounts as statement, which is the input for the AS.Sign function. The transaction output accounts $\mathsf{acc}_i^{\mathcal{T}}$ are used as messages. An offer then consists of $\mathsf{off}(\mathcal{S}, \mathcal{R}, \mathcal{T})$, defined in Eq (6.1), and $\mathfrak{o} = (\{\mathfrak{r}_i, \mathsf{com}'_i, \mathsf{ck}'_i\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a})$.

Algorithm 6.12 Receive(acc, lts)

parse acc as $(\mathsf{pk}, \mathsf{com}, \tilde{\mathsf{ek}}, \tilde{\mathsf{ck}})$
parse lts as $(\mathsf{vsk}, \mathsf{ask}, \bar{\mathsf{sk}})$
$\mathsf{ek} = \mathsf{Dec}(\mathsf{ask}, (\mathsf{pk}, \mathsf{com}), \tilde{\mathsf{ek}})$
$(\mathsf{ty}, a, \mathsf{ck}) = \mathsf{Dec}(\mathsf{vsk}, (\mathsf{pk}, \mathsf{com}), \tilde{\mathsf{ck}})$
$s := \mathfrak{h}(\mathsf{ltp}, \mathsf{ek})$
$\mathsf{sk} := \bar{\mathsf{sk}} + s$
$\mathsf{pk}' = \mathsf{TagKGen}(sk)$
$\mathsf{com}' = \mathsf{Commit}(\mathsf{ty}, a; \mathsf{ck}))$
if $(\mathsf{pk}, \mathsf{com}) \neq (\mathsf{pk}', \mathsf{com}')$ **then**
  return $\perp$
$\mathsf{tag} \leftarrow \mathsf{TagEval}(\mathsf{sk})$
return $(\mathsf{tag}, \mathsf{sk}, a, \mathsf{ty}, \mathsf{ck})$

Algorithm 6.13 Offer($\mathcal{S}, \mathcal{R}, \mathcal{T}$)

parse $\mathcal{S}$ as $\{(\mathsf{tag}_i, j_i, \mathsf{sk}_i, a_i^{\mathcal{S}}, \mathsf{ty}_i^{\mathcal{S}}, \mathsf{ck}_i^{\mathcal{S}})\}_{i=1}^{|\mathcal{S}|}$
parse $\mathcal{R}$ as $\{\{\mathsf{acc}_{i,j}^{\mathcal{R}} := (\mathsf{pk}_{i,j}^{\mathcal{R}}, \mathsf{com}_{i,j}^{\mathcal{R}}, \cdot)\}_{j=1}^{|\mathcal{R}_i|}\}_{i=1}^{|\mathcal{S}|}$
parse $\mathcal{T}$ as $\{(\mathsf{acc}_i^{\mathcal{T}}, a_i^{\mathcal{T}}, \mathsf{ty}_i^{\mathcal{T}}, \mathsf{ck}_i^{\mathcal{T}})\}_{i=1}^{|\mathcal{T}|}$
$\{\mathsf{ck}_i' \xleftarrow{\$} \mathbb{R}, \mathsf{com}_i' \leftarrow \mathsf{Commit}(\mathsf{ty}_i^{\mathcal{S}}, a_i^{\mathcal{S}}; \mathsf{ck}_i')\}_{i=1}^{|\mathcal{S}|}$
$\{\mathsf{stmt}_i = \left(\{(\mathsf{pk}_{i,j}^{\mathcal{R}}, \mathsf{com}_{i,j}^{\mathcal{R}})\}_{j=1}^{|\mathcal{R}_i|}, \mathsf{tag}_i, \mathsf{com}_i'\right)\}_{i=1}^{|\mathcal{S}|}$
$\{\mathsf{wit}_i = (j_i, \mathsf{sk}_i, a_i^{\mathcal{S}}, \mathsf{ty}_i^{\mathcal{S}}, \mathsf{ck}_i^{\mathcal{S}}, \mathsf{ck}_i')\}_{i=1}^{|\mathcal{S}|}$
$(\{\mathfrak{r}_i\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a}) \leftarrow \mathsf{AS.Sign}_{\mathsf{SoK}[\mathcal{L}^{\mathsf{ring}}]}(\{(\mathsf{stmt}_i, \mathsf{wit}_i)\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{acc}_j^{\mathcal{T}}\}_{j=1}^{|\mathcal{T}|})$
return $(\{\mathfrak{r}_i, \mathsf{com}_i', \mathsf{ck}_i'\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a})$

With the underlying aggregatable signature scheme AS, merging offers $\mathfrak{o}_1, \mathfrak{o}_2$ with Merge (Algorithm 6.14) directly translates to merging aggregatable signatures $\mathfrak{a}_1, \mathfrak{a}_2$. The offers along with the temporary commitments com′ and authorization signatures $\mathfrak{r}$ are combined by using their union.

Algorithm 6.14 Merge($\mathfrak{o}_1, \mathfrak{o}_2$)

parse $\mathfrak{o}_1$ as $(\{(\mathfrak{r}_i^1, \mathsf{com'}_i^1, \mathsf{ck'}_i^1)\}_{i=1}^{|\mathcal{S}_1|}, \mathfrak{a}_1)$
parse $\mathfrak{o}_2$ as $(\{(\mathfrak{r}_i^2, \mathsf{com'}_i^2, \mathsf{ck'}_i^2)\}_{i=1}^{|\mathcal{S}_2|}, \mathfrak{a}_2)$
$\mathfrak{a} \leftarrow \mathsf{AS.Merge}(\mathfrak{a}_1, \mathfrak{a}_2)$
return $(\{(\mathfrak{r}_i^1, \mathsf{com'}_i^1, \mathsf{ck'}_i^1)\}_{i=1}^{|\mathcal{S}_1|} \cup \{(\mathfrak{r}_i^2, \mathsf{com'}_i^2, \mathsf{ck'}_i^2)\}_{i=1}^{|\mathcal{S}_2|}, \mathfrak{a})$

Offers are verifiable by VfOffer (Algorithm 6.15) which checks that the commitments $\mathsf{com}', \mathsf{com}^{\mathcal{T}}$ agree with the opened values $a^{\mathcal{S}}, \mathsf{ty}^{\mathcal{S}}, a^{\mathcal{T}}, \mathsf{ty}^{\mathcal{T}}$ and verifies $\mathfrak{o}$ with AS.Verify.
Once an offer is balanced and valid, it can be sealed. Seal (Algorithm 6.16) uses a SoK with the seal language $\mathcal{L}_\emptyset$ with a message of $\mathsf{tx} = \mathsf{tx}(\mathsf{off})$ and a statement $\mathsf{stmt}(\mathsf{tx}) := (\{\mathsf{com}_i'\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{com}_j^{\mathcal{T}}\}_{j=1}^{|\mathcal{T}|})$ containing the relevant in-

Algorithm 6.15 $\mathsf{VfOffer}(\mathsf{off}, \mathfrak{o})$

parse off as $\left(\{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|}, \{\{\mathsf{acc}_{i,j}^{\mathcal{R}}\}_{j=1}^{|\mathcal{R}_i|}\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{acc}_i^{\mathcal{T}}, a_i^{\mathcal{T}}, \mathsf{ty}_i^{\mathcal{T}}, \mathsf{ck}_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}\right)$
parse $\mathsf{acc}_i^{\mathcal{T}}$ as $(\mathsf{pk}_i, \mathsf{com}_i^{\mathcal{T}}, \tilde{\mathsf{ek}}, \tilde{\mathsf{ck}})$ and $\mathfrak{o}$ as $(\{\mathfrak{r}_i, \mathsf{com}'_i, \mathsf{ck}'_i\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a})$
for all $i \in [|\mathcal{S}|]$ **do**
  if $\mathsf{com}'_i \neq \mathsf{Commit}(\mathsf{ty}_i, a_i; \mathsf{ck}'_i)$ **then return** $\bot$
  $\mathsf{stmt}_i := (\{(\mathsf{pk}_{i,j}^{\mathcal{R}}, \mathsf{com}_{i,j}^{\mathcal{R}})\}_{j=1}^{|\mathcal{R}_i|}, \mathsf{tag}_i, \mathsf{com}'_i)$
for all $j \in [|\mathcal{T}|]$ **do**
  if $\mathsf{com}_j^{\mathcal{T}} \neq \mathsf{Commit}(\mathsf{ty}_j^{\mathcal{T}}, a_j^{\mathcal{T}}; \mathsf{ck}_j^{\mathcal{T}})$ **then return** $\bot$
return $\mathsf{AS.Verify}_{\mathsf{SoK}[\mathcal{L}^{\mathsf{ring}}]}(\{(\mathfrak{r}_i, \mathsf{stmt}_i)\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a}, \{\mathsf{acc}_j^{\mathcal{T}}\}_{j=1}^{|\mathcal{T}|})$

termediate commitments com' and output commitments $\mathsf{com}^{\mathcal{T}}$. The matching witness is the set of committed values: $\mathsf{wit}(\mathcal{S}, \{\mathsf{com}'_i, \mathsf{ck}'_i\}_{i=1}^{|\mathcal{S}|}, \mathcal{T}) = (\{\mathsf{ty}_i^{\mathcal{S}}, a_i^{\mathcal{S}}, \mathsf{ck}_i^{\mathcal{S}}\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{ty}_j^{\mathcal{T}}, a_j^{\mathcal{T}}, \mathsf{ck}_j^{\mathcal{T}}\}_{j=1}^{|\mathcal{T}|})$. The seal algorithm operates on the temporary commitments com' and keeps the real sender hidden in the set of ring accounts. The TRS ensures that com' commits to the same type and value as the real input. The seal signature $\mathfrak{t}$ then contains the SoK signature $\mathfrak{s}$ and all parts of the offer signature $\mathfrak{o}$, without the temporary coin keys ck'.

Algorithm 6.16 $\mathsf{Seal}(\mathsf{off}, \mathfrak{o})$

if $\mathsf{VfOffer}(\mathsf{off}, \mathfrak{o}) \neq 1$ **then return** $\bot$
$\mathsf{tx} = \mathsf{tx}(\mathsf{off})$
parse $\mathfrak{o}$ as $(\{\mathfrak{r}_i, \mathsf{com}'_i, \mathsf{ck}'_i\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a})$ and off as $(\mathcal{S}, \mathcal{R}, \mathcal{T})$
$\mathfrak{s} \leftarrow \mathsf{SoK}[\mathcal{L}_\emptyset]\mathsf{Sign}(\mathsf{stmt}(\mathsf{tx}), \mathsf{wit}(\mathcal{S}, \{\mathsf{com}'_i, \mathsf{ck}'_i\}_{i=1}^{|\mathcal{S}|}, \mathcal{T}), \mathsf{tx})$
return $\mathfrak{t} = (\mathfrak{s}, \{\mathfrak{r}_i, \mathsf{com}'_i\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a})$

Many public ledgers use financial incentives. We suggest using a common *native* type for all incentives (transactions fees and mining rewards) as it is equally valued by every participant. A block reward is generated by $(\mathsf{acc}_{\mathrm{reward}}, \mathsf{ck}) \leftarrow \mathsf{OTGen}(\mathsf{ltp}_{\mathrm{miner}}, a_{\mathrm{reward}}, \mathsf{ty}_{\mathrm{native}})$ and $\mathsf{acc}_{\mathrm{reward}}$ is included in the block. A transaction fee is handled by generating a commitment $\mathsf{com}_{\mathsf{fee}} = \mathsf{Commit}(\mathsf{ty}_{\mathrm{native}}, a_{\mathrm{fee}}, r)$ and appending it to the transaction with the plaintext values a_{fee}, r. A verifier checks the commitment and then appends it to the $\mathcal{L}_\emptyset$ statement of output commitments $\{\mathsf{com}_j^{\mathcal{T}}\}_{j=1}^{|\mathcal{T}|} \cup \{\mathsf{com}_{\mathsf{fee}}\}$. This assures that the inputs provide enough tokens in the native type to satisfy all regular outputs and the fee. In a swap, offers may include a small surplus of native tokens not claimed by any output. The merger creates a single commitment to the sum of surplus from each offer and proceeds as explained above. Exchanges may request operation fees. Therefore they accept only offers which have the requested surplus to be claimed on merging in a regular output to the exchange. Regarding the transaction size, this requires one additional input and output for each merged transaction by the exchange.

A transaction is verified with VfTx (Algorithm 6.17) which proceeds similarly to VfOffer except that it does not verify the openings of the commitments com'_i and $\mathsf{com}_j^{\mathcal{T}}$ but only the aggregatable signature $\mathfrak{a}$. Instead, the commitments are checked by verifying the seal signature $\mathfrak{s}$. ChkAcc (Algorithm 6.18) and ChkTag (Algorithm 6.19) verify the consistency of the inputs by verifying the THC and TAG schemes.

Algorithm 6.17 $\mathsf{VfTx}(\mathsf{tx}, \mathfrak{t})$

parse tx as $\left(\{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|}, \{\{\mathsf{acc}_{i,j}^{\mathcal{R}}\}_{j=1}^{|\mathcal{R}_i|}\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{acc}_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}\right)$
parse $\mathsf{acc}_{i,j}^{\mathcal{R}}$ as $(\mathsf{pk}_{i,j}^{\mathcal{R}}, \mathsf{com}_{i,j}^{\mathcal{R}}, \cdot)$
parse $\mathsf{acc}_i^{\mathcal{T}}$ as $(\mathsf{pk}_i, \mathsf{com}_i^{\mathcal{T}}, \cdot)$
parse $\mathfrak{t}$ as $(\mathfrak{s}, \{\mathfrak{r}_i, \mathsf{com}'_i\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a})$
$\forall i \in [|\mathcal{S}|] : \mathsf{stmt}_i := (\{(\mathsf{pk}_{i,j}^{\mathcal{R}}, \mathsf{com}_{i,j}^{\mathcal{R}})\}_{j=1}^{|\mathcal{R}_i|}, \mathsf{tag}_i, \mathsf{com}'_i)$
$b_0 := |\mathcal{T}| < 2^{\alpha}$
$b_1 := \mathsf{AS.Verify}_{\mathsf{SoK}[\mathcal{L}^{\mathsf{ring}}]}(\{(\mathfrak{r}_i, \mathsf{stmt}_i)\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a}, \{\mathsf{acc}_j^{\mathcal{T}}\}_{j=1}^{|\mathcal{T}|})$
$b_2 := \mathsf{SoK}[\mathcal{L}_{\emptyset}]\mathsf{Verify}(\mathfrak{s}, \mathsf{stmt}(\mathsf{tx}), \mathsf{tx}))$
return $b := b_0 \wedge b_1 \wedge b_2$

Algorithm 6.18 $\mathsf{ChkAcc}(\mathsf{acc}, \mathsf{ck}, a, \mathsf{ty})$

parse acc as $(\mathsf{pk}, \mathsf{com}, \tilde{\mathsf{ek}}, \tilde{\mathsf{ck}})$
return $\mathsf{com} = \mathsf{Commit}(\mathsf{ty}, a; \mathsf{ck})$

Algorithm 6.19 $\mathsf{ChkTag}(\mathsf{acc}, \mathsf{sk}, \mathsf{tag})$

parse acc as $(\mathsf{pk}, \mathsf{com}, \tilde{\mathsf{ek}}, \tilde{\mathsf{ck}})$
return $\mathsf{tag} = \mathsf{TagEval}(\mathsf{sk}) \wedge \mathsf{pk} = \mathsf{TagKGen}(\mathsf{sk})$

6.6 ANALYSIS

The construction is correct and satisfies to the following security properties.

Theorem 6.2 (Non Slanderability). *If* AS *is extractable and simulatable,* TAG *is related-input one-way, and* $\mathfrak{h}$ *is modeled as a random oracle, then the construction* Ξ *is non-slanderable.*

As we use the same TAG scheme as Omniring, and their security proof for non-slanderability requires only a simulator for the transaction signature, Theorem 6.2 holds in our setting, as our new transaction signature is simulatable by $\mathsf{AS.Sim}_{\mathcal{L}^{\mathsf{ring}}}$ (Theorem 3.2) and $\mathsf{SoK}[\mathcal{L}_{\emptyset}]\mathsf{Sim}$.

Theorem 6.3 (Theft Prevention). *If* THC *is binding and* AS *is secure and* Ξ *is non-slanderable, it prevents theft.*

Proof of Theorem 6.3 (Theft). For ChkTag to be computationally binding, assume a PPT adversary which outputs two valid openings $(\mathsf{acc}, \mathsf{sk}, \mathsf{tag}, \mathsf{sk}', \mathsf{tag}')$. The validity requires $\mathsf{pk} = \mathsf{TagKGen}(\mathsf{sk}) = \mathsf{TagKGen}(\mathsf{sk}')$ which forces $\mathsf{sk} = \mathsf{sk}'$ because TagKGen is a bijection. As TagEval is deterministic, $\mathsf{tag} = \mathsf{tag}'$, contradicting $(\mathsf{sk}, \mathsf{tag}) \neq (\mathsf{sk}', \mathsf{tag}')$
ChkAcc is binding because it requires $\mathsf{Commit}(\mathsf{ty}, a; \mathsf{ck}) = \mathsf{Commit}(\mathsf{ty}', a'; \mathsf{ck}')$ with $(\mathsf{ty}, a) \neq (\mathsf{ty}', a')$ which contradicts the THC binding property.
We show that an adversary cannot change a valid offer

$$\mathsf{off} := (\{\mathsf{tag}, \cdot\}_{i=1}^{|\mathcal{S}|}, \mathcal{R}, \{\mathsf{acc}_i^{\mathcal{T}}, \cdot\}_{i=1}^{|\mathcal{T}|}), \mathfrak{o} := (\{\mathfrak{r}_i, \cdot\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a})$$

to $\mathsf{off}' := (\{\mathsf{tag}', \cdot\}_{i=1}^{|\mathcal{S}'|}, \mathcal{R}', \{\mathsf{acc}'^{\mathcal{T}}_i, \cdot\}_{i=1}^{|\mathcal{T}'|}), \mathfrak{o}' := (\{\mathfrak{r}'_i, \cdot\}_{i=1}^{|\mathcal{S}'|}, \mathfrak{a}')$, such that it is valid $(\mathsf{VfOffer}(\mathsf{off}', \mathfrak{o}') = 1)$, reuses a tag from off $(\{\mathsf{tag}'_i\}_{i=1}^{|\mathcal{S}'|} \cap \{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|} \neq \emptyset)$ and changes or removes an output $\mathsf{acc}^{\mathcal{T}}$ from off.
Assume that some $\mathsf{tag}_* \in \{\mathsf{tag}_i\}_{i=1}^{|\mathcal{S}|}$ from off is reused in off' $(\mathsf{tag}_* \in \{\mathsf{tag}'_i\}_{i=1}^{|\mathcal{S}'|})$ and an output $\mathsf{acc}_*^{\mathcal{T}} \in \{\mathsf{acc}_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}$ was modified or removed $(\mathsf{acc}_*^{\mathcal{T}} \notin \{\mathsf{acc}'^{\mathcal{T}}_i\}_{i=1}^{|\mathcal{T}'|})$. As the offer off' is valid, this implies that $\mathfrak{a}'$ is valid by AS.Verify in VfOffer. The security of the AS scheme from Definition 3.3 then implies that no signature was reused $(\{\mathfrak{r}_i\}_{i=1}^{|\mathcal{S}|} \cap \{\mathfrak{r}'_i,\}_{i=1}^{|\mathcal{S}|} = \emptyset)$, as at least one output message, namely $\mathsf{acc}_*^{\mathcal{T}}$, was changed. An efficient adversary against Theorem 6.3 can be used to construct an efficient adversary against the security of $\mathsf{SoK}[\mathcal{L}^{\mathsf{ring}}]$ as the verification in $\mathsf{AS.Verify}_{\mathsf{SoK}[\mathcal{L}^{\mathsf{ring}}]}$ requires that for each $\mathsf{SoK}[\mathcal{L}^{\mathsf{ring}}]\mathsf{Verify}(\mathfrak{r}_i, \mathsf{stmt}_i, \cdot) = 1$. The non-slanderability of Theorem 6.2 prevents exactly this. □

Theorem 6.4 (Balance)**.** *If* THC *is binding and* $\mathsf{SoK}[\mathcal{L}_\emptyset]$ *is extractable and* Ξ *is non-slanderable the construction is balanced.*

Proof of Theorem 6.4 (Balance). To show the balance property, we proceed by constructing an efficient extractor $\mathcal{E}$. As $\mathsf{VfTx}(\mathsf{tx}, \mathsf{t}) = 1$ implies that $\mathsf{SoK}[\mathcal{L}_\emptyset]\mathsf{Verify}(\mathsf{t}, \mathsf{stmt}(\mathsf{tx}), \mathsf{tx}) = 1$. Then there exists an efficient extractor $\mathsf{SoK}[\mathcal{L}_\emptyset]\mathcal{E}_{\mathcal{A}}$ extracting a wit for $\mathsf{stmt}(\mathsf{tx})$. Parse the statement as

$$\mathsf{stmt} = (\{\mathsf{com}'_i\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{com}_j^{\mathcal{T}}\}_{j=1}^{|\mathcal{T}|})$$

and the witness as

$$\mathsf{wit} = (\{\mathsf{ty}_i^{\mathcal{S}}, a_i^{\mathcal{S}}, \mathsf{ck}'_i\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{ty}_j^{\mathcal{T}}, a_j^{\mathcal{T}}, \mathsf{ck}_j^{\mathcal{T}}\}_{j=1}^{|\mathcal{T}|})$$

where

$$\begin{aligned}
\forall i \in [|\mathcal{S}|] : \mathsf{com}'_i &= \mathsf{Commit}(\mathsf{ty}_i^{\mathcal{S}}, a_i^{\mathcal{S}}; \mathsf{ck}'_i) \\
\forall j \in [|\mathcal{T}|] : \mathsf{com}_j^{\mathcal{T}} &= \mathsf{Commit}(\mathsf{ty}_j^{\mathcal{T}}, a_j^{\mathcal{T}}; \mathsf{ck}_j^{\mathcal{T}}) \\
\forall \mathsf{ty} \in \{\mathsf{ty}_j^{\mathcal{T}}\}_{j=1}^{|\mathcal{T}|} : \sum \{a_i^{\mathcal{S}} | \mathsf{ty}_i^{\mathcal{S}} = \mathsf{ty}\}_{i=1}^{|\mathcal{S}|} &= \sum \{a_i^{\mathcal{T}} | \mathsf{ty}_i^{\mathcal{T}} = \mathsf{ty}\}_{i=1}^{|\mathcal{T}|}
\end{aligned}$$

holds. This directly implies the following conditions of the balance experiment:

– $\forall i \in [|\overleftarrow{\mathcal{T}}|], \mathsf{ChkAcc}(\mathsf{acc}_i^{\overleftarrow{\mathcal{T}}}, a_i^{\overleftarrow{\mathcal{T}}}, \mathsf{ty}_i^{\overleftarrow{\mathcal{T}}}, \mathsf{ck}_i^{\overleftarrow{\mathcal{T}}}) = 1$

– $\forall i \in [|\overleftarrow{\mathcal{T}}|], \mathsf{ty}_i^{\overleftarrow{\mathcal{T}}} \in \{\mathsf{ty}_j^{\mathcal{S}}\}_{j=1}^{|\mathcal{S}|}$

– and $\forall \mathsf{ty} \in \{\mathsf{ty}_i^{\overleftarrow{\mathcal{T}}}\}_{i=1}^{|\overleftarrow{\mathcal{T}}|}$:
$\sum\{a_i^{\mathcal{S}} | \mathsf{ty}_i^{\mathcal{S}} = \mathsf{ty}\}_{i=1}^{|\mathcal{S}|} = \sum\{a_i^{\overleftarrow{\mathcal{T}}} | \mathsf{ty}_i^{\overleftarrow{\mathcal{T}}} = \mathsf{ty}\}_{i=1}^{|\overleftarrow{\mathcal{T}}|}$.

The validity of $\mathsf{VfTx}(\mathsf{tx}, \mathfrak{t}) = 1$ additionaly requires

$$\mathsf{AS.Verify}_{\mathsf{SoK}[\mathcal{L}^{\mathsf{ring}}]}(\{(\mathfrak{r}_i, \mathsf{stmt}_i)\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a}, \{\mathsf{acc}_j^{\overleftarrow{\mathcal{T}}}\}_{j=1}^{|\overleftarrow{\mathcal{T}}|}) = 1.$$

This is only true, if for each $i \in [|\mathcal{S}|]$: $\mathsf{SoK}[\mathcal{L}^{\mathsf{ring}}]\mathsf{Verify}(\mathfrak{r}_i, \mathsf{stmt}_i, \cdot) = 1$ holds. Due to the extended witness emulation of SoKs, there exist efficient extractors $\mathsf{SoK}[\mathcal{L}^{\mathsf{ring}}]\mathcal{E}_{\mathcal{A},i}$ extracting wit_i for stmt_i. Parse the statements as $\mathsf{stmt}_i = (\{(\mathsf{pk}_{i,k}, \mathsf{com}_{i,k})\}_{k=1}^{|\mathcal{R}|}, \mathsf{tag}_i, \mathsf{com}_i')$ and the witnesses as $\mathsf{wit}_i = (j_i, \mathsf{sk}_i, a_i^{\mathcal{S}}, \mathsf{ty}_i^{\mathcal{S}}, \mathsf{ck}_i^{\mathcal{S}}, \mathsf{ck}_i')$. $\mathcal{L}^{\mathsf{ring}}$ then enforces

$$\mathsf{pk}_{j_i} = \mathsf{TagKGen}(\mathsf{sk}_i) \mathsf{tag}_i = \mathsf{TagEval}(\mathsf{sk}_i)$$
$$\mathsf{com}_{i,j_i} = \mathsf{Commit}(\mathsf{ty}_i, a_i; \mathsf{ck}_i) \mathsf{com}_i' = \mathsf{Commit}(\mathsf{ty}_i, a_i; \mathsf{ck}_i')$$

which implies the remaining conditions of the balance experiment, namely for each i,

– $\mathsf{ChkTag}(\mathsf{acc}_{i,j_i}^{\mathcal{R}}, \mathsf{sk}_i, \mathsf{tag}_i) = 1$ and

– $\mathsf{ChkAcc}(\mathsf{acc}_{i,j_i}^{\mathcal{R}}, a_i^{\mathcal{S}}, \mathsf{ty}_i^{\mathcal{S}}, \mathsf{ck}_i^{\mathcal{S}}) = 1$

hold. With a binding THC, intermediate commitments com_i' commit to the same witnesses $(a^{\mathcal{S}}, \mathsf{ty}^{\mathcal{S}})$ in Offer and Seal. Thereby, it holds that

$$\Pr[\mathsf{Balance}_{\mathcal{A},\mathcal{E}}(1^\lambda, 1^\alpha, 1^\beta) = 1] \leq \mathsf{negl}(\lambda).$$

□

Theorem 6.5 (Offer Privacy)**.** *If* THC *is hiding and binding,* PKE *is IND-CCA secure and key-private,* AS *is simulatable and private, and* TAG *is related-input pseudo random,* Ξ *has offer privacy.*

Proof of Theorem 6.5 (Offer Privacy). We use a series of hybrids to prove the offer privacy by progressing in indistinguishable steps from the experiment with $b = 0$ to $b = 1$. The hybrids are defined as:
$\overline{\mathsf{OHyb}_1}$ is the same as $\mathsf{OffPv}_{\mathcal{A}}^0$
$\overline{\mathsf{OHyb}_2}$ differs in that the aggregatable signature AS in Offer is simulated by $\overline{\mathsf{AS.Sim}}_{\mathcal{L}^{\mathsf{TRS}}}()$. The information available to the Adversary are off_0 which includes

$$\mathsf{off}_0 = \Big(\{\mathsf{tag}_{0,i}, a_{0,i}^{\mathcal{S}}, \mathsf{ty}_{0,i}^{\mathcal{S}}\}_{i=1}^{|\mathcal{S}|}, \{\{\mathsf{acc}_{i,j}^{\mathcal{R}}\}_{j=1}^{|\mathcal{R}_i|}\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{acc}_{0,i}^{\overleftarrow{\mathcal{T}}}, a_i^{\overleftarrow{\mathcal{T}}}, \mathsf{ty}_i^{\overleftarrow{\mathcal{T}}}, \mathsf{ck}_{0,i}^{\overleftarrow{\mathcal{T}}}\}_{i=1}^{|\overleftarrow{\mathcal{T}}|}\Big)$$

and the intermediate commitments with coin keys $\{\mathsf{com}'_{0,i}, \mathsf{ck}'_{0,i}\}_{i=1}^{|\mathcal{S}|}$. Everything else in the signature $\mathfrak{o}_0 = (\{(\mathfrak{r}_{0,i}, \mathsf{com}'_{0,i}, \mathsf{ck}'_{0,i})\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a}_0)$ is simulated. Values indepent of b are ignored, reducing the data to

$$\left(\{\mathsf{tag}_{0,i}\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{acc}_{0,i}^{\mathcal{T}}, \mathsf{ck}_{0,i}^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}\right)$$

and $\{\mathsf{com}'_{0,i}, \mathsf{ck}'_{0,i}\}_{i=1}^{|\mathcal{S}|}$

$\underline{\mathsf{OHyb}_3}$ changes the intermediate commitment to $\{\mathsf{com}'_{1,i}, \mathsf{ck}'_{1,i}\}_{i=1}^{|\mathcal{S}|}$, leaving

$$\left(\{\mathsf{tag}_{0,i}\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{acc}_{0,i}^{\mathcal{T}}, \mathsf{ck}_{0,i}^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}\right)$$

$\underline{\mathsf{OHyb}_4}$ changes the tags to the experiment with $b = 1$, resulting in

$$\left(\{\mathsf{tag}_{1,i}\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{acc}_{0,i}^{\mathcal{T}}, \mathsf{ck}_{0,i}^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}\right)$$

and $\{\mathsf{com}'_{1,i}, \mathsf{ck}'_{1,i}\}_{i=1}^{|\mathcal{S}|}$

$\underline{\mathsf{OHyb}_5}$: The output accounts consist of a public key and a commitment $\mathsf{acc}_{0,i}^{\mathcal{T}} = (\mathsf{pk}_{0,i}^{\mathcal{T}}, \mathsf{com}_{0,i}^{\mathcal{T}})$. First we change the public keys to

$$\left(\{\mathsf{tag}_{1,i}\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{pk}_{1,i}^{\mathcal{T}}, \mathsf{com}_{0,i}^{\mathcal{T}}, \mathsf{ck}_{0,i}^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}\right)$$

and $\{\mathsf{com}'_{1,i}, \mathsf{ck}'_{1,i}\}_{i=1}^{|\mathcal{S}|}$

$\underline{\mathsf{OHyb}_6}$ changes output commitments and coin keys to

$$\left(\{\mathsf{tag}_{1,i}\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{pk}_{1,i}^{\mathcal{T}}, \mathsf{com}_{1,i}^{\mathcal{T}}, \mathsf{ck}_{1,i}^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}\right)$$

and $\{\mathsf{com}'_{1,i}, \mathsf{ck}'_{1,i}\}_{i=1}^{|\mathcal{S}|}$

$\underline{\mathsf{OHyb}_7}$ reverts to the real signature AS.Sign instead of the simulated one, which results in $\mathsf{OffPv}_{\mathcal{A}}^1$.

We now show the indistinguishability of the hybrids.

$\underline{\mathsf{OHyb}_1 \equiv \mathsf{OHyb}_2}$ follows from the simulatability of AS.

$\underline{\mathsf{OHyb}_2 \equiv \mathsf{OHyb}_3}$: To show the equivalence, we define $|\mathcal{S}| + 1$ sub-hybrids where each changes one intermediate commitment. The first sub-hybrid is equal to $\mathsf{OHyb}_{2,0} = \mathsf{OHyb}_2$ and the last is $\mathsf{OHyb}_{2,|\mathcal{S}|} = \mathsf{OHyb}_3$. In $\mathsf{OHyb}_{2,l}$ the information available to the adversary is

$$\left(\{\mathsf{tag}_{0,i}\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{acc}_{0,i}^{\mathcal{T}}, \mathsf{ck}_{0,i}^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}\right)$$

and $\{\mathsf{com}'_{1,i}, \mathsf{ck}'_{1,i}\}_{i=1}^{l} \cup \{\mathsf{com}'_{0,i}, \mathsf{ck}'_{0,i}\}_{i=l+1}^{|\mathcal{S}|}$. To show that $\mathsf{OHyb}_{2,l-1} \equiv \mathsf{OHyb}_{2,l}$ we know that the amount $a_l^{\mathcal{S}}$ and type $\mathsf{ty}_l^{\mathcal{S}}$ committed to in $\mathsf{com}'_{k,l}$ are equal for both $k \in \{0, 1\}$. The coin keys $\mathsf{ck}'_{k,l}$ are distributed uniformly at random. Thereby, the commitment $\mathsf{com}'_{1,l}$ is fully defined.

$\underline{\mathsf{OHyb}_3 \approx_c \mathsf{OHyb}_4}$: To show the indistinguishability of the tags, we again define $|\mathcal{S}| + 1$ sub-hybrids with $\mathsf{OHyb}_{3,0} = \mathsf{OHyb}_3$ to $\mathsf{OHyb}_{3,|\mathcal{S}|} = \mathsf{OHyb}_4$. The information available to the adversary in $\mathsf{OHyb}_{3,l}$ is

$$\left(\{\mathsf{tag}_{0,i}\}_{i=1}^{l} \cup \{\mathsf{tag}_{1,i}\}_{i=l+1}^{|\mathcal{S}|}, \{\mathsf{acc}_{0,i}^{\mathcal{T}}, \mathsf{ck}_{0,i}^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}\right)$$

and $\{\mathsf{com}'_{1,i}, \mathsf{ck}'_{1,i}\}_{i=1}^{|\mathcal{S}|}$. The indistinguishability $\mathsf{OHyb}_{3,l-1} \approx_c \mathsf{OHyb}_{3,l}$ holds because TagEval is called with a uniformly random value $x + s$. According to the related-input pseudorandomness of TAG, $\mathsf{tag}_{0,l}$ and $\mathsf{tag}_{1,l}$ are computationally indistinguishable.

$\underline{\mathsf{OHyb}_4 \equiv \mathsf{OHyb}_5}$: To show the equivalence, we define $|\mathcal{J}| + 1$ sub-hybrids where each changes one public key of the account. The first sub-hybrid is equal to $\mathsf{OHyb}_{4,0} = \mathsf{OHyb}_4$ and the last is $\mathsf{OHyb}_{4,|\mathcal{J}|} = \mathsf{OHyb}_5$. In $\mathsf{OHyb}_{4,l}$ the information available to the adversary is

$$\left(\{\mathsf{pk}^{\mathcal{J}}_{1,i}, \mathsf{com}^{\mathcal{J}}_{0,i}, \mathsf{ck}^{\mathcal{J}}_{0,i}\}_{i=1}^{l} \cup \{\mathsf{pk}^{\mathcal{J}}_{0,i}, \mathsf{com}^{\mathcal{J}}_{0,i}, \mathsf{ck}^{\mathcal{J}}_{0,i}\}_{i=l+1}^{|\mathcal{J}|}\right) \{\mathsf{tag}_{1,i}\}_{i=1}^{|\mathcal{S}|}$$

and $\{\mathsf{com}'_{1,i}, \mathsf{ck}'_{1,i}\}_{i=1}^{|\mathcal{S}|}$. As $\mathsf{pk}^{\mathcal{J}}_{0,l}$ and $\mathsf{pk}^{\mathcal{J}}_{1,l}$ are identically distributed, $\mathsf{OHyb}_{4,l-1} \equiv \mathsf{OHyb}_{4,l}$ holds.

$\underline{\mathsf{OHyb}_5 \equiv \mathsf{OHyb}_6}$: To show the equivalence, we define $|\mathcal{J}| + 1$ sub-hybrids where each commitment and coin key is changed by the same argument as in $\mathsf{OHyb}_2 \equiv \mathsf{OHyb}_3$.

$\underline{\mathsf{OHyb}_6 \equiv \mathsf{OHyb}_7}$ holds by the simulatability of AS.

□

Theorem 6.6 (Transaction Privacy)**.** *If* Ξ *has offer privacy and* $\mathsf{SoK}[\mathcal{L}_\emptyset]$ *is simulatable, the construction* Ξ *has transaction privacy.*

Proof of Theorem 6.6 (Transactions). Similarly to the offer privacy, we prove the transaction privacy with a set of hybrids.

$\underline{\mathsf{THyb}_1}$ is the same as $\mathsf{TxPv}^0_{\mathcal{A}}$

$\underline{\mathsf{THyb}_2}$ differs such that in each Seal call, $\mathfrak{t}$ is simulated by

$$\mathsf{SoK}[\mathcal{L}_\emptyset]\mathsf{Sim}(\mathsf{stmt}(tx), tx)$$

$\underline{\mathsf{THyb}_3}$ differs in each call to Offer, the aggregatable signature is simulated by $\overline{\mathsf{AS.Sim}}_{\mathcal{L}^{\mathsf{TRS}}}$.

The information available to the adversary is now limited to

$$\mathsf{tx}_0 = \left(\{\mathsf{tag}_{0,i}\}_{i=1}^{|\mathcal{S}|}, \{\{\mathsf{acc}^{\mathcal{R}}_{i,j}\}_{j=1}^{|\mathcal{R}_i|}\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{acc}^{\mathcal{J}}_{0,i}\}_{i=1}^{|\mathcal{J}|}\right)$$

and $\{\mathsf{com}'_{0,i}\}_{i=1}^{|\mathcal{S}|}$ in $\mathfrak{t}_0$, as everything else in $\mathfrak{t}_0 = (\mathfrak{s}_0, \{\mathfrak{r}_{0,i}, \mathsf{com}'_{0,i}\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a}_0)$ is simulated. The ring accounts do not contain any information about b. Only $\left(\{\mathsf{tag}_{0,i}\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{acc}^{\mathcal{J}}_{0,i}\}_{i=1}^{|\mathcal{J}|}\right)$ and $\{\mathsf{com}'_{0,i}\}_{i=1}^{|\mathcal{S}|}$ is left, which we gradually change to $\left(\{\mathsf{tag}_{1,i}\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{acc}^{\mathcal{J}}_{1,i}\}_{i=1}^{|\mathcal{J}|}\right)$ and $\{\mathsf{com}'_{1,i}\}_{i=1}^{|\mathcal{S}|}$.

$\underline{\mathsf{THyb}_4}$ changes the intermediate commitments and simulate the proofs from

$$\left(\{\mathsf{tag}_{0,i}\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{acc}^{\mathcal{J}}_{0,i}\}_{i=1}^{|\mathcal{J}|}\right)$$

and $\{\mathsf{com}'_{1,i}\}_{i=1}^{|\mathcal{S}|}$

$\underline{\mathsf{THyb}_5}$ changes the tags and simulate the proofs from

$$\left(\{\mathsf{tag}_{1,i}\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{acc}^{\mathcal{J}}_{0,i}\}_{i=1}^{|\mathcal{J}|}\right)$$

and $\{\mathsf{com}'_{1,i}\}_{i=1}^{|\mathcal{S}|}$
$\underline{\mathsf{THyb}_6}$ changes the output accounts which consist of $\mathsf{acc}_i^{\mathcal{T}} := (\mathsf{pk}_i^{\mathcal{T}}, \mathsf{com}_i^{\mathcal{T}})$ and this hybrid changes the output accounts and simulates the proofs from

$$\left(\{\mathsf{tag}_{1,i}\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{pk}_{1,i}^{\mathcal{T}}, \mathsf{com}_{0,i}^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}\right)$$

and $\{\mathsf{com}'_{1,i}\}_{i=1}^{|\mathcal{S}|}$
$\underline{\mathsf{THyb}_7}$ changes the output commitments and simulates the proofs from

$$\left(\{\mathsf{tag}_{1,i}\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{pk}_{1,i}^{\mathcal{T}}, \mathsf{com}_{1,i}^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}\right)$$

and $\{\mathsf{com}'_{1,i}\}_{i=1}^{|\mathcal{S}|}$
$\underline{\mathsf{THyb}_8}$ differs such that the aggregatable signature is no longer simulated but the actual $\mathsf{AS.Sign}_{\mathcal{L}^{\mathsf{TRS}}}()$ is used.
$\underline{\mathsf{THyb}_9}$ differs such that t in Seal is no longer simulated but a real

$$\mathsf{SoK}[\mathcal{L}_\emptyset]\mathsf{Sign}(\mathsf{stmt}(\mathsf{tx}_1), \mathsf{wit}(\mathcal{S}_1, \{\mathsf{com}'_{1,i}, \mathsf{ck}'_{1,i}\}_{i=1}^{|\mathcal{S}|}), \mathsf{tx}_1)$$

which is equal to the privacy experiment $\mathsf{TxPv}_{\mathcal{A}}^1$
We now show the indistinguishability of the hybrids:
$\underline{\mathsf{THyb}_1 \equiv \mathsf{THyb}_2}$ follows from the simulatability of SoKs.
$\underline{\mathsf{THyb}_2 \equiv \mathsf{THyb}_3}$ follows from the simulatability of AS.
$\underline{\mathsf{THyb}_3 \approx_c \mathsf{THyb}_4}$: To show the indistinguishability, we define $|\mathcal{S}| + 1$ sub-hybrids where each changes one intermediate commitment. The first sub-hybrid is equal to $\mathsf{THyb}_{3,0} = \mathsf{THyb}_3$ and the last is $\mathsf{THyb}_{3,|\mathcal{S}|} = \mathsf{THyb}_4$. The information available to the adversary in $\mathsf{THyb}_{3,l}$ is $\left(\{\mathsf{tag}_{0,i}\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{acc}_{0,i}^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}\right)$ and $\{\mathsf{com}'_{1,i}\}_{i=1}^{l} \cup \{\mathsf{com}'_{0,i}\}_{i=l+1}^{|\mathcal{S}|}$. With the hiding property of THC, $\mathsf{THyb}_{3,l-1} \approx_c \mathsf{THyb}_{3,l}$ holds.
$\underline{\mathsf{THyb}_4 \approx_c \mathsf{THyb}_5}$: Changing the tags holds by the same reasoning as $\underline{\mathsf{OHyb}_3 \approx_c \mathsf{OHyb}_4}$.
$\underline{\mathsf{THyb}_5 \equiv \mathsf{THyb}_6}$ holds because of the randomly distributed pk in the same reason as $\underline{\mathsf{OHyb}_4 \equiv \mathsf{OHyb}_5}$.
$\underline{\mathsf{THyb}_6 \approx_c \mathsf{THyb}_7}$: To show the indistinguishability, we define $|\mathcal{T}| + 1$ sub-hybrids where each changes one intermediate commitment. The first sub-hybrid is equal to $\mathsf{THyb}_{6,0} = \mathsf{THyb}_6$ and the last is $\mathsf{THyb}_{6,|\mathcal{T}|} = \mathsf{THyb}_7$. In $\mathsf{THyb}_{6,l}$ the information available to the adversary is

$$\left(\{\mathsf{tag}_{1,i}\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{pk}_{1,i}^{\mathcal{T}}, \mathsf{com}_{1,i}^{\mathcal{T}}\}_{i=1}^{l} \cup \{\mathsf{pk}_{1,i}^{\mathcal{T}}, \mathsf{com}_{0,i}^{\mathcal{T}}\}_{i=l+1}^{|\mathcal{T}|}\right)$$

and $\{\mathsf{com}'_{1,i}\}_{i=1}^{|\mathcal{S}|}$. With the hiding property of THC, $\mathsf{THyb}_{6,l-1} \approx_c \mathsf{THyb}_{6,l}$ holds.
$\underline{\mathsf{THyb}_7 \equiv \mathsf{THyb}_8}$ holds by the simulatability of AS.
$\underline{\mathsf{THyb}_8 \equiv \mathsf{THyb}_9}$ holds by the simulatability of SoK.

□

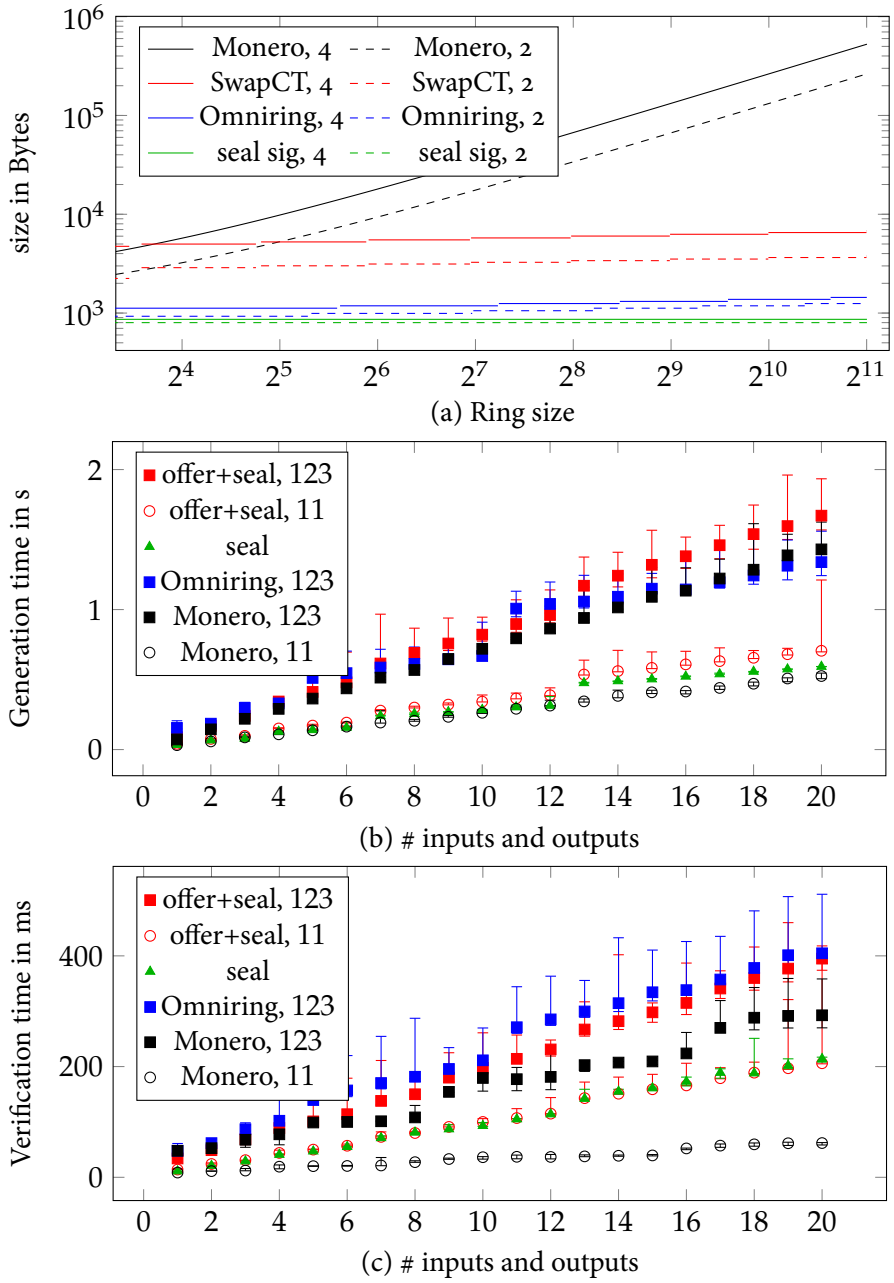


Figure 6.3: a)Transaction size for 4 and 2 inputs and outputs depending on the ring size. For Omniring, we assumed a common ring size of r. Run time for (b) transaction generation with ring size independent sealing time part and (c) transaction verification with a ring independent seal verification in SwapCT, Omniring and Monero (`RCTsimple`) with same number of inputs and outputs for two different ring sizes $r \in \{11, 123\}$ (Monero's default is 11 and there is no Omniring data for 11, as the ring size must be larger than number of inputs). The points show the median and the error bars the minimum and maximum time of 30 runs.[EMP+21]

6.7 EVALUATION

The offer and transaction sizes of our SwapCT system are competitive compared to other RingCT systems. The parameters which influence the transaction size are the number of inputs m, the number of outputs n, and the size of the anonymity set r. We denote the size of an elliptic curve point $\tilde{\mathbb{G}}$ and a field element $\tilde{\mathbb{Z}}$, both 32 Bytes in our implementation with `curve25519`. The final transactions consist of an offer and a nearly constant seal signature ($\approx$ 800 B) which is independent of r. Both signatures together with the outputs, keys and commitments total to

RINGS: $m \cdot (5\tilde{\mathbb{Z}} + (2 + 1 + 4 + 2 \cdot \lceil \log_2(r + 5) \rceil)\tilde{\mathbb{G}})$
AGGREGATION: $+m(1\tilde{\mathbb{Z}} + 2\tilde{\mathbb{G}}) + n(2\tilde{\mathbb{Z}} + 2\tilde{\mathbb{G}}) + 1\tilde{\mathbb{Z}}$
SEAL: $+5\tilde{\mathbb{Z}} + (4 + 2\lceil \log_2(2 + m + n + m \cdot n + 64n) \rceil)\tilde{\mathbb{G}}$
OUTPUTS $+3\tilde{\mathbb{G}}$

While significantly better than Monero's proofs ($O(m \cdot r + \log(n))$), we observe our size to be asymptotically linear in m and n while single [LRR+19] and multi-type (Chapter 5) Omniring transaction have no linear components: $O(\log(r \cdot m + n))$. The possibility to non-interactively merge offers requires independent proofs for each input and output, prohibiting aggregation. However, we achieve a similar logarithmic dependency on r. Absolute transaction sizes are shown in Figure (7.2a) where a common transaction (4 in/4 out) requires approximately 5 kB (offer: 4.5 kB, seal: 0.8 kB). These sizes even hold for ring sizes of 1000, out of reach for the current Monero transaction signatures. To show the applicability of our SwapCT scheme, we implemented a prototype in `rust` based on `curve25519_dalek` [Isi20]. All benchmarks are compiled with rustc 1.48 and run on a ThinkPad T460p with a i7-6820HQ CPU running kubuntu 20.10 on kernel 5.8.0-43. Timings for your hardware are easily generated by running our published code[1]. We provide a Dockerfile with all dependencies, however execution in a container might impact performance. For comparison, we chose Monero's `RCTsimple` as it is the only system with an implementation available, which includes all aspects of transaction generation, e.g. encryption of account values. For a better comparison to Omniring, we implemented a full Omniring system and provide a performance comparison in the `omniring` branch.

Compared to systems without swaps, the total time to create a SwapCT transaction consists of creating an offer and then sealing it (Figure 7.2b). Either a single signer creates a balanced offer themselves, or multiple offers are merged by a sub millisecond operation of adding randomness. Signing an offer depends on the anonymity set size. A transaction always requires a sealing operation, independent of the anonymity set size, which is shown as an offset.

The verification in SwapCT consists of the same two parts (Figure 7.2c). An ring size independent seal signature verification and the offer verification. As Monero only supports complete transactions, we compare the sum of necessary steps in SwapCT (offer+seal and verify offer+verify seal) to the Monero implementation.

1 https://github.com/SwapCT/SwapCT

While our prototype is slightly slower than Monero, it is comparable to the deployed production system with fewer features. The largest discrepancy for low ring size transaction verification is the result of meticulous optimization of the Monero verification code over multiple years, as it is run by every participant of the network. For larger anonymity sets, we perform on par, showing that our protocol works efficiently and is production-ready after a security audit of the implementation.

6.8 CONCLUSION

With our SwapCT system, we present a novel decentralized transaction system which supports both privacy-preserving transactions and non-interactive atomic swaps as answer to our second research question. We formalize the system and provide an efficient instantiation which offers logarithmically sized transactions for large anonymity sets. Our prototype implementation demonstrates equal performance to current systems that do not support multiple tokens or swap transactions. Thereby, our SwapCT system enables secure and private trading of multiple types for decentralized transaction systems and digital currencies. At a larger scale, our system allows anyone to operate a fully functional decentralized token exchange.

III

APPLICATIONS AND OUTLOOK

CONFIDENTIAL TOKEN LICENSING

7

7.1 OVERVIEW

The applications possible with a privacy-preserving multi-type system are manyfold. Like on Ethereum, a type may be purely used for random speculation or sold as entry ticket to raise funding for a service to be built. With the time lock capability, even complex ICO schemes, such as flexible air-drop mechanics and proportional distribution of initial stake according to the maximum value raise are implementable.

One might argue that our privacy-preserving multi-type systems are not suitable for more complex state transitions, which are usually handled with smart contracts. Especially the transaction structure does not allow transferring tokens from a party depending on some predicate. Only an active party can transfer their tokens. In the example below, we showcase how a licensing system for 3D printed objects is implemented under these limitations on top of our system. A naive, non-confidential implementation with a global state as e.g. an Ethereum smart contract, directly specifies public rules on which license updates are allowed. The license management approach presented in this chapter was earlier published as [ESG+21].

The license management system handles all necessary operations to deal with registration of objects to licenses, selling of licenses and production of physical parts. After manufacturing of parts, our system enables track and trace throughout the life-cycle and decommissioning of the part.

We achieve this by assigning each part in the system its own account into which parts are able to receive tokens. The account has the special property that it has no key to spend the received tokens, thereby locking them to the part forever. The confidentiality of amounts and types allows only the handlers of the part to recover which tokens the part owns. All other participants of the system get no insight at all. Different properties are encoded using special types registered to a specific company. For 3D model files, the type includes a implicit integrity check.

As the system is purely digital, it relies on the interfaces to the real world to act honestly. It incentivizes honest behavior as honest participants have a record of their actions in the case of a dispute.

7.2 USE CASE SETTING

In this section, we describe the general setting in which we apply our license management system shown in Figure 7.1. First, there is a designer creating a digital representation of a 3D object ①. This 3D file is then registered in our system by the designer, attributing this design to the designer. Once registered, the designer sells licenses either directly to manufacturers ③ or to intermediaries ②. Anyone with a license can manufacture the part and register it by

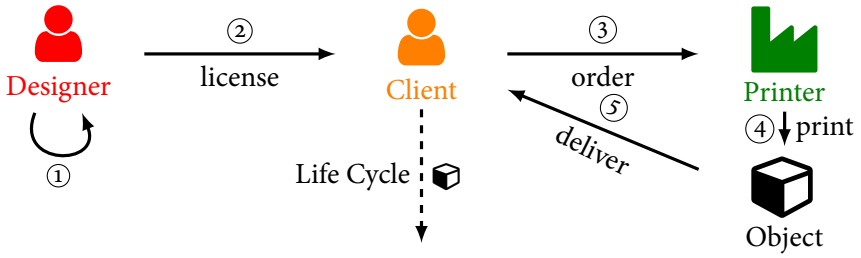


Figure 7.1: Setting of a license management system with entities and interactions.[ESG+21]

permanently binding the license to a unique identifier assigned to the part ④. Therefore parts are first-class citizens in our transaction system and have their own accounts owning the license. This allows someone handling the part to verify its legitimacy ⑤. Further attributes arising during the part's life cycle, e.g. assembly, disassembly, certification and recycling, are also permanently logged to allow for continuous tracking of the part. This process flow is especially relevant in industries where parts need to be meticulously documented like aviation. As licensing and production data provides detailed insight and often leads to a competitive advantage, the actions must only be visible to entities involved in handling the parts. In summary, we identified the following requirements for our license management system:

- Only correctly licensed objects are verifiable as legitimate.
- Once a design is registered, it cannot be re-registered by another participant.
- The general public and competitors can verify that no licenses are misused (e.g. assigning the same license to multiple printed objects).
- Only the directly involved parties get insight into plaintext information. This allows oversight of operations by competitors without revealing detailed insight. Participants must participate anonymously as otherwise meta-data information about their location and general transaction volumes get leaked.

7.3 CONFIDENTIAL TOKEN TRANSACTION SYSTEM

Our system is built on top of a privacy-preserving unspent transaction output (UTXO) protocol for typed tokens such as the ones presented in Chapters 5 and 6. While we our transaction systems provide an efficient platform, we describe our licensing construction in a way, where another multi-type privacy-preserving UTXO scheme, such as Stellar (although with reduced anonymity guarantees), can be used.
Our System requires two types of long term accounts, one from which tokens are spendable and receive only accounts. The regular long term accounts,

have a secret key lts with $(\mathsf{lts}, \mathsf{ltv}, \mathsf{ltp}) \xleftarrow{\$} \mathsf{AccGen}()$ and lts is used to authorize spending. Receive only accounts have no secret key and thereby are not able to sign outgoing transactions $(\mathsf{ltv}, \mathsf{ltp}) \xleftarrow{\$} \mathsf{ItemGen}(\mathsf{eID})$ where eID is any identifier. A recipient retrieves all received tokens, from a read only account or a standard account with $\mathsf{View}(\mathsf{ltv}, \mathsf{acc})$ where ltv is the long term view key available for regular account and receive only account.
We present the algorithms required for our application which is a superset of the privacy-preserving definition of the UTXO formalization in Definition 2.9.

Definition 7.1. *A privacy-preserving multi-type system based on one-time accounts consists of the following algorithms which are a close adaptation of SwapCT:*

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$: *takes the security parameter λ and outputs public parameters* pp*, implicitly given to the subsequent algorithms.*

$(\mathsf{lts}, \mathsf{ltv}, \mathsf{ltp}) \leftarrow \mathsf{AccGen}(\mathsf{seed})$: *optionally takes a seed, if non provided uses a random seed and outputs a long term spend key* lts*, view key* ltv *and the matching public key* ltp*.*

$(\mathsf{ltv}, \mathsf{ltp}) \leftarrow \mathsf{ItemGen}(\mathsf{eID})$: *takes a part* $\mathsf{eID} \in \{0,1\}^\lambda$ *and outputs a long term view key* lts *with matching public key* ltp*.*

$(\mathsf{acc}, \mathsf{ck}) \leftarrow \mathsf{OTGen}(\mathsf{ltp}, \mathsf{ty}, a)$: *generates a one-time account from a long term public* ltp*, a type* $\mathsf{ty} \in \mathbb{T}$ *and an amount a. It returns an account* acc *and a coin key* ck*.*

$\sigma \leftarrow \mathsf{Spend}(\mathcal{S}, \mathcal{T})$: *takes a set of inputs* $\mathcal{S} = \{a_i^{\mathcal{S}}, \mathsf{ty}_i^{\mathcal{S}}, \mathsf{ck}_i^{\mathcal{S}}, \mathsf{sk}_i, \mathsf{acc}_i^{\mathcal{S}}\}_{i=1}^{|\mathcal{S}|}$ *with amounts* $a_i^{\mathcal{S}}$ *of typ* $\mathsf{ty}_i^{\mathcal{S}}$ *with coin key* $\mathsf{ck}_i^{\mathcal{S}}$ *and secret key* sk_i *for the account* $\mathsf{acc}_i^{\mathcal{S}}$*. The outputs* $\mathcal{T} = (\mathsf{ck}_i^{\mathcal{T}}, a_i^{\mathcal{T}}, \mathsf{ty}_i^{\mathcal{T}}, \mathsf{acc}_i^{\mathcal{T}})$ *are defined by their amount* $a_i^{\mathcal{T}}$ *of type* $\mathsf{ty}_i^{\mathcal{T}}$ *hidden by coin key* $\mathsf{ck}_i^{\mathcal{T}}$ *in the account* $\mathsf{acc}_i^{\mathcal{T}}$*. The algorithm outputs a signature σ for the transaction* tx *which is defined as* $\mathsf{tx}(\mathcal{S}, \mathcal{T}) := (\{\mathsf{aux}_i^{\mathcal{S}}\}_{i=1}^{\mathcal{S}}, \{\mathsf{acc}_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|})$ *with some auxiliary information* aux_i *anonymously referencing the inputs.*

$\sigma \leftarrow \mathsf{CoinGen}(\mathcal{T})$: *takes a set of outputs defined as* $\mathcal{T} = (\mathsf{ck}_i^{\mathcal{T}}, a_i^{\mathcal{T}}, \mathsf{ty}_i^{\mathcal{T}}, \mathsf{acc}_i^{\mathcal{T}})$ *and outputs a signature σ for the token initiation transaction defined as* $\mathsf{tx}(\mathcal{T}) := (\{\}, \{\mathsf{ty}_i^{\mathcal{T}}, \mathsf{acc}_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|})$.

$\mathsf{state}' / \perp \leftarrow \mathsf{Verify}(\mathsf{state}, \mathsf{tx}, \sigma)$: *takes the state* state *of the system and a transaction* tx *with its signature σ. It either outputs a new state* state′ *with the transaction included or $\perp$.*

$(a, \mathsf{ty}, \mathsf{ck}) \leftarrow \mathsf{View}(\mathsf{ltv}, \mathsf{acc})$: *takes a long term view key* ltv *for an account* acc *and outputs the containing amount a of typ* ty *with coin key* ck*.*

$\mathsf{sk} \leftarrow \mathsf{Receive}(\mathsf{lts}, \mathsf{acc})$: *takes a long term secret key* lts *for an account* acc *and outputs the corresponding account secret key* sk*.*

$0/1 \leftarrow \mathsf{ChkKey}(\mathsf{sk}, \mathsf{acc})$: *takes an account* acc *and a secret key* sk *and outputs a bit depending on the validity.*

$0/1 \leftarrow \mathsf{ChkValue}(\mathsf{ck}, \mathsf{ty}, a, \mathsf{acc})$: *takes an account* acc *a coin key* ck *and checks if the* ty *and amount a match the account.*

The system has to fulfill the following correctness criteria:

Definition 7.2 (Correctness)**.** *A multi-type transaction system is correct if*

CORRECTLY GENERATED ONE-TIME ACCOUNTS ARE VIEWABLE: *For all* $\mathsf{ty} \in \mathbb{T}$ *and all* $a \in \{0, \dots, 2^{64}-1\}$ *and* $(\mathsf{lts}, \mathsf{ltv}, \mathsf{ltp}) \leftarrow \mathsf{AccGen}()$ *or* $(\mathsf{ltv}, \mathsf{ltp}) \leftarrow \mathsf{ItemGen}(\mathsf{eID})$ *the account* $(\mathsf{acc}, \mathsf{ck}) \leftarrow \mathsf{OTGen}(\mathsf{ltp}, \mathsf{ty}, a)$ *is viewable with* $(a', \mathsf{ty}', \mathsf{ck}') \leftarrow \mathsf{View}(\mathsf{ltv}, \mathsf{acc})$ *such that* $(a, \mathsf{ty}, \mathsf{ck}) = (a', \mathsf{ty}', \mathsf{ck}')$.

CORRECTLY GENERATED ONE-TIME ACCOUNTS FROM AccGen IS ***receivable:*** *For all* $\mathsf{ty} \in \mathbb{T}$ *and all* $a \in \{0, \dots, 2^{64}-1\}$ *and* $(\mathsf{lts}, \mathsf{ltv}, \mathsf{ltp}) \leftarrow \mathsf{AccGen}()$ *the account* $(\mathsf{acc}, \mathsf{ck}) \leftarrow \mathsf{OTGen}(\mathsf{ltp}, \mathsf{ty}, a)$ *is receivable with* $\mathsf{sk} \leftarrow \mathsf{Receive}(\mathsf{lts}, \mathsf{acc})$ *such that* $\mathsf{ChkKey}(\mathsf{sk}', \mathsf{acc}) = 1$.

HONESTLY GENERATED TRANSACTIONS SHOULD VERIFY: *For any tuple* $\mathcal{S}, \mathcal{T}$ *with the structure from* Spend*, where*

- $\forall i \in [|\mathcal{S}|] : \mathsf{ChkKey}(\mathsf{sk}_i, \mathsf{acc}_i^{\mathcal{S}}) = 1$
- $\forall i \in [|\mathcal{S}|] : \mathsf{ChkValue}(\mathsf{ck}_i^{\mathcal{S}}, \mathsf{ty}_i^{\mathcal{S}}, a_i^{\mathcal{S}}, \mathsf{acc}_i^{\mathcal{S}}) = 1$
- $\forall i \in [|\mathcal{T}|] : \mathsf{ChkValue}(\mathsf{ck}_i^{\mathcal{T}}, \mathsf{ty}_i^{\mathcal{T}}, a_i^{\mathcal{T}}, \mathsf{acc}_i^{\mathcal{T}}) = 1$
- $\forall \mathsf{ty} \in \{\mathsf{ty}_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|} : \sum_{\{i:\mathsf{ty}_i^{\mathcal{S}}=\mathsf{ty}\}} a_i^{\mathcal{S}} = \sum_{\{i:\mathsf{ty}_i^{\mathcal{T}}=\mathsf{ty}\}} a_i^{\mathcal{T}}$

holds and all input accounts $\mathsf{acc}_i^{\mathcal{S}}$ *are spendable in* state*, it holds for any proof* $\sigma \leftarrow \mathsf{Spend}(\mathcal{S}, \mathcal{T})$ *that* $\mathsf{Verify}(\mathsf{state}, \mathsf{tx}, \sigma) \neq \bot$ *with* $\mathsf{tx} = \mathsf{tx}(\mathcal{S}, \mathcal{T})$.

NEW GENERATED TYPES ARE VALID: *For all tuples* $\mathcal{T}$ *where all* $\mathsf{ty}_i^{\mathcal{T}}$ *are not previously registered in* state *and*

$$\forall i \in [|\mathcal{T}|] : \mathsf{ChkValue}(\mathsf{ck}_i^{\mathcal{T}}, \mathsf{ty}_i^{\mathcal{T}}, a_i^{\mathcal{T}}, \mathsf{acc}_i^{\mathcal{T}}) = 1,$$

it holds for any $\sigma \leftarrow \mathsf{CoinGen}(\mathcal{T})$ *that* $\mathsf{Verify}(\mathsf{state}, \mathsf{tx}, \sigma) \neq \bot$ *with* $\mathsf{tx} = \mathsf{tx}(\mathcal{T})$.

As one concrete example of such a system, we use the SwapCT instantiation of Chapter 6.

7.3.1 *Construction for missing algorithms*

For the algorithms which have no equivalent in SwapCT (Chapter 6) or Coloring (Chapter 5), we present the construction in this section. When using

SwapCT, Spend is the immediate combination of $\mathsf{Seal}(\mathsf{Offer}(\mathcal{S}, \mathcal{T}))$. To generate a receive only account, we us a key derivation function $\mathsf{KDF} : \{0,1\}^* \times D \to \mathbb{K}$ where D is the domain of labels for what the key is used and $\mathbb{K}$ is the appropriate key space. We present ItemGen in Algorithm 7.1. The tracking and view long term secret keys tsk, vsk are derived from the eID and their corresponding public keys are generated with the public key function PK. Importantly, the spending long term public key pk is directly generated by the KDF such that the private key is infeasible to calculate.

Algorithm 7.1 ItemGen

Require: eID
  tsk ← KDF(eID, `track`)
  vsk ← KDF(eID, `view`)
  vpk ← PK(vsk)
  tpk ← PK(tsk)
  pk ← KDF(eID, own)
  ltp = (vpk, tpk, pk)
  ltv = (vsk, tsk) **return** ltv, ltp

7.3.2 *Our requirements on a private transaction System*

If the above mentioned operations are supported in a privacy-preserving manner, the protocol is suitable for our application. We require the following privacy guarantees from the token protocol summarized in Definition 5.4. For our system to be secure:

SENDER-ANONYMITY: The owners of transaction inputs must only be known to the transaction creator.

RECIPIENT-ANONYMITY: The owners of outputs of a transaction must only be known to the sender, i.e. the creator of the transaction.

CONFIDENTIALITY: The amounts and types of tokens transferred in a transaction must only be known to the sender. Each recipient should only have access to the amount and type of their respective output.

7.3.3 *Generic Operations*

As most operations of our proposed system require a transfer of tokens or issuing a new token type, we present high level operations.
Given a state, an amount a for a new type with domain $d \in \{0,1\}^*$ and preimage $p \in \{0,1\}^*$ to the owner of the long term key ltp we issue a token type with Algorithm 7.2. We use a random oracle $\mathsf{H} : D \times \{0,1\}^* \to \mathbb{T}$ from the set of domains D and arbitrary input to a type.

Algorithm 7.2 Issue Token

Require: $\mathsf{state}, a, d, p, \mathsf{ltp}$
  $\mathsf{ty} \leftarrow \mathsf{H}(d \| p)$
  $(\mathsf{acc}, \mathsf{ck}) \leftarrow \mathsf{OTGen}(\mathsf{ty}, a, \mathsf{ltp})$
  $\sigma \leftarrow \mathsf{CoinGen}(\mathcal{T} = \{(\mathsf{ck}, a, \mathsf{ty}, \mathsf{acc})\})$
    return $\mathsf{Verify}(\mathsf{state}, \mathsf{tx}(\mathcal{T}), \sigma)$

Given a state, an amount a of tokens of typ ty from a one-time account acc with keys $(\mathsf{lts}, \mathsf{ltv}, \mathsf{ltp})$, Algorithm 7.3 transfers the tokens to the owner of the long term key $\mathsf{ltp}^{\mathcal{T}}$.

Algorithm 7.3 Transfer Token

Require: $\mathsf{state}, a, \mathsf{ty}, \mathsf{acc}^{\mathcal{S}}, (\mathsf{lts}^{\mathcal{S}}, \mathsf{ltv}^{\mathcal{S}}, \mathsf{ltp}^{\mathcal{S}}), \mathsf{ltp}^{\mathcal{T}}$
  $\mathsf{sk} \leftarrow \mathsf{Receive}(\mathsf{lts}^{\mathcal{S}}, \mathsf{acc}^{\mathcal{S}})$
  $(\mathsf{ty}', s, \mathsf{ck}) \leftarrow \mathsf{View}(\mathsf{ltv}^{\mathcal{S}}, \mathsf{acc}^{\mathcal{S}})$
  if $\mathsf{ty}' \neq \mathsf{ty} \vee s < a$ **then return** $\perp$
  $(\mathsf{acc}^{\mathcal{T}}, \mathsf{ck}^{\mathcal{T}}) \leftarrow \mathsf{OTGen}(\mathsf{ty}, a, \mathsf{ltp}^{\mathcal{T}})$
  $\mathcal{S} = \{(s, \mathsf{ty}, \mathsf{ck}, \mathsf{sk}, \mathsf{acc}^{\mathcal{S}})\}$
  if $s - a = 0$ **then**
    $\sigma \leftarrow \mathsf{Spend}(\mathcal{S}, \mathcal{T} = \{(\mathsf{ck}^{\mathcal{T}}, a, \mathsf{ty}, \mathsf{acc}^{\mathcal{T}})\})$
  else
    $(\mathsf{acc}_R, \mathsf{ck}_R) \leftarrow \mathsf{OTGen}(\mathsf{ty}, s - a, \mathsf{ltp}^{\mathcal{S}})$
    $\mathcal{T} = \{(\mathsf{ck}^{\mathcal{T}}, a, \mathsf{ty}, \mathsf{acc}^{\mathcal{T}}), (\mathsf{ck}_R, s - a, \mathsf{ty}, \mathsf{acc}_R)\}$
    $\sigma \leftarrow \mathsf{Spend}(\mathcal{S}, \mathcal{T})$
  return $\mathsf{Verify}(\mathsf{state}, \mathsf{tx}(\mathcal{S}, \mathcal{T}), \sigma)$

7.4 OUR TOKEN-BASED LICENSING SYSTEM

Given the scenario above and a multi-type confidential transaction protocol as explained before, our system requires to track licenses and the state of objects in a distributed system. The persisted information must be immutable and should be visible only to those parties concerned. To achieve the persistence and immutability without a trusted third party, we use a distributed ledger.
A functional approach for a blockchain-based license transaction system is to store the state of the licenses and tracked objects in a smart contract and programmatically update their state according to some smart contract rules. Here, we show that a license management system does not require arbitrary state changes, as is possible with e.g. Ethereum, but can work with the much simpler UTXO model. Our approach achieves this using a transaction logic for multiple token types. The benefit of this reduced requirement is an enhanced level of privacy. We propose an application on a blockchain based system to confidentially store the license and life-cycle information of parts which relies on distributed trust and zero knowledge proofs achieving a publicly verifiable

history of operations. Therefore, we represent licenses and attributes of objects as tokens. Each 3D model is linked to its own type. Every token of this type is a license. Abstract attributes are types too and the tokens of this type certify that their owner holds the attribute. All tokens are then transferred confidentially. To assign a license or attribute to an object, the according tokens are transferred to the account of the object. Access to the eID of a part allows reconstruction of the part's account for the verification of its attributes. In the remainder of the work, we use a multi-type system defined in Chapter 6.
The following sections describe the actions of the participants in more detail. First a new design is created and issued. Then license tokens of this design are traded. Once the part is manufactured, it gets a unique identifier and an account in the system. By transferring a license to the part it is permanently registered to the system. Any further actions in the life-cycle of the part, such as post processing or quality assurance, are then logged by attributes bound to the part's account. The actions are verifiable by anyone with access to the unique identifier.

7.4.1 *Design Issuance*

The process starts with a designer creating a 3D object as a CAD file. The designers and engineers or their companies, respectively, own the intellectual property of the object's design. As our goal is to manage licenses of such designs, we have to identify the first and original owner and prevent others from claiming the property rights after the first registration. We identify a design by a cryptographic hash value of its CAD file.
Small adaptations of the CAD file lead to different object identities, but the exact same file always results in the same identity. Automatically assessing the similarity of two designs is difficult and out of scope of our work. Various authors [HSKK01; CHL+17; MR11; LQQ+16] proposed different approaches of similarity measures, all ineffective in a malicious setting due to heuristic approaches. In our setting the designers are malicious entities trying to claim ownership of an existing design of another designer.
The designer derives a new token type $F_{\square} = \mathsf{H}(\texttt{'design'}\|\texttt{file.cad})$ and registers it in our system with an initial amount as a new token on the ledger. The designer now owns the initial supply of tokens of type $F_{\square}$. This initial supply should be chosen large enough because no new tokens of the same type can be generated again. For general parts, we suggest to have an initial supply of e.g. $2^{64} - 1$ which is plentiful. On the other hand, artificial scarcity is achieved by, e.g., issuing only 100 tokens. The designer is now in possession of the newly issued tokens. A transfer requires the matching private key of the designer. For all other participants, the registration process reveals only that someone registered a token type. Everyone can verify that this type was not registered before but gains no additional information.
Algorithm 7.4 shows the steps to register a design and requires an amount a, by default $a = 2^{64} - 1$, the file and the long term public key of the designer

ltp_D generated by AccGen. The domain `'design'` is used to separate tokens representing a CAD design from other token types in the system.

Algorithm 7.4 Issue Design

Require: state, a, `file.cad`, ltp_D
return IssueToken(state, a, `'design'`, `file.cad`, ltp_D)

7.4.2 *Trading*

For any registered design, the designer initially owns all license tokens. Each token represents a license that allows the owner to manufacture one physical object. Most of the designers are not manufacturing the objects themselves but rather sell licenses to manufacturing companies, e.g. 3D print services. License tokens are therefore traded for other tokens in the system, most likely (but not limited to) representing fiat currencies.
To exchange tokens of different types when using SwapCT, it provides a swap mechanism which ensures atomic execution. Either both parties receive each other's tokens, or the swap is not performed at all. The exchanged tokens can either be used to manufacture the object or be traded again by performing another swap transaction. Each owner of a license token can use it to manufacture and register a legitimate object of this type. Using Coloring, an external exchange is needed.
The representation of a license as a token deviates slightly from a license in most legal systems. Tokens of a specific type are indifferentiable from each other and not batched together in purchases. Another difference is that there might be many intermediary owners of the license token, before it reaches the manufacturer which is uncommon in legal licensing contracts. Given such intermediary owners and due to the sender anonymity, the reconstruction of a transaction path back to the designer requires all previous owners to cooperate. Nevertheless, every participant of the system can verify that the total amount of tokens per type stays constant.
In a multi-type system using vector Pedersen commitments, it is possible to implement an additional attribute which specifies the age of a token. The conservation NIZK then assures that all output tokens are one epoch older than the inputs. Thereby the token owner can verify the number of intermediaries.

7.4.3 *Manufacturing*

A manufacturer owning a license for an object can download the corresponding file from the designer or any shared storage. Our system implicitly protects the integrity of the file by linking it to the token type of the license. To verify the integrity, the file is hashed to a type $F' = \mathsf{H}(\texttt{'design'} \| \texttt{file.cad})$ and then compared to the token type $F' \stackrel{?}{=} F_{\square}$. If the two hash values match, the fetched file is correct and can be printed.

In most cases, the CAD file is fed into a CNC mill or a 3D printer to create the object. After a successful object of the design exists, the manufacturer is trusted to, e.g., either measure (microscopic surface structure patterns, physically unclonable functions) or assign a unique identifier eID to this part. Ideally, this eID is integrated monolithically in the part in order to prevent an undesirable non-destructive extraction or alternation of the eID. A typical representation of an electronic identification feature is a radio-frequency identification (RFID) tag. RFID tags are widely used in a variety of branches like logistics, commerce, and identification documents.
From a security perspective, the eID must not be guessable for an object. According to the NIST SP800 [Dwo07] publication, this is achieved by a random bit-string of at least 96 bit. More importantly the eID needs to be a unique feature of the part and difficult to copy to another part. For high stake parts, where identification is important, an integrated smart card might be used [RE04]. The smart card has the benefit of generating a unique random private key and supports authentication challenges to prove the authenticity. This is much harder to copy onto another part as an RFID identifier, but is orders of magnitude more expensive.
From the eID, the manufacturer generates a new read only account $(\mathsf{ltv}, \mathsf{ltp}) = \mathsf{ItemGen}(\mathsf{eID})$. This enables anyone knowing the eID to verify which tokens were sent to the part by $\mathsf{View}(\mathsf{ltv})$. To identify this part as legitimately manufactured, the license token has to be transferred to the part's account. The special feature of such an account, generating an immutable log of properties (received tokens), is the absence of a spend key. Algorithm 7.5 denotes the steps to register a valid part with an eID, the CAD file and the manufacturer's funding account acc_M along with the long term secret keys $(\mathsf{lts}_M, \mathsf{ltv}_M, \mathsf{ltp}_M)$.

Algorithm 7.5 Register Item

Require: $\mathsf{state}, \mathsf{eID}, \texttt{file.cad}, \mathsf{acc}_M, (\mathsf{lts}_M, \mathsf{ltv}_M, \mathsf{ltp}_M)$
  $\mathsf{ty} \leftarrow \mathsf{H}(\texttt{'design'} \| \texttt{file.cad})$
  $(\mathsf{ltv}_I, \mathsf{ltp}_I) = \mathsf{ItemGen}(\mathsf{eID})$
  return $\mathsf{TransferToken}(\mathsf{state}, 1, \mathsf{ty}, \mathsf{acc}_M, (\mathsf{lts}_M, \mathsf{ltv}_M, \mathsf{ltp}_M), \mathsf{ltp}_I)$

7.4.4 *Post Processing & Quality Assurance*

After manufacturing, parts regularly undergo post-processing and inspections. To attest such processes in the history of the part, the testing entity transfers a token to the part.
These tokens are not license tokens, but behave the same and can therefore use the same underlying confidential transactions. A property of parts is also represented in a new token type. Instead of the hash value of a CAD file, the new type is derived from hashing a unique string, e.g. `"Quality Assured by A. Inc."` in the domain `'property'`. The creator should publish this unique string to allow others to verify if a token really originated from them. The new type F_Q is then registered on the ledger by Algorithm 7.6 taking the

unique string Q, the authorities ltp_Q and a large initial amount a. Every time the company decides to tag a part with this property, they transfer one token of the type to the part's account.

Algorithm 7.6 Issue Certificate Tokens

Require: $\mathsf{state}, a, Q, \mathsf{ltp}_Q$
  return $\mathsf{IssueToken}(\mathsf{state}, a, \texttt{'property'}, Q, \mathsf{ltp}_Q)$

The part then has the license token and the F_Q token of the manufacturer. The absence of a private key for the part locks these tokens forever to the part. Thus, the tokens cannot be secretly transferred to a different part, given the unique identifier cannot be copied or cloned.
This process is repeated, if required, to log multiple properties to the part. On publication of a transfer, the transaction is timestamped persisting the date and time when the part received the property. The transaction to a part's account is not limited to simple binary properties. Small to medium scalar values can be stored by transferring multiple tokens of the same type to the object. Algorithm 7.7 requires the part eID, the type of certificate Q and the spending key lts_Q of the owner of acc_Q. The amount of tokens a transferred represents the value. An example could be the quality of the part on a scale from one to five, depending on the number of Q tokens received. Instead of scalar values, the number of tokens can be interpreted as a map to strings.

Algorithm 7.7 Attest Post-Processing

Require: $\mathsf{state}, \mathsf{eID}, Q, a = 1, \mathsf{acc}_Q, (\mathsf{lts}_Q, \mathsf{ltv}_Q, \mathsf{ltp}_Q)$
  $\mathsf{ty} \leftarrow \mathsf{H}(\texttt{'property'} \| Q)$
  $(\mathsf{ltv}_I, \mathsf{ltp}_I) = \mathsf{ItemGen}(\mathsf{eID})$
  return $\mathsf{TransferToken}(\mathsf{state}, a, \mathsf{ty}, \mathsf{acc}_Q, (\mathsf{lts}_Q, \mathsf{ltv}_Q, \mathsf{ltp}_Q), \mathsf{ltp}_I)$

7.4.5 *Verification*

Anyone with physical access to the part can retrieve the eID from the object and derive the account key with the same steps as the manufacturer previously did. With this key, all tokens transferred to the part can be recovered from the public ledger. The license token is checked to verify the original designer, Q tokens attest a correct print and quality check by a known manufacturer and any further properties. This is shown in Algorithm 7.8 which requires the parts eID and recovers all received tokens. The transactions of the tokens are all timestamped, which allows the reconstruction of the part's history.
All digital systems are only as reliable as the information inserted into it. The interface between the physical world and the immutable ledger has to be trusted. At least our system incentivizes actors to act correctly, as an immutable documentation proves their operations in case of dispute.

Algorithm 7.8 Item Verification

Require: state, eID
  $t = \emptyset$
  $(\mathsf{ltv}_I, \mathsf{ltp}_I) = \mathsf{ItemGen}(\mathsf{eID})$
  for all acc $\in$ state **do**
    $d \leftarrow \mathsf{Receive}(\mathsf{ltv}_I, \mathsf{acc})$
    if $d \neq \perp$ **then**
      parse d as $(\mathsf{ty}, a, \mathsf{ck})$
      $t = t \cup (\mathsf{ty}, a, \mathsf{ck})$
  return t

For all other participants of the system without the part or the part number (eID), it is impossible to identify which tokens the part possesses. Thereby, the information is available to parties with legitimate interests only.
Our system can be a basis for a complete tracking and tracing system of parts which is often desired by industries, especially in the aviation industry, where a life-long tracking of parts is not only done in production but also during the full life-cycle of a part. This can easily be used to establish a real digital twin not only on system level, but on part level. While there are already several tracking and tracing systems on part level in aviation industry, none of them presents a complete end-to-end solution as our proposed system would.

7.4.6 *Proxy Accounts for Sensitive Parts*

For sensitive parts, it might be interesting to limit the scope of which parties have insight into the tokens owned by the part. While the ownership of tokens is directly derived from the eID and every token received by the part is not spendable, we provide an extension where tracking and viewing of tokens is limited to a part owner. In this case, the view secret key ltv is not derived from the eID directly but randomized by the owner by concatenating randomness s. For verification, the owner creates a designated verifier proof that the item received a specific amount of a given type without disclosing possible other tokens of the part. For a designated verifier party with key ltp_V, the part owner creates a zero knowledge proof of knowledge (PoK):

$$\begin{aligned}&\mathsf{PoK}(\mathsf{stmt} = (\mathsf{acc}, \mathsf{ltp}_V, \mathsf{ty}, a, \mathsf{eID})\exists \mathsf{wit} = (\mathsf{lts}_V, \mathsf{ck}, s)\\ &\quad \text{s.t. } \mathsf{ItemGen}(\mathsf{eID}\|s) = (\mathsf{ltv}_I, \mathsf{ltp}_I) \wedge \mathsf{View}(\mathsf{ltv}_I, \mathsf{acc}) = (\mathsf{ck}, \mathsf{ty}, a)\\ &\qquad \vee \mathsf{ltp}_V = \mathsf{PK}(\mathsf{lts}_V))\end{aligned}$$

Such a proof transcript still reveals the one-time account acc belonging to the part. To improve privacy at the cost of performance, the proxy party creates an anonymous reference with an anonymity set of n accounts $\{\mathsf{acc}_i\}_{i=1}^n$:

$$\begin{aligned}&\mathsf{PoK}(\mathsf{stmt} = (\{\mathsf{acc}_i\}_{i=1}^n, \mathsf{ltp}_V, \mathsf{ty}, a)\exists \mathsf{wit} = (\mathsf{lts}_V, j \in \{1, \dots, n\}, \mathsf{ck})\\ &\quad \text{s.t. } \mathsf{ItemGen}(\mathsf{eID}\|s) = (\mathsf{ltv}_I, \mathsf{ltp}_I) \wedge \mathsf{View}(\mathsf{ltv}_I, \mathsf{acc}_j) = (\mathsf{ck}, \mathsf{ty}, a)\\ &\qquad \vee \mathsf{ltp}_V = \mathsf{PK}(\mathsf{lts}_V))\end{aligned}$$

This does not reveal to the verifier which transaction output contains the license or property.

7.4.7 *Life cycle*

The digital representation of properties in an immutable ledger is especially relevant during the manufacturing process, but also provides multiple benefits during the life-cycle of the part. Parts being resold or refurbished are tracked by a token of a new type. As an example, B. Inc.'s business model is to buy old parts, refurbish them and then sell them again. This is realised with a token of type `"refurbished by B. Inc."`, which is transferred to the refurbished part. Everyone trusting that this token is really from B. Inc. can then check if a part is genuine and can view the history of production. This is again performed by Algorithm 7.6 and 7.7.
The identifier of the type belonging to known companies might be signed in a traditional PKI based system of which the root is controlled by a regulatory authority of the industry sector. E.g., aviation authorities such as the European Union Aviation Safety Agency (EASA) can sign token types from Original Equipment Manufacturer (OEM) or Tier-1 Suppliers in order to verify the originality of parts or services such as a refurbishment.
Once a property token is transferred to a part, it is impossible to take it away due to the lack or a private key. If removing tokens is desired, a simple convention can resolve this, by issuing a secondary token type, which negates the original one. Any part with only the positive token is valid, but if the part also owns the negated one, it is invalid. Once negated, no one can remove the negated token from the part. One example for such a negated token is the revocation of a part. Assume a company's inventory of parts get stolen. In this scenario, the company can issue a revocation token type `"revoked by A. Inc."` and transfer tokens to the stolen parts. Everyone trusting A. Inc. can check whether the part was revoked. Such revocation tokens have to be carefully verified if they are genuine or not, as they enable a denial of service attack, if such a token is transferred to a still working part.

7.4.8 *Transient Properties*

For many tokens owned by parts, it is crucial that they cannot be transferred away from the part to a different account belonging to a different part. This would facilitate the theft of license tokens and certification of subpar parts. However, some life-cycle properties are transient and being passed around between the part and people or machines interacting with it. This can e.g. be used for a per item payment system. Each part processed earns one token which is then used to claim a reward.
For this requirement, participants must be able to transfer tokens away from the object. This is achieved by a secondary account of each item with a spend key $P = \mathsf{AccGen}(\mathsf{eID})$. A set of a tokens of type T is applied to a part eID

from some operator with private key lts_O from account acc_O with Algorithm 7.9.

Algorithm 7.9 Apply Transient

Require: $\mathsf{state}, \mathsf{eID}, T, a, \mathsf{acc}_O, (\mathsf{lts}_O, \mathsf{ltv}_O, \mathsf{ltp}_O)$
$\mathsf{ty} \leftarrow \mathsf{H}(\texttt{'attribute'} \| T)$
$(\mathsf{ltv}_I, \mathsf{lts}_I, \mathsf{ltp}_I) = \mathsf{AccGen}(\mathsf{eID})$
return $\mathsf{TransferToken}(\mathsf{state}, a, \mathsf{ty}, \mathsf{acc}_O, (\mathsf{lts}_O, \mathsf{ltv}_O, \mathsf{ltp}_O), \mathsf{ltp}_I)$

All tokens held in this account can be claimed from the object by anyone having access to the eID, the type T, an amount a (mostly $a = 1$) and a recipient operator ltp_O with Algorithm 7.10. Even after a token was transferred away from the part, it is still possible to verify that the object once owned the property token. The receiving and removal transactions even leave timestamps for both operations.

Algorithm 7.10 Recover Transient

Require: $\mathsf{state}, \mathsf{eID}, a, \mathsf{acc}_I, \mathsf{ltp}_O$
$(\mathsf{ltv}_I, \mathsf{lts}_I, \mathsf{ltp}_I) = \mathsf{AccGen}(\mathsf{eID})$
$\mathsf{sk} \leftarrow \mathsf{Receive}(\mathsf{lts}_I, \mathsf{acc}_I)$
$(\mathsf{ty}, s, \mathsf{ck}) \leftarrow \mathsf{View}(\mathsf{ltv}_I, \mathsf{acc}_I)$
return $\mathsf{TransferToken}(\mathsf{state}, a, \mathsf{ty}, \mathsf{acc}_I, (\mathsf{lts}_I, \mathsf{ltv}_I, \mathsf{ltp}_I), \mathsf{ltp}_O)$

7.5 SECURITY ANALYSIS

The confidentiality of the transaction data is important for the participants. Transfers of licenses quite always reflect some business interactions, which gives competitors valuable insight into metrics private to a company. The public nature of a blockchain requires all transactions to be public, so that a consensus can be reached. Our approach uses a blockchain which reaches a consensus and allows to verify the conservation rules publicly without revealing the data. This is achieved by using privacy-preserving transactions [EMP+21]. Each transaction includes non-interactive zero knowledge (NIZK) proofs to convince other participants of the system that the transactor created the transaction according to the conservation rules. These NIZK proofs are publicly verifiable and are used to reach consensus about the validity of a transaction.

7.5.1 *Attacker Model*

We assume all actors of the system are malicious with the exception that they semi-honestly provide inputs from the physical world. Meaning they try to attack the transaction system but do not insert bogus measurements. The requirement for semi-honest interfaces cannot be solved by any purely digital system. However the persistent logging incentivizes all participants to report correct data as they can be held accountable retrospectively. If for example a

manufacturer creates two parts with an identical eID, a single license makes both parts appear as legitimate. To prevent this, the system has to trust the manufacturers that no two objects are created with the same eID. Thanks to the persistence, once two identical parts are found, the creator of the duplicate cannot repudiate their actions. A more pro-active security measure, which decreases the trust in the manufacturer, includes trusted hardware integrated into printers. This enables securely booted platforms which ensure that every printed part is assigned exactly one license and the eID are created with good randomness. In such a setting, we can model an adversary as malicious.
The verification and persistence of transactions is performed by the entirety of actors following a consensus protocol. We assume that the majority of actors are honest and we therefore model the verification party as honest-but-curious. Instead of a Proof of Work consensus, used e.g. in Bitcoin where an unknown number of entities participate in the consensus protocol and their voting power is proportional to some brute force computation, we suggest running our transaction system on a ledger which is controlled by a classical consensus protocol such a paxos [Lam19]. Including a large number of suppliers, manufacturers, clients and designers in the consensus process, most of them need to collude to change a sequence of historical events. Having a public set of entities, each with equal voting power in the consensus process, prevents sibyl attacks [Dou02] where one entity pretends to be multiple voters to unfairly gain more votes.

7.5.2 *Security Reduction*

All operations of our system either create transactions or receive transaction outputs in a way that the security model of the underlying transaction system is taking care of. So regarding anonymity and confidentiality, our system provides the same guarantees as the used privacy-preserving transaction system. An efficient attacker to the security of our system can be directly used to break the security properties of the underlying UTXO transaction system.
The permanent binding of a license to accounts of parts holds because the accounts do not have a secret key by design. Without the secret key, it is computationally infeasible to create a valid spend transaction for the underlying transaction system to transfer the tokens away. To generally prove ownership of a license, any holder of a license token is able to prove that they have not yet spent it by revealing the tag or creating a zero knowledge proof of solvency. Protocols therefore are presented for multiple privacy-preserving protocols [DV19; DJV19].
To securely deploy our token licensing system, several components require special attention. In the following sections, we elaborate on specific areas of confidentiality, which are important to license transactions and life-cycle tracking.

7.5.3 *Revealing Transactions*

In some circumstances, transactors have to reveal past transactions to third parties or authorities. If the underlying ledger uses anonymity sets, where each transactor is hidden in a set of unrelated transactors and too many transactions in a system are publicly known, it gets more and more difficult to stay anonymous within the network of transactions. So revealing some selected transactions to business partners or regulatory entities does not interfere with the anonymity in the system. It is generally in the participants' interest to keep their transactions secret from the general public and especially from competitors.
However, in some industries access to information needs to be granted to certain institutions or authorities. Example use-cases are regular audits to ensure compliance or an investigation following an accident. To achieve this, an escrow key of the involved companies is stored at a safe place. The escrow key cannot be abused to gain insights into transactions of other participants.
If it is sufficient for the auditing entity, the part owners may present zero knowledge proofs of compliance instead of revealing the secret keys. This is achieved equivalently to the proxy accounts from Section 7.4.6.

7.5.4 *Item* eID *Enumeration*

Another important issue to care about, are eIDs of parts. As they are the seed of the public account keys, they must not be guessable by someone not owning or having access to the part. Otherwise the part's account can be revealed. Also enumeration attacks where parts have consecutive numbers can lead to attacks on the confidentiality of tokens sent to these parts. A reliably non guessable source as of NIST-SP800 [Dwo07] is to use 96 bits of uniformly random data. From this, a part account is derived which is not guessable. This keeps the tokens transferred to the part secret to the owners of the part only. As previously mentioned, another option is to integrate a smart card with a randomly generated private key.
It is still important to know that everyone who had access to the part in its history is able to store the eID and track future transactions of the part.

7.6 PERFORMANCE

To evaluate the performance of the system we take a closer look at the operations required and estimate a number of transactions per second. Operations are defined as the transfer of tokens from one account to another. Going back to the example of the aircraft industry an airplane is made up of around 350,000 [NBC10] to 6 million [Alt16] parts from hundreds of different suppliers and subsuppliers. All those parts could be integrated into our license management system. Airbus and Boeing manufacture around 800 airplanes per year [Air19]. This equals to around 8,87 operations per second for a company like Airbus. Taking into account other industries like car manufacturing

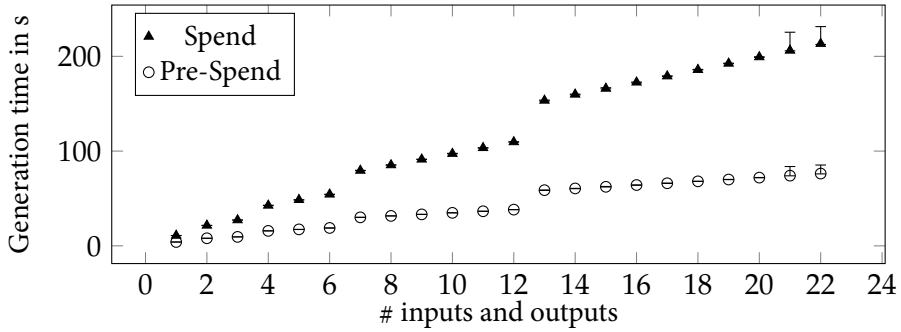


Figure 7.2: Transaction generation time of our prototype implementation on a Raspberry Pi first generation, representing an embedded system within a printer. As the secret keys are only required in part of the transaction, it is sufficient to execute the Pre-Spend on the embedded device and the remaining part can be offloaded to a powerful computer with minimal privacy loss. We use an anonymity set size of 27. The error bars are the minimum and maximum runtime of 10 executions. [ESG+21]

the numbers of operations are significantly higher. A car is made up of around 7,000 [Alt16] to 15,000 [KR08] parts and 97 Million cars and commercial vehicles were produced in 2017 [Man18]. If every part is produced based on a token transfer, this results in around 46,000 operations per second. Extending to the multitude of manufacturing sectors where a confidential license management system could be integrated the potential number of operations could be manifold.

To show the applicability of our scheme, we evaluated the performance relevant for different scenarios. The transaction generation for e.g. registering items is most likely performed by the 3D printer. We use a Raspberry Pi first generation to represent the resource constraint device. Figure 7.2 shows that usual transactions with few inputs and outputs require less than one minute, negligible in relation to the printing duration.

Transaction verification is most likely performed in an on-premise data center with powerful machines. As test payload we used transactions with 2 inputs and 2 outputs, as they will be most frequent in the system. On an dual socket AMD EPYC 7281 Server (64 threads) with Ubuntu 20.04, we achieved a median of 591 transaction verifications per second with a worst case performance of 582.

Another relevant time is scanning all transaction outputs for ones belonging to a part or an account. Only with the elD it is possible to detect an output. We measured a median of 260,000 output verifications per second on the above mentioned server.

From these measurements and an estimation of 50,000 operations per second and 5 kB storage per transaction, we derive the computational requirements of our system. Depending on the time by which a transaction output is reused in a subsequent transaction, the verification of transactions can be massively

parallelized. For the verification to keep up with the transaction generation, each participant requires around 100 servers to verify all transactions in real time and one server to track new outputs. This is comparable to other systems with similar throughput. To store all transactions, around 20 TB per day are required. An efficient consensus mechanism easily achieves this throughput resulting in a global total order of transactions.
Within a trusted environment, e.g. in a single company, only one node is required to verify and store all transactions. Smaller nodes trusting the central entity can validate part properties based on the central, verified storage. For a system tracking each part of every car manufactured, these requirements are possible with current commodity hardware and allow an immutable, publicly verified log of all operations

7.6.1 *Concurrency*

Every state change of a part is reflected with a transaction. However, the order of the persisted transactions cannot be determined in advance. Transacting two tokens to the account of a part at approximately the same instant will result in an arbitrary order. There are no guarantees like first in first out, even for a single part, as each transaction is hiding the recipient and there might be a separate actor transmitting a token to said part. To implement strict conditional dependencies, it is necessary to query the persisted log, if the condition is persisted.
The certification accounts have to split their initial output, after issuing, into multiple smaller outputs. Using the change output from the previous part is only possible after the transaction is persisted, so repetitively using the change to attest many parts in short succession is not possible. This issue is mitigated by using smaller outputs in a round robin fashion.

7.7 CONCLUSION

The tracking and tracing of parts in complex industries such as automotive or aviation is a fundamental task, especially having intricate business relations. In addition, reliably differentiating legitimately licensed parts from reputable sources is a difficult feat. On top, the information about parts must remain secret from competitors and alike.
Our proposed system is able to provide all required functionality while maintaining confidentiality of operations in an untrusted environment. We defined protocols to represent the complex life-cycle actions in the form of basic transfers of tokens. By building the system on top of a privacy-preserving multi-type blockchain, our system inherits the confidentiality guarantees. Given an efficient consensus mechanism for transaction ordering, the verification is massively parallelizable and can handle up to 10 transactions per core per second. These features combined with a decent performance makes our system directly applicable to many real world life-cycle management scenarios.

This application especially demonstrates the usefulness of our previous contributions in building privacy-preserving multi-type transaction systems.

CONCLUSION & FUTURE WORK

8.1 RESEARCH ANSWERS

8

In this thesis, we answered the two main research questions. The first one was: *How to integrate confidential types into private transaction systems?* To integrate confidential types into a transaction system, we formalized a Coloring system in Chapter 5. Then we proposed an conservation NIZK which is designed for succinct typed commitments in Chapter 4. This cuts the linear component of transaction sizes in half compared to previously existing schemes. We embedded the conservation NIZK into a transaction system to achieve efficient, confidential multi-type transactions Chapter 5. For the full functionality, we integrated an authorization NIZK for a m-out-of-n anonymity set. To showcase the flexibility and space efficiency of our scheme, we added an additional attribute to transaction outputs, namely a time lock. Time locked outputs allow building second layer solutions, such as payment channels, on top of a transaction system.
For the second research question on *how to enable fair, anonymous trading in a multi-type system*, we proposed a new concept of partial transactions, called offers in Chapter 6. Mergeable offers enable atomic and fair exchange of typed tokens. Compared to previous exchange protocols, our solution results in a single transaction persisted on the ledger and requires only a single message from the offer creators. The core of the mergeable offers is an anonymously aggregatable signature scheme which we formalize and efficiently instantiate in Chapter 3. Thereby, offer creators authorize contributing their tokens if and only if the final transaction is a superset of their intended outputs.
To substantiate our claim that Coloring and SwapCT as privacy-preserving multi-type transaction systems provide an expressive platform, we presented a fully featured confidential license management system in Chapter 7. In addition to distribution and verification of licenses the system is capable confidentially persisting manufacturing supply chains of sizable industries. Such an application with high confidentiality requirements is impossible without support for confidential types.

8.2 LIMITATIONS

While this work presents novel techniques for flexible privacy-preserving transaction systems, there are still multiple obstacles to overcome towards more privacy-preserving functionality.
In swap transactions, the final transaction submitted to the ledger will benefit from very similar privacy guarantees as current privacy-preserving transaction systems. This assumes that the statistical sampling method of anonymity sets are the same across offers and only depend on the real input. Otherwise transactions may be decomposed according to their anonymity set sampling.

An example is an old offer, which obviously sampled the anonymity set from outputs existing at the time of creation, merged with a new offer allows separating inputs by their anonymity set. Input anonymity sets which contain new outputs can only belong to the new offer. Limiting offers to epochs might remedy this issue, but this introduces new parameters, such as the length of these epochs, which are highly dependent on the specific environment where the system is deployed.

However, for unfinished offers, we provide no amount or type confidentiality and suggest users to disseminate them carefully. To be matched, offers need to be published to a group of peers with enough liquidity. We have no experimental data on how much advantage observed offers provide to an adversary against the anonymity of persisted transactions.

Summarizing the issues above, the privacy achieved by our transaction system depends on the carefulness of the participants. From the adoption of anonymity sets in Monero [MSH+18], we expect that users only choose privacy friendly methods if they are forced to and otherwise optimize for lower transaction fees. Applied to offers, a privacy unfriendly scenario could arise, where all offers are publicly broadcasted, allowing an always listening adversary to fully reconstruct the transaction graph. Therefore it is important to incentivize the participants to limit the exposure of their secret information. From a technical perspective, it is sufficient for offers to publish an aggregate imbalance, however our solution requires that the amounts and types of each input and output are revealed. Once a real transaction system with confidential types is deployed, it will be interesting to perform an analysis if it facilitates transaction graph reconstructions.

In a more general perspective, privacy-preserving UTXO systems, such as the ones we propose always fall short in pruning historical data. Because of the anonymity guarantee, all old data might be relevant. A more favorable transaction system has a state size linear in the number of active users and independent of time. Systems as Zether [BAZB20] and QuisQuis [FMMO19] solve this, but they are architecturally incompatible with our approach to non-interactive merging of transactions. The major difference is, that in these systems, it is important to always use the most recent version of an account. This prevents creating offers which are eventually matched in the future by when the inputs used for the offer are already obsolete.

After applying our security definitions of swap transactions to other scenarios, we noticed that in a more general setting, we either require that offers commute under merging, or redesign the oracle to increase the power of the adversary. To specify a precise, general security definition of swap transactions requires further research.

Apart from the technical issues, one can argue from an economical perspective that our offers, fixed in exchange rate, do not create an efficient market. Our system does not support the equivalent of market orders, where an exchange matches an offer at whatever price is currently available. This requires some form of malleable offers which we have not yet considered. Other directions to investigate are privacy-preserving automated market makers (AMM)

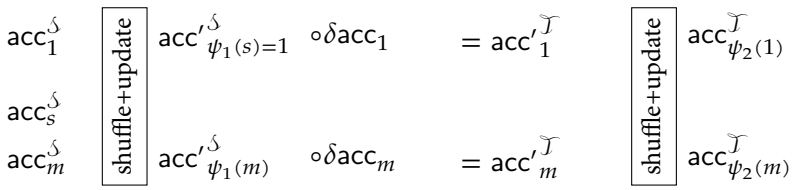


Figure 8.1: QuisQuis transaction overview.

[XVPC21] or secure multi-party computation based exchanges [BDF21] which distribute the trust over multiple entities.
To address some of the limitations we continue our research for possible solutions in follow up projects:

8.3 QUISQUIS AND KEY VALUE COMMITMENTS

One major issue with UTXO based privacy-preserving systems is the inability to decide if an output is already spent or still available. This prevents clients to prune the set of outputs and only store the unspent outputs. Therefore the state held by the clients has a monotonically increasing size.
QuisQuis [FMMO19] solves this problem by using updatable accounts. A QuisQuis transaction consumes a set of accounts $\mathsf{acc}^{\mathcal{S}}$ and creates a new set of accounts $\mathsf{acc}^{\mathcal{T}}$ (Figure 8.1). All old inputs are then pruned by the clients. To achieve set anonymity, the sender has to include foreign accounts as inputs. The transaction NIZK assures that only accounts to which the sender presents a valid key $\mathsf{acc}_s^{\mathcal{S}}$ decrease in value and all other accounts may only increase. The updates $\delta\mathsf{acc}$ hold positive amounts except for the first one, corresponding to the sender. The accounts are built with ElGamal style commitments, which are rerandomizeable and hide the permutation between inputs and outputs of a transaction.
One obstacle of integrating confidential types into a QuisQuis like system is the non-interactive update of accounts. In RingCT like systems, we are assured that each output holds exactly one type. This does not hold when other parties can deposit tokens of a new type to an existing account.
To support accounts with multiple types, we are currently working on a homomorphic key value commitment with efficient NIZK proofs. It allows commitments to scalar values for keys from a large universe (exponentially in the security parameter). The homomorphic operation on two commitments results in a commitment to the union of keys. Each value from the intersection of the openings will be added together. This property is equivalent to the intuitive deposit to a multi-type account. Whenever the type is already present, the value gets increased and types not present have an implicit balance of zero.
To build the authorization and conservation NIZK for a multi-type QuisQuis version, it is necessary to prove properties of all values v_k in a key value map

commitment without revealing the keys. We therefore formalize a generic NIZK on a arithmetic circuit C as

$$\mathsf{AoK}(\mathsf{stmt} = \{\mathsf{com}_i\}_{i=1}^n, \mathsf{wit} = \{\{v_{k,i}\}_{k\in K}\}_{i=1}^n \text{ s.t.}$$
$$C(\{\{v_{k,i}, k\}_{k\in K}\}_{i=1}^n) = 1 \wedge \forall i \in [n] : \mathsf{com}_i = \mathsf{Commit}(\{v_{k,i}\}_{k\in K}))$$

with a runtime linear in $|K|$ and a transcript size logarithmic in the number of multiplication gates in C. While we have a candidate construction, there is still work to be done on a rigorous proof of our protocol.

8.4 UNION ONLY SIGNATURES

While we previously just claimed other application of our anonymously aggregatable signature scheme, we discovered that it solves a property, most homomorphic signature schemes do not have. Johnson et al. [JMSW02] proposed the open problem if there exists a signature scheme which is homomorphic only to the union of messages, but not to their intersection. Such schemes are interesting in settings where multiple signers want to merge their signatures and only all messages together should be considered. A subset of the merged message might be incriminating and with a union only signature, a valid signature on a subset cannot be created without the signers' help. Given two signatures over the sets of messages U and V, the scheme allows non-interactive merging into a single signature of $U \cup V$. However, importantly, it was not possible to create a valid signature for $U \setminus V$. Molnar [Mol] proposed a solution based on groups with infeasible inversion (GII), however, we are not aware of a construction from standard assumptions. For their deduction, they assume that signatures are deterministic which does not apply for our scheme. Therefore our schemes does not provide a construction for GII.
Due to the anonymity and history hiding properties, it is suitable for aggregation of medical data in combination with differently incentivized parties. In most settings the signed messages are unique. This results in the same real union and multiset union. To achieve a real union of messages, where equal messages collapse when calculating the union, we might be able to describe a recursive SNARK.

8.5 OFFERS IN ZCASH

From the beginning of our project, we focused on a transparent setup which is important for public transaction systems. Still, our formalizations of the atomic swap exchange can easily be adapted to the anonymity model in Zcash. We aim to build Zcash based construction with even higher anonymity guarantees as SwapCT. What remains is the idea of aggregatable partial transactions. Given the efficiency of SNARKs provided by the trusted setup, we can integrate input output ambiguity, such that the transaction does not reveal if an account is used as input or output. In a cross over to our work on Multi-type QuisQuis, it is possible to have transaction outputs which hold amounts

in multiple types. With the homomorphic property of key value map commitments, offers are easily aggregated and the conservation is achieved by a Schnorr signature.
Such a Zcash based system might additionally achieve better anonymity guarantees of the offers themselves. SwapCT provides sender and receiver anonymity for offers but reveals the opening of the value and type commitments. In a Zcash like offer, we hope that it is sufficient to reveal a single aggregate opening for the offer, hiding its internal distribution. This was not possible in SwapCT as the sealing party requires the full opening to create the asset surjection proof. This surjection proof is replaced by a direct pre-image proof of a hash function in zero knowledge.
Multi-type and swap support in Zcash provides improved verifier runtime and is suitable for our licensing application. The participants of the licensing system might be cooperating in honestly creating the required setup.

8.6 CONCLUSION

In conclusion, our work on building privacy-preserving transaction systems which accommodate confidential types demonstrates that these systems are not limited to simple transfers of amounts. Complemented with research on improving type-unaware protocols, we notice a trend towards privacy-preserving systems with extensive functionality. This helps the participants to keep their personal information private while enabling them to participate in complex financial or other token-based operations. Augmenting tokens with additional attributes next to amounts is common practice in non-privacy-preserving transaction systems and there helps to create more efficient protocols like payment channels.
We see our work as an important step on the path towards privacy-friendly digital currencies. Embracing digital financial services is important for increasing the convenience and reducing friction in our society. However this convenience must not result in an intrusion into our private lives. Even though many fellow citizens care little about their digital footprint, their private information is used against their interest by companies extracting profit. Our privacy-preserving multi-type transaction system serves as tool to finely balance the required societal oversight and the privacy of every individual. We close the gap between siloed single-type systems and enable a unified approach, increasing anonymity and decreasing friction.

BIBLIOGRAPHY

PUBLICATIONS BY THE AUTHOR

Parts of this dissertation were previously published in the following articles.

[EHNS18] F. Engelmann, M. Holland, C. Nigischer, and J. Stjepandić. "Intellectual property protection and licensing of 3D print with blockchain technology." In: *Transdisciplinary Engineering Methods for Social Innovation of Industry 4.0: Proceedings of the 25th ISPE Inc. International Conference on Transdisciplinary Engineering, July 3–6, 2018*. Vol. 7. IOS Press. 2018, p. 103. Copyright: Open access, licensed under Attribution-NonCommercial 4.0 International (CC BY-NC 4.0), https://creativecommons.org/licenses/by-nc/4.0/deed.en_US (cit. on p. 6).

[EKB18] F. Engelmann, F. Kargl, and C. Bösch. "Coloured Ring Confidential Transactions." In: *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Ed. by J. Garcia-Alfaro, J. Herrera-Joancomartí, G. Livraga, and R. Rios. Cham: Springer International Publishing, 2018, pp. 188–196. ISBN: 978-3-030-00305-0. Copyright: © 2018 Springer Nature (cit. on pp. 6, 28).

[EKK+17] F. Engelmann, H. Kopp, F. Kargl, F. Glaser, and C. Weinhardt. "Towards an Economic Analysis of Routing in Payment Channel Networks." In: *Proceedings of the 1st Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*. SERIAL '17. Las Vegas, Nevada: ACM, 2017, 2:1–2:6. ISBN: 978-1-4503-5173-7. DOI: 10.1145/3152824.3152826. Copyright: © 2017 Copyright held by the owner/author(s). Publication rights licensed to Association for Computing Machinery. (Cit. on pp. 6, 73).

[EMP+21] F. Engelmann, L. Müller, A. Peter, F. Kargl, and C. Bösch. "SwapCT: Swap Confidential Transactions for Privacy-Preserving Multi-Token Exchanges." In: *Proceedings on Privacy Enhancing Technologies* 2021.4 (2021), pp. 270–290. DOI: doi:10.2478/popets-2021-0070. Copyright: © 2021 Felix Engelmann et al., published by Sciendo. This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 3.0 (CC BY-NC-ND 3.0) License, https://creativecommons.org/licenses/by-nc-nd/3.0/ (cit. on pp. 4–6, 26, 37, 79, 80, 82, 103, 121).

[ESG+21] F. Engelmann, J. P. Speichert, R. God, F. Kargl, and C. Bösch. "Confidential Token-Based License Management." In: *Proceedings of the 2021 Workshop on Additive Manufacturing (3D Printing) Security*. 2021, pp. 39–48. DOI: 10.1145/3462223.3485

619. Copyright: © 2021 Association for Computing Machinery. (Cit. on pp. 5, 6, 109, 110, 124).

[HEM+17] R. W. van der Heijden, F. Engelmann, D. Mödinger, F. Schönig, and F. Kargl. "Blackchain: Scalability for Resource-constrained Accountable Vehicle-to-x Communication." In: *Proceedings of the 1st Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*. SERIAL '17. Las Vegas, Nevada: ACM, 2017, 4:1–4:5. ISBN: 978-1-4503-5173-7. DOI: 10.1145/3152824.3152828. Copyright: © 2017 Copyright held by the owner/author(s). Publication rights licensed to Association for Computing Machinery. (Cit. on p. 6).

REFERENCES

[ACR20] T. Attema, R. Cramer, and M. Rambaud. "Compressed Σ-Protocols for Bilinear Group Arithmetic Circuits and Applications." In: 2020 (cit. on p. 19).

[AGN+19] O. Andreev, B. Glickstein, V. Niu, T. Rinearson, D. Sur, and C. Yun. *ZkVM: fast, private, flexible blockchain contracts*. Tech. rep. 2019 (cit. on p. 33).

[AH18] K. M. Alonso and J. Herrera-Joancomartí. "Monero - Privacy in the Blockchain." In: *IACR Cryptology ePrint Archive* (2018) (cit. on pp. 33, 91).

[Air19] Airbus. *Airbus achieves new commercial aircraft delivery record in 2018*. 2019. URL: https://www.airbus.com/newsroom/press-releases/en/2019/01/airbus-achieves-new-commercial-aircraft-delivery-record-in-2018.html (visited on 07/25/2019) (cit. on p. 123).

[Alt16] H.-H. Altfeld. *Commercial aircraft projects: Managing the development of highly complex products*. Routledge, 2016 (cit. on pp. 123, 124).

[And] O. Andreev. *Cloak*. URL: https://github.com/stellar/slingshot/blob/main/spacesuit/spec.md (cit. on pp. 10, 31).

[ARR21] T. Attema, R. Ronald Cramer, and M. Rambaud. "Compressed Sigma-Protocols for Bilinear Group Arithmetic Circuits and Application to Logarithmic Transparent Threshold Signatures." In: *ASIACRYPT 2021* (2021) (cit. on p. 48).

[BAZB20] B. Bünz, S. Agrawal, M. Zamani, and D. Boneh. "Zether: Towards privacy in a smart contract world." In: *International Conference on Financial Cryptography and Data Security*. Springer. 2020, pp. 423–443 (cit. on pp. 7, 128).

[BBB+18] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. "Bulletproofs: Short proofs for confidential transactions and more." In: *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2018 (cit. on pp. 10, 18, 50).

[BDF21] C. Baum, B. David, and T. K. Frederiksen. "P2DEX: Privacy-Preserving Decentralized Cryptocurrency Exchange." In: *Applied Cryptography and Network Security*. Ed. by K. Sako and N. O. Tippenhauer. Cham: Springer International Publishing, 2021, pp. 163–194. ISBN: 978-3-030-78372-3 (cit. on p. 129).

[BJ10] A. Bagherzandi and S. Jarecki. "Identity-Based Aggregate and Multi-Signature Schemes Based on RSA." In: *Public Key Cryptography – PKC 2010*. Ed. by P. Q. Nguyen and D. Pointcheval. Springer, 2010. ISBN: 978-3-642-13013-7 (cit. on p. 40).

[BNM+14] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten. "Mixcoin: Anonymity for bitcoin with accountable mixes." In: *International Conference on Financial Cryptography and Data Security*. Springer. 2014, pp. 486–504 (cit. on p. 7).

[CHL+17] X. Chen, J. Hao, H. Liu, Z. Han, and S. Ye. "Research on Similarity Measurements of 3D Models Based on Skeleton Trees." In: *Computers* 6.2 (2017), p. 17 (cit. on p. 115).

[CL06] M. Chase and A. Lysyanskaya. "On signatures of knowledge." In: *International Cryptology Conference*. Springer. 2006 (cit. on p. 18).

[DFM20] J. Don, S. Fehr, and C. Majenz. "The measure-and-reprogram technique 2.0: multi-round Fiat-Shamir and more." In: *Annual International Cryptology Conference*. Springer. 2020, pp. 602–631 (cit. on pp. 61, 92).

[dGTP17] J. M. de Fuentes, L. González-Manzano, J. Tapiador, and P. Peris-Lopez. "PRACIS: Privacy-preserving and aggregatable cybersecurity information sharing." In: *Computers & Security* (2017). Security Data Science and Cyber Threat Mgnt. ISSN: 0167-4048. DOI: 10.1016/j.cose.2016.12.011 (cit. on p. 40).

[DH20] A. Deshpande and M. Herlihy. "Privacy-Preserving Cross-Chain Atomic Swaps." In: *International Conference on Financial Cryptography and Data Security*. Springer. 2020 (cit. on pp. 31, 33).

[Dia21] B. E. Diamond. "Many-out-of-Many Proofs and Applications to Anonymous Zether." In: *2021 IEEE Symposium on Security and Privacy (SP)*. 2021, pp. 1800–1817. DOI: 10.1109/SP40001.2021.00026 (cit. on p. 4).

[DJV19] A. Dutta, A. Jana, and S. Vijayakumaran. "Nummatus: A Privacy Preserving Proof of Reserves Protocol for Quisquis." In: *Progress in Cryptology – INDOCRYPT 2019*. Ed. by F. Hao, S. Ruj, and S. Sen Gupta. Cham: Springer International Publishing, 2019, pp. 195–215. ISBN: 978-3-030-35423-7 (cit. on p. 122).

[Dou02] J. R. Douceur. "The Sybil Attack." In: *Peer-to-Peer Systems*. Ed. by P. Druschel, F. Kaashoek, and A. Rowstron. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 251–260. ISBN: 978-3-540-45748-0 (cit. on p. 122).

[DV19] A. Dutta and S. Vijayakumaran. "MProve: A Proof of Reserves Protocol for Monero Exchanges." In: *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*. 2019, pp. 330–339. DOI: 10.1109/EuroSPW.2019.00043 (cit. on p. 122).

[Dwo07] M. J. Dworkin. *Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC*. Tech. rep. 2007 (cit. on pp. 117, 123).

[FKP15] M. Fleder, M. S. Kester, and S. Pillai. "Bitcoin transaction graph analysis." In: *arXiv preprint arXiv:1502.01657* (2015) (cit. on p. 7).

[FMMO19] P. Fauzi, S. Meiklejohn, R. Mercer, and C. Orlandi. "Quisquis: A new design for anonymous cryptocurrencies." In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2019, pp. 649–678 (cit. on pp. 7, 128, 129).

[FOS19] G. Fuchsbauer, M. Orrù, and Y. Seurin. "Aggregate cash systems: A cryptographic investigation of mimblewimble." In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2019 (cit. on p. 33).

[FS] A. Fiat and A. Shamir. "How To Prove Yourself: Practical Solutions to Identification and Signature Problems." In: *Advances in Cryptology — CRYPTO' 86*. Ed. by A. M. Odlyzko (cit. on pp. 17, 61, 92).

[GI08] J. Groth and Y. Ishai. "Sub-linear zero-knowledge argument for correctness of a shuffle." In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2008, pp. 379–396 (cit. on p. 13).

[GKR+21] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger. "Poseidon: A new hash function for zero-knowledge proof systems." In: *30th {USENIX} Security Symposium ({USENIX} Security 21)*. 2021 (cit. on p. 4).

[GXK+19] Z. Gao, L. Xu, K. Kasichainula, L. Chen, B. Carbunar, and W. Shi. "Private and Atomic Exchange of Assets over Zero Knowledge Based Payment Ledger." In: *arXiv preprint arXiv:1909.06535* (2019) (cit. on pp. 31, 33, 79).

[HSKK01] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. "Topology matching for fully automatic similarity estimation of 3D shapes." In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM. 2001, pp. 203–212 (cit. on p. 115).

[Isi20] Isis Agora Lovecruft and Henry de Valence. *curve25519_dalek* https://doc.dalek.rs/curve25519_dalek/. Version 3. Nov. 30, 2020 (cit. on p. 104).

[JMSW02] R. Johnson, D. Molnar, D. Song, and D. Wagner. "Homomorphic signature schemes." In: *Cryptographers' track at the RSA conference*. Springer. 2002, pp. 244–262 (cit. on pp. 45, 130).

[KR08] T. H. Klier and J. M. Rubenstein. "Who really made your car?" In: *Employment Research Newsletter* 15.2 (2008), p. 1 (cit. on p. 124).

[Lam19] L. Lamport. "Time, Clocks, and the Ordering of Events in a Distributed System." In: *Concurrency: The Works of Leslie Lamport*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 179–196. ISBN: 9781450372701 (cit. on p. 122).

[Lin03] Y. Lindell. "Parallel coin-tossing and constant-round secure two-party computation." In: *Journal of Cryptology* 16.3 (2003) (cit. on p. 13).

[LMR19] R. W. Lai, G. Malavolta, and V. Ronge. "Succinct arguments for bilinear group arithmetic: Practical structure-preserving cryptography." In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019, pp. 2057–2074 (cit. on pp. 19, 48).

[LQQ+16] W. Lu, Y. Qin, Q. Qi, W. Zeng, Y. Zhong, X. Liu, and X. Jiang. "Selecting a semantic similarity measure for concepts in two different CAD model data ontologies." In: *Advanced Engineering Informatics* 30.3 (2016), pp. 449–466 (cit. on p. 115).

[LRR+19] R. W. Lai, V. Ronge, T. Ruffing, D. Schröder, S. A. K. Thyagarajan, and J. Wang. "Omniring: Scaling Private Payments Without Trusted Setup." In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019 (cit. on pp. 19, 23, 24, 26, 28, 33, 50, 82, 85, 92, 104).

[Man18] I. O. M. V. of Manufacturers. *2017 Production statistics*. 2018. URL: http://www.oica.net/category/production-statistics/2017-statistics/ (visited on 07/25/2019) (cit. on p. 124).

[Max15] G. Maxwell. *Confidential Transactions*. 2015. URL: https://elementsproject.org/features/confidential-transactions/investigation (cit. on p. 7).

[MBL+20] P. Moreno-Sanchez, A. Blue, D. V. Le, S. Noether, B. Goodell, and A. Kate. "DLSAG: non-interactive refund transactions for interoperable payment channels in monero." In: *International Conference on Financial Cryptography and Data Security*. Springer. 2020, pp. 325–345 (cit. on p. 72).

[MGGR13] I. Miers, C. Garman, M. Green, and A. D. Rubin. "Zerocoin: Anonymous Distributed E-Cash from Bitcoin." In: *2013 IEEE Symposium on Security and Privacy*. 2013, pp. 397–411. DOI: 10.1109/SP.2013.34 (cit. on p. 2).

[MM18] S. Meiklejohn and R. Mercer. "Möbius: Trustless tumbling for transaction privacy." In: (2018) (cit. on p. 7).

[Mol] D. Molnar. "Homomorphic Signature Schemes." MA thesis (cit. on p. 130).

[MPJ+13] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage. "A fistful of bitcoins: characterizing payments among men with no names." In: *Proceedings of the 2013 conference on Internet measurement conference*. 2013, pp. 127–140 (cit. on p. 7).

[MR11] D. Mun and K. Ramani. "Knowledge-based part similarity measurement utilizing ontology and multi-criteria decision making technique." In: *Advanced Engineering Informatics* 25.2 (2011), pp. 119–130 (cit. on p. 115).

[MSH+18] M. Möser, K. Soska, E. Heilman, K. Lee, H. Heffan, S. Srivastava, K. Hogan, J. Hennessey, A. Miller, A. Narayanan, et al. "An empirical analysis of traceability in the monero blockchain." In: *PoPETs* (2018) (cit. on pp. 25, 81, 82, 128).

[Nak08] S. Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. Tech. rep. 2008 (cit. on pp. 1, 7).

[NBC10] A. L. NBC News. *Hundreds of suppliers, one Boing 737 airplane*. 2010. URL: http://www.nbcnews.com/id/36507420/ns/business-us%5C_business/t/hundreds-suppliers-one-boeing-airplane/%5C#.XTmO4ZMzZaQ (visited on 07/25/2019) (cit. on p. 123).

[Noe15] S. Noether. "Ring Signature Confidential Transactions for Monero." In: *IACR Cryptology ePrint Archive* 2015 (2015), p. 1098 (cit. on pp. 2, 3, 7, 9).

[OKH13] M. Ober, S. Katzenbeisser, and K. Hamacher. "Structure and anonymity of the bitcoin transaction graph." In: *Future internet* 5.2 (2013), pp. 237–250 (cit. on p. 7).

[PBF+17] A. Poelstra, A. Back, M. Friedenbach, G. Maxwell, and P. Wuille. "Confidential assets." In: *Financial Cryptography Bitcoin Workshop*. 2017 (cit. on pp. 5, 10, 29, 30, 33, 47, 48).

[Ped91] T. P. Pedersen. "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing." In: *CRYPTO'91*. 1991 (cit. on p. 12).

[REo4] W. Rankl and W. Effing. *Smart card handbook*. John Wiley & Sons, 2004 (cit. on p. 117).

[REL+20] V. Ronge, C. Egger, R. W. F. Lai, D. Schröder, and H. H. F. Yin. "Foundations of Ring Sampling." In: (2020). https://eprint.iacr.org/2020/1550 (cit. on p. 25).

[RH13] F. Reid and M. Harrigan. "An analysis of anonymity in the bitcoin system." In: *Security and privacy in social networks*. Springer, 2013, pp. 197–223 (cit. on p. 7).

[RMK14] T. Ruffing, P. Moreno-Sanchez, and A. Kate. "Coinshuffle: Practical decentralized coin mixing for bitcoin." In: *European Symposium on Research in Computer Security*. Springer. 2014, pp. 345–364 (cit. on p. 7).

[RS13] D. Ron and A. Shamir. "Quantitative analysis of the full bitcoin transaction graph." In: *International Conference on Financial Cryptography and Data Security*. Springer. 2013, pp. 6–24 (cit. on p. 7).

[RS92] C. Rackoff and D. R. Simon. "Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack." In: *Advances in Cryptology — CRYPTO '91*. Ed. by J. Feigenbaum. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 433–444. ISBN: 978-3-540-46766-3 (cit. on p. 3).

[SCG+14] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. "Zerocash: Decentralized anonymous payments from bitcoin." In: *2014 IEEE Symposium on Security and Privacy*. IEEE. 2014, pp. 459–474 (cit. on pp. 2, 3, 7, 9, 23, 33).

[TMSS20] S. A. K. Thyagarajan, G. Malavolta, F. Schmidt, and D. Schröder. "PayMo: Payment Channels For Monero." In: (2020) (cit. on p. 72).

[Woo+14] G. Wood et al. "Ethereum: A secure decentralised generalised transaction ledger." In: *Ethereum project yellow paper* 151.2014 (2014), pp. 1–32 (cit. on p. 7).

[XVPC21] J. Xu, N. Vavryk, K. Paruch, and S. Cousaert. "SoK: Decentralized Exchanges (DEX) with Automated Market Maker (AMM) protocols." In: *ArXiv* abs/2103.12732 (2021) (cit. on p. 129).

[ZYD+20] Y. Zheng, H. Ye, P. Dai, T. Sun, and V. Gelfer. "Confidential Assets on MimbleWimble." In: *rin* 1000 (2020), p. 1 (cit. on p. 33).