Felix Engelmann*, Lukas Müller, Andreas Peter, Frank Kargl, and Christoph Bösch

# SwapCT: Swap Confidential Transactions for Privacy-Preserving Multi-Token Exchanges

**Abstract:** Decentralized token exchanges allow for secure trading of tokens without a trusted third party. However, decentralization is mostly achieved at the expense of transaction privacy. For a fair exchange, transactions must remain private to hide the participants and volumes while maintaining the possibility for non-interactive execution of trades. In this paper we present a swap confidential transaction system (SwapCT) which is related to ring confidential transactions (e.g. used in Monero) but supports multiple token types to trade among and enables secure, partial transactions for non-interactive swaps. We prove that SwapCT is secure in a strict, formal model and present its efficient performance in a prototype implementation with logarithmic signature sizes for large anonymity sets. For our construction we design an aggregatable signature scheme which might be of independent interest. Our SwapCT system thereby enables a secure and private exchange for tokens without a trusted third party.

**Keywords:** atomic swap, exchange, typed tokens

## 1 Introduction

Crypto-currencies are primarily used to trade tokens. Either they are traded between systems or within one environment, such as within Ethereum with its ERC20 [19] token standard. The thousands of ERC20 tokens with multiple billions USD of market capitalization and similar daily trading volumes[1] are used to trade e.g. reputation points, lottery tickets, shares of companies, fiat currency, precious metals and many more. While the token's security is backed by a distributed ledger (DLT) sometimes called (block)chain, most are traded on centralized exchanges which operate like stock exchanges. These central exchanges provide the required privacy for fair trading such that participants have no insight into a detailed order book or trades of competitors. Unfortunately, users have to trust the exchange operators which are susceptible to malicious activity like theft [6] or hacks for data exfiltration. Due to these reasons, self-governing exchanges based on consensus in a distributed ledger system are beneficial. There are plenty of smart contract based decentralized exchanges, however, to allow public auditing, they typically reveal details about the performed trades. With all bids and offers persisted on the DLT, the exchange is no longer private and any malicious participant in the system has the advantage of detailed insights into the order book and upcoming offers in the pool of unconfirmed DLT transactions (mempool). This enables e.g. front-running [10] attacks, where a miner who notices a large bid in the mempool may just persist an order of themselves first and then profit from selling it later to the prospective buyer. The underlying issues are persisted, public offers (smart contract calls) when ordered by an untrusted party.

Many applications which build on top of a token system require an efficient and fair exchange mechanism. Exchange happens between tokens of different types and between a type and the base currency to initially buy and sell the tokens or pay fees. Additionally, a large part of Ethereum's success is attributed to the possibility of creating arbitrary tokens independent of the base currency. However, all token types inherit the security of the whole Ethereum network, which is not the case for tokens deployed and maintained on their own ledger. We illustrate the benefits of a single DLT with swaps between types with a concrete example: a DLT-based license management system for 3D-printed objects where everyone can create new 3D object designs and license them to others for printing. Such transfers of licenses are represented by transactions on the DLT and each object type is represented by a separate token type. Introduc-

***Corresponding Author: Felix Engelmann:** Aarhus University, Denmark, E-mail: felix.engelmann@cs.au.dk or fe-research@nlogn.org
**Lukas Müller:** Ulm University, Germany, E-mail: lukas.mueller@alumni.uni-ulm.de
**Andreas Peter:** University of Twente, The Netherlands, E-mail: a.peter@utwente.nl
**Frank Kargl:** Ulm University, Germany, E-mail: frank.kargl@uni-ulm.de
**Christoph Bösch:** Ulm University, Germany, E-mail: christoph.boesch@uni-ulm.de

1 https://coinmarketcap.com/tokens/

**Table 1.** Overview of systems and their supported properties. Partial support of a feature is noted with (✓).

| | SRA | COA | COT | DSE | NIT | DEX |
|---|---|---|---|---|---|---|
| **MimbleWimble [12]** | | ✓ | ✓[22] | ✓ | | (✓) |
| **Confidential Assets [17]** | | ✓ | ✓ | ✓ | ✓ | |
| **Stellar [2]** | | ✓ | ✓ | ✓ | ✓ | |
| **Zcash [18]** | ✓ | ✓ | ✓[13] | | ✓ | (✓[13]) |
| **X-Chain Swap [8]** | ✓ | ✓ | | ✓ | ✓ | (✓) |
| **Monero [1]** | ✓ | ✓ | | ✓ | ✓ | |
| **Omniring [15]** | ✓ | ✓ | | ✓ | ✓ | |
| **Our SwapCT** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

ing a new object design, which happens frequently, involves introducing a new token type. Setting up and maintaining a separate chain for each token type generates a much higher overhead than a single multi type chain. In addition to performance, this example demonstrates our privacy and confidentiality requirements. If information about company A licensing a certain number of prints for a specific design is available on the DLT, then this means a leak of crucial information to competitors. Even the meta-data from the existence of private transactions on a chain dedicated for one design inevitably leaks information about business activities of company A. Therefore, we require a DLT with support for user anonymity and confidentiality for transactions that hide the token type and whether a transaction is a regular transfer or a swap.

To realize a privacy friendly multi-token system which provides similar operations as ERC20 tokens, its functionality has to include registration of new token types for any user and trading between all types. Most applications using ERC20 tokens, i.e. those pegged to fiat currency, rely only on these features and therefore can seamlessly switch to a privacy-preserving multi-token system. Trading needs to be **1. private**, such that only selected parties have insight into what is traded and **2. fair**, i.e. no malicious party can abort with a profit. For this, we identify the following six properties required by a privacy-preserving DLT 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. **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 allow a single message only from the offering party to e.g. an exchange. 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.

A transaction system with all the above properties provides a privacy friendly alternative for many existing applications currently based on public tokens. Additionally, new platforms with strong privacy requirements, as described in the above-mentioned licensing example, are easily developed on top of such a private system. Table 1 provides an overview of existing systems with support for a large subset of our required properties. We describe the details on these systems in Section 2.

Our goal is to design a transaction system, that satisfies all six aforementioned properties. While Omniring provides a thorough security definition for a single token system, there exists no formal description of a multi token system with swap transactions. Regarding a construction, the restriction on communication (NIT) requires participants of a swap to prepare a partial transaction signature which is mergeable into a transaction by an untrusted party, perfectly hiding the contributors of the final signature. To our knowledge, there is no signature scheme supporting this merge operation. Therefore, we propose a novel aggregatable signature scheme for this purpose.

To address all six properties combined for a decentralized exchange and to overcome the drawbacks of existing systems, we propose Swap Confidential Transactions (*SwapCT*), a decentralized and privacy-preserving transactions system for multiple token types supporting non-interactive, atomic swaps. SwapCT is based on unspent transaction outputs (UTXO) where tokens are stored in transaction outputs and spent by anonymously referencing these. All referenced outputs become invalid to prevent double spending. We make use of primitives proven efficient and secure in Monero [1] and Omniring [15] (DSE) such as logarithmically sized, linkable ring signatures for sender anonymity and one-time addresses for receiver anonymity (SRA). Instead of Pedersen commitments for amounts, we integrate typed homomorphic commitments similar to confidential assets to hide the amounts (COA) and types (COT). To prove conservation of types and amounts combined, we design an efficient zero-knowledge proof based on Bulletproofs [4].

For atomic swap transactions that are indistinguishable from simple transactions, we propose a transaction structure which uses our new aggregatable signature scheme. For such an atomic swap transaction, senders create offers which capture the intent to spend some tokens in return for different types of tokens. Offers provide sender and receiver anonymity and are not persisted on a DLT. For additional sender privacy, offers are mergeable non-interactively from original senders (NIT). An atomic swap is performed by merging two matching offers. If the offers do not match exactly, a third offer, prepared by the merger with their own inputs, is added. Merged offers are persisted as a single transaction, indistinguishable from regular transactions (DEX).
Our contributions can be summarized as follows:

- We specify a formal model for SwapCT systems which extends the type-less model of Omniring.
- We provide an efficient instantiation of our SwapCT system which natively supports multiple token types, large anonymity sets and swap transactions that are indistinguishable from regular transactions.
- We prove the security of our SwapCT protocol.
- To non-interactively merge partial transactions and hide the original participants, we propose a novel anonymously aggregatable signature scheme, which might be of independent interest.
- We implement our SwapCT scheme and provide performance numbers comparable to Monero demonstrating its efficiency.

## 2 Related Work

Previous research and commercial projects achieved some properties required for a private, decentralized exchange system as shown in Table 1.

Zcash [18] is a production system providing user privacy and amount confidentiality without support for types. It hides amounts in commitments and references previous outputs through zero-knowledge proofs. The efficiency of Zcash's zk-SNARK proofs is dependent on a trusted setup. Recipients' identities are protected by one-time addresses. To facilitate trade between such single type systems, there are multiple approaches, summarized by Zamyatin et al. [21], even in a private setting with privacy-preserving cross-chain swaps [8]. Cross-chain transactions generally create one transaction on each chain which privately reference each other. Only if both transactions are valid, the tokens are exchanged. Independent single-type DLTs with cross-chain swaps allow for flexible governance and implementations optimized for the specific token's challenges. As cross-chain swaps require a new DLT for each token they cannot provide type confidentiality.

Poelstra et al. [17] propose confidential assets, a commitment scheme to provide confidential types and confidential amounts. The interactive transaction generation of MimbleWimble [12] allows for a direct replacement of the single type commitment scheme with confidential assets, demonstrated by Zheng et al. [22]. Instead of using addresses to non-interactively receive tokens, the MimbleWimble transaction generation is an interactive protocol between sender and receivers, reducing the anonymity of both.

Stellar is an already deployed transaction system with multiple asset types. It uses ZkVM [2] transactions to enable confidential transactions that satisfy an arbitrary arithmetic circuit. To create a transaction, the sender provides a proof of correctness according to the constraint system. However, the system does not provide sender anonymity or non-interactive swaps.

Gao et al. propose a private atomic exchange [13] where they introduce the notion of debt which has to be settled on spending. The authors claim that their model can be instantiated in a transparent setup, and provide a Zcash [18] based version without a performance evaluation or implementation. Their system requires persisting multiple transactions per swap, increasing the overall cost and duration.

Monero [1], Omniring [15] and RingCT 3.0 [20] are privacy-preserving transaction systems with a transparent setup, sender and receiver anonymity, non-interactive transactions and amount confidentiality. These systems use anonymity sets to hide senders and generate one-time addresses for receiver anonymity. Their transaction verification proves conservation of a

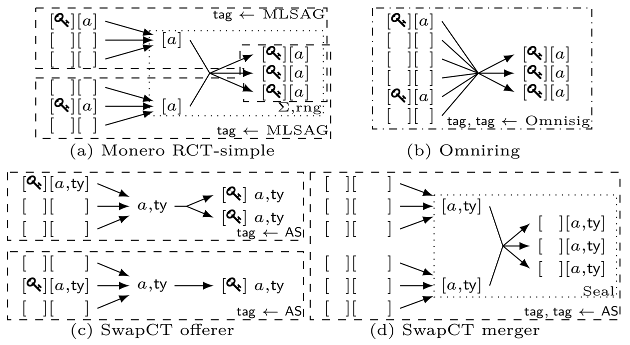

**Fig. 1.** Transaction structures for (a) Monero RCT-simple, (b) Omniring, (c) two SwapCT offers and (d) one SwapCT transaction. $a$ represent arbitrary amounts, ty is any type. Dashed lines are signatures to authorize spending and dotted lines ensure wealth conservation. $[\cdot]$ denotes a commitment to the value and if not blank, the value is known to the signer.

single currency and has no notion of types. Without multiple token types, they do not support atomic swaps.

## 3 Private UTXO Transactions

In this section we provide an overview of different approaches to UTXO based transaction structures and describe how our new SwapCT structure relates to them. Figure 1 shows the transaction structures most similar to our scheme ((a) Monero RCT-simple and (b) Omniring). All examples use 2 inputs and 3 outputs. The visibility of secrets is from the senders perspective. To achieve sender anonymity, both schemes rely on an anonymity set where each real input $[\ ][a]$ of the sender on the left is hidden in a set of other possible inputs, denoted by empty brackets $[\ ][\ \ ]$. For the real inputs, a tag is provided such that reusing the same input is detectable. The consensus algorithm maintains only a list of spent tags and rejects all transactions with a tag already used as double spend attempts. While Monero uses an independent ring signature for each input, Omniring, true to its name, uses a joint ring as anonymity set for all inputs. The outputs of a transaction, on the right in all schemes, are one-time accounts derived from recipient identities by the sender.

Transactions are signed to attest the intent to spend the real input to designated outputs by proving knowledge of the inputs' private keys and to ensure that the cumulative wealth in the output is equal to the one in the inputs. Omniring uses an all encompassing ring signature (Omnisig) to prove both requirements. Monero first copies the amounts into intermediate commitments $[a]$. Then, MLSAG [1] is used as the ring signature which proves that the signer knows the inputs' secret keys and that the intermediate commitment is equal to the real input. Wealth conservation is then proven by a Schnorr signature $\Sigma$ over intermediate and output commitments and Bulletproof range proofs [4] (rng) in Figure (1a).

Monero as well as Omniring require the final set of outputs 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 (Figure 1 (c,d)) which slightly resembles Monero. Figure (1c) shows two independent parties (upper and lower), each with one input $[\ ][a]$. However, instead of directly signing inputs with ring signatures, using all outputs concatenated as signing message, we propose a new anonymously aggregatable signature (AS) 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 $[\ ]\ a$,ty which are joined by merging offers. We use the intermediate copies for amounts and types $a$,ty as Monero does for amounts $[a]$ 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. 2. The offer is sent to a peer who created a matching offer. The offers are merged 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 lightweight 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 the final set of outputs and intermediate commitments. 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 DLT 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.

These are similar requirements to other privacy preserving DLTs such as Zcash and Monero and allow SwapCT to be readily deployed on a clone of their DLT infrastructure. Alternatively SwapCT can be integrated into a blockchain framework such as https://substrate.dev. Substrate is a rust framework which provides modules to operate a blockchain. Our rust prototype implementation of SwapCT is easily adapted to form the transaction generation and validation module, as it already has an interface close to substrate's architecture. The remainder of necessary components for a blockchain, such as peer-to-peer network with discovery, block generation and dissemination is provided by the framework.

For efficient ring signatures and conservation proofs in SwapCT, we adapt an elegant Bulletproof non-interactive, zero-knowledge approach, which is similar to the single signature in Omniring. With an anonymity set size of $r$ and $m$ inputs, this results in transaction sizes of SwapCT between the logarithmic efficiency of Omniring $\mathcal{O}(\log(m \cdot r))$ and the linear efficiency of Monero $\mathcal{O}(m \cdot r)$. We achieve a transaction size logarithmic in the size of the anonymity set per input $\mathcal{O}(m \log(r))$ because of the non-interactive ring signature creation.

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. [16] 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 40 to 60. The effective untraceability is the anonymity set expected after running statistical deanonymization attacks on the transaction graph.

To show the usual interaction of the algorithms of our SwapCT system, we describe a swap between Alice and Bob in Figure 2. The system is transparently set up and to participate both users generate a long term key with $\mathsf{KeyGen}$ serving as their identity. As there are no types registered yet, both create their own type with $\mathsf{TypeGen}$ and set themselves as recipients. The total supply of each type is fixed in this operation. The consensus accepts the registration of a new token type as the identifiers $(\mathrm{g}, \mathrm{r})$ are unique regarding all previous registrations. 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 $\mathsf{acc}_A$ and give her 5 red tokens. She authorizes the tentative spending of $\mathsf{acc}_A$ by creating an offer $\mathsf{off}_A$ with inputs $\mathcal{S}_A$, outputs $\mathcal{T}_A$ and ring members $\mathcal{R}_A$ signed by $\mathsf{Offer}$. Bob creates a complementary offer, which trades 5 red tokens for 2 green tokens. The spending authorization consists of publishing a $\mathsf{tag}$ corresponding to the input account. The offers are then merged with $\mathsf{Merge}$ by either an independent Merger, Alice or Bob. 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.

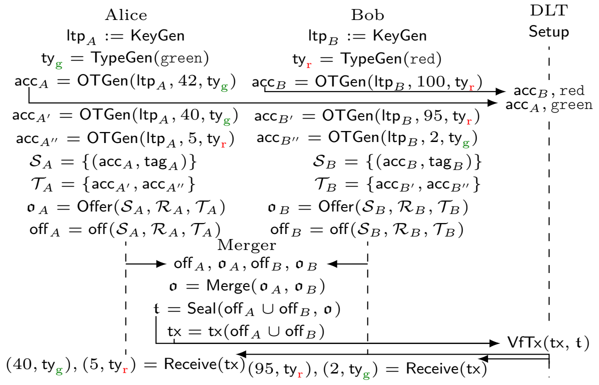
Alice
Bob
DLT
Setup
$\mathsf{ltp}_A := \mathsf{KeyGen}$
$\mathsf{ltp}_B := \mathsf{KeyGen}$
$\mathsf{ty}_{\mathrm{g}} = \mathsf{TypeGen}(\mathtt{green})$
$\mathsf{ty}_{\mathrm{r}} = \mathsf{TypeGen}(\mathtt{red})$
$\mathsf{acc}_A = \mathsf{OTGen}(\mathsf{ltp}_A, 42, \mathsf{ty}_{\mathrm{g}})$
$\mathsf{acc}_B = \mathsf{OTGen}(\mathsf{ltp}_B, 100, \mathsf{ty}_{\mathrm{r}})$
$\mathsf{acc}_B, \mathtt{red}$
$\mathsf{acc}_A, \mathtt{green}$
$\mathsf{acc}_{A'} = \mathsf{OTGen}(\mathsf{ltp}_A, 40, \mathsf{ty}_{\mathrm{g}})$
$\mathsf{acc}_{B'} = \mathsf{OTGen}(\mathsf{ltp}_B, 95, \mathsf{ty}_{\mathrm{r}})$
$\mathsf{acc}_{A''} = \mathsf{OTGen}(\mathsf{ltp}_A, 5, \mathsf{ty}_{\mathrm{r}})$
$\mathsf{acc}_{B''} = \mathsf{OTGen}(\mathsf{ltp}_B, 2, \mathsf{ty}_{\mathrm{g}})$
$\mathcal{S}_A = \{(\mathsf{acc}_A, \mathsf{tag}_A)\}$
$\mathcal{S}_B = \{(\mathsf{acc}_B, \mathsf{tag}_B)\}$
$\mathcal{T}_A = \{\mathsf{acc}_{A'}, \mathsf{acc}_{A''}\}$
$\mathcal{T}_B = \{\mathsf{acc}_{B'}, \mathsf{acc}_{B''}\}$
$\mathfrak{o}_A = \mathsf{Offer}(\mathcal{S}_A, \mathcal{R}_A, \mathcal{T}_A)$
$\mathfrak{o}_B = \mathsf{Offer}(\mathcal{S}_B, \mathcal{R}_B, \mathcal{T}_B)$
$\mathsf{off}_A = \mathsf{off}(\mathcal{S}_A, \mathcal{R}_A, \mathcal{T}_A)$
$\mathsf{off}_B = \mathsf{off}(\mathcal{S}_B, \mathcal{R}_B, \mathcal{T}_B)$
Merger
$\mathsf{off}_A, \mathfrak{o}_A, \mathsf{off}_B, \mathfrak{o}_B$
$\mathfrak{o} = \mathsf{Merge}(\mathfrak{o}_A, \mathfrak{o}_B)$
$\mathfrak{t} = \mathsf{Seal}(\mathsf{off}_A \cup \mathsf{off}_B, \mathfrak{o})$
$\mathsf{tx} = \mathsf{tx}(\mathsf{off}_A \cup \mathsf{off}_B)$
$\mathsf{VfTx}(\mathsf{tx}, \mathfrak{t})$
$(40, \mathsf{ty}_{\mathrm{g}}), (5, \mathsf{ty}_{\mathrm{r}}) = \mathsf{Receive}(\mathsf{tx})$
$(95, \mathsf{ty}_{\mathrm{r}}), (2, \mathsf{ty}_{\mathrm{g}}) = \mathsf{Receive}(\mathsf{tx})$

**Fig. 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 [16].

A balanced offer, where the input and output is equal, is then sealed with $\mathsf{Seal}$ and submitted to the consensus for verification ($\mathsf{VfTx}$). The transaction is persisted, if the signatures are valid and all $\mathsf{tags}$ 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 $\mathsf{Receive}$ and usable as future transaction inputs.

## 4 Formalizing Swap Transactions

In our SwapCT system, we distinguish the following roles in our security model, where each participant of the system may hold multiple roles at once and over time. We model the security of our system on top of an anonymous broadcast peer-to-peer network such as I2P or libp2p using any consensus algorithm for transaction ordering which prevents information leakage through meta-data. For transaction verification, nodes need access to the full history of transactions. **Verifier:** All participants verify persisted transactions. In this role everyone acts honestly as all their outgoing messages are verified by the consensus rules. However they try to deanonymize users or reveal transferred amounts and types. **Offerer:** A participant creating an offer is modeled as malicious as they have incentives to steal or create funds. **Merger/Exchange:** Anyone matching, merging and sealing offers into a transaction is modeled as malicious. Their goals are to deanonymize participants from their offers or redirect funds in an offer to themselves. When sealing an offer to create a transaction, the mergers are trusted to scrub the offers off the plaintext values and types, which were required to match offers together. Publishing the amounts and types reduces the anonymity set. Therefore, we require multiple, non-colluding mergers, but each of them may act maliciously. Knowing a fraction of plaintext amounts and types (the ones merged by the adversary) in the graph of many transactions from other mergers and direct transactions is not sufficient to deanonymize users. In case of a single merger for the whole SwapCT system, this merger acts as a central gateway to the ledger and can partially deanonymize the transaction graph and trace the amounts and types of transactions they have merged. There, the only anonymity set left is the set of direct transactions submitted to the ledger by users, bypassing the single merger. In summary, the anonymity set of offers depends on the ratio of uncompromized transactions persisted on the ledger.

First, we introduce some general notation. Given a security parameter $\lambda \in \mathbb{N}$, we denote $\mathsf{negl}(\lambda)$ as negligible in $\lambda$, equivalent to $\frac{1}{\mathsf{poly}(\lambda)}$. To sample an element $x$ uniformly at random from the set $S$, we write $x \xleftarrow{\$} S$. The set of integers up to $n$ is defined as $[n] := \{1, \dots, n\}$ and in general, sets are ordered if not specified otherwise. For any set $S$ we define $\sum S := \sum_{s \in S} s$. If an algorithm fails, it will always abort with $\perp$. We write PPT for probabilistic polynomial time complexity.

The security of single-type set anonymous systems is well formalized by Omniring [15]. We extend this formalization to encompass type support and swap transactions. Parts in blue are very similar to the Omniring description, e.g. only the type added, and are included for better understanding and to conclude rigorous security proofs where adding a type to Omniring's definitions may be ambiguous.

**Definition 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:*

$\underline{\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\alpha, 1^\beta)}$ takes the security parameter $\lambda$ and integers $\alpha$ for a maximum of $2^\alpha$ outputs of a transaction where each has an amount maximum of $2^\beta - 1$. Then it outputs public parameters $\mathsf{pp}$ which are implicitly given to all the following algorithms. Setup is called once when a SwapCT system is initialized.
$\underline{(\mathsf{ltp}, \mathsf{lts}) \leftarrow \mathsf{KeyGen}()}$ generates a long term secret key $\mathsf{lts}$ with the corresponding long term public key $\mathsf{ltp}$ for participants to initially join the system. The $\mathsf{ltp}$ is distributed and serves as a recipient address.
$\underline{\mathsf{ty} \leftarrow \mathsf{TypeGen}(\mathsf{name})}$ generates a type $\mathsf{ty}$ given a $\mathsf{name}$.
$\underline{\mathsf{acc}, \mathsf{ck} \leftarrow \mathsf{OTGen}(\mathsf{ltp}, a, \mathsf{ty})}$ creates a one-time account $\mathsf{acc}$ with coin key $\mathsf{ck}$ from a long term public key $\mathsf{ltp}$ and an amount $a$ of a type $\mathsf{ty}$ to then use this account as an output in an offer or a new type registration.
$\underline{\mathfrak{o} \leftarrow \mathsf{Offer}(\mathcal{S}, \mathcal{R}, \mathcal{T})}$ 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 $\mathsf{tag}_i$ corresponding to $\mathsf{acc}_{i,j_i}^\mathcal{R}$ at index $j_i \in [|\mathcal{R}_i|]$, secret key $\mathsf{sk}_i$, 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 $\mathfrak{o}$ as authorization to spend the inputs.
$\underline{b \leftarrow \mathsf{VfOffer}(\mathsf{off}, \mathfrak{o})}$ takes a signature $\mathfrak{o}$ and an offer

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

and returns a bit $b$ specifying if the offer is valid.
$\underline{\mathfrak{o} \leftarrow \mathsf{Merge}(\mathfrak{o}_0, \mathfrak{o}_1)}$ takes two offer signatures and generates a combined one $\mathfrak{o}$ valid for the union of both offers.
$\underline{\mathsf{t} \leftarrow \mathsf{Seal}(\mathsf{off}, \mathfrak{o})}$ Takes a valid balanced offer defined as above and its signature $\mathfrak{o}$ and outputs a seal proof $\mathsf{t}$.
$\underline{b \leftarrow \mathsf{VfTx}(\mathsf{tx}, \mathsf{t})}$ 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 $\mathsf{t}$ and returns a bit $b$ representing the validity.
$\underline{(\mathsf{tag}, \mathsf{sk}, a, \mathsf{ty}, \mathsf{ck}) \leftarrow \mathsf{Receive}(\mathsf{acc}, \mathsf{lts})}$ gets an account $\mathsf{acc}$ and a long term secret $\mathsf{lts}$ and returns the matching $\mathsf{tag}$, secret key $\mathsf{sk}$, amount $a$, type $\mathsf{ty}$ and coin key $\mathsf{ck}$ for $\mathsf{acc}$.

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, 0 otherwise.

**Definition 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* $a \in \{0, \ldots, 2^\beta - 1\}$, *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:*

- $|\mathcal{T}| \le 2^\alpha$, $\forall i \in [|\mathcal{T}|] : a^{\mathcal{T}}_i \in \{0, \ldots, 2^\beta - 1\}$
- $\forall \mathsf{ty} \in \{\mathsf{ty}^{\mathcal{T}}_i\}^{|\mathcal{T}|}_{i=1}$ :
$$\sum \{a^{\mathcal{S}}_i | \mathsf{ty}^{\mathcal{S}}_i = \mathsf{ty}\}^{|\mathcal{S}|}_{i=1} = \sum \{a^{\mathcal{T}}_i | \mathsf{ty}^{\mathcal{T}}_i = \mathsf{ty}\}^{|\mathcal{T}|}_{i=1}$$

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

## 5 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 [15], which allows an attacker to make arbitrary transactions through oracles and who must then output a valid transaction which uses a tag of the 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.

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 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*

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

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

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

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

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

**Fig. 3.** Balance experiment

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 3 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 $\mathsf{tags}$ match the inputs $\mathsf{acc}_{i,j_i}^\mathcal{R}$. In addition, the amounts and types must match the input and output accounts with $\mathsf{ChkAcc}$. 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 $\mathsf{tag}$ or change the amount or type.

**Definition 4** (Balance)**.** *A SwapCT scheme is balanced if it prevents theft (Def. 3) 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* $\Pr[\mathrm{Balance}_{\mathcal{A},\mathcal{E}_\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta) = 1] \leq \mathsf{negl}(\lambda)$ *with* $\mathrm{Balance}_{\mathcal{A},\mathcal{E}_\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta)$ *defined in Figure 3.*

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

**Fig. 4.** Oracles for the privacy experiments

## 6 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 transaction.

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 $\mathsf{OffPv}^b$ in Figure 5 with a bit $b \in \{0, 1\}$. An adversary $\mathcal{A}$ queries the available oracles (Figure 4) 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 iden-

```
$\mathsf{OffPv}^b_{\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta)$
    $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\alpha, 1^\beta), \mathsf{InitOracles}()$
    $\mathbb{O} = \{\mathsf{KeyGen}\mathcal{O}, \mathsf{Offer}\mathcal{O}\}$
    $(I, J, \mathsf{off}, \mathfrak{o}) \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{pp})$
    $\mathfrak{o}_0 := \mathfrak{o}_1 := \mathfrak{o}, \mathsf{off}_0 := \mathsf{off}_1 := \mathsf{off}, \mathcal{S}_0 := \mathcal{S}_1 := \mathcal{T}_0 := \mathcal{T}_1 := \emptyset$
    if $\mathsf{VfOffer}(\mathsf{off}, \mathfrak{o}) = 0$ then return $0$
    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 $0$
        if $a_{0,i} \neq a_{1,i} \vee \mathsf{ty}_{0,i} \neq \mathsf{ty}_{1,i}$ then return $0$
    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 $0$
        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 $0$
            $\mathsf{off}_t := \mathsf{off}_t \cup \mathsf{off}(\mathcal{S}^k_t, \mathcal{R}^k_t, \mathcal{T}^k_t)$
            $\mathfrak{o}_t \leftarrow \mathsf{Merge}(\mathfrak{o}_t, \mathfrak{o}^k_t)$
    $b' \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{off}_b, \mathfrak{o}_b)$ return $b'$
```

**Fig. 5.** Offer privacy experiment

tifier of the party who should use input $i$ depending on the selected bit $b$. The set of all input identifiers $\{u^{\mathcal{S}}_{t,i}\}^{|I|}_{i=1}$ 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 $\{\mathsf{acc}^{\mathcal{R}}_{i,j}\}^{|\mathcal{R}_i|}_{j=1}$ 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 $\{(\{u^{\mathcal{T}}_{t,i}, k^{\mathcal{T}}_{t,i}\}^1_{t=0}, a^{\mathcal{T}}_i, \mathsf{ty}^{\mathcal{T}}_i)\}^{|J|}_{i=1}$ 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 $\mathsf{off}^b_k$, along with the malicious $\mathsf{off}$ are merged. The adversary wins by correctly guessing the bit $b$.

```
$\mathsf{TxPv}^b_{\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta)$
    $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\alpha, 1^\beta), \mathsf{InitOracles}()$
    $\mathbb{O} = \{\mathsf{KeyGen}\mathcal{O}, \mathsf{Offer}\mathcal{O}\}$
    $(I, J, \mathsf{off}, \sigma) \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{pp})$
    $\mathcal{S}_0 := \mathcal{T}_0 := \mathcal{S}_1 := \mathcal{T}_1 := \mathcal{R} := \emptyset$
    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 $0$
    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
        $\mathfrak{o}_t \leftarrow \mathsf{Merge}(\mathfrak{o}, \mathsf{Offer}(\mathcal{S}_t, \mathcal{R}, \mathcal{T}_t))$
        $\mathsf{off}_t := \mathsf{off}(\mathcal{S}_t, R, \mathcal{T}_t) \cup \mathsf{off}$
        $\mathsf{tx}_t := \mathsf{tx}(\mathsf{off}_t)$
        $\mathfrak{t}_t \leftarrow \mathsf{Seal}(\mathsf{off}_t, \mathfrak{o}_t)$
        if $\mathsf{VfOffer}(\mathsf{off}_t, \mathfrak{o}_t) = 0 \vee \mathsf{VfTx}(\mathsf{tx}_t, \mathfrak{t}_t) = 0$ then return $0$
    $b' \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{tx}_b, \mathfrak{t}_b)$ return $b'$
```

**Fig. 6.** Transaction privacy experiment

**Definition 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 5.*

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 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 then seals the merged transaction at the end.

**Definition 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 $\left|\Pr[\mathsf{TxPv}^0_{\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta) = 1] - \Pr[\mathsf{TxPv}^1_{\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta) = 1]\right| \leq \mathsf{negl}(\lambda)$ with $\mathsf{TxPv}^b_{\mathcal{A}}(1^\lambda, 1^\alpha, 1^\beta)$ defined in Figure 6.*

## 7 SwapCT Components

Our construction of a SwapCT system as defined above depends on the following six components. (1) We make use of the Omniring tagging scheme [15] for double-spend detection, which 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 $\psi$. TagKGen is homomorphic, i.e. for any $x, x' \in \chi, \mathsf{TagKGen}(x) \cdot \mathsf{TagKGen}(x') = \mathsf{TagKGen}(x + x')$. TagEval takes a key $x \in \chi$ and outputs a tag $\in \mathcal{X}$. We require related-input one-wayness and pseudorandomness as defined in Omniring. (2) Further we make use of a labeled public-key encryption scheme $\mathsf{PKE} = (\mathsf{PKESetup}, \mathsf{PKGen}, \mathsf{Enc}, \mathsf{Dec})$ which is IND-CCA and IK-CCA secure. (3) We base a typed homomorphic commitment on the idea of confidential assets [17] to hide values and types. It provides the algorithms THC =(ComSetup, ComTypeGen, Commit) defined as

$\underline{\mathsf{pp} \leftarrow \mathsf{ComSetup}(1^\lambda)}$ generates the public parameters for the rest of the algorithms with a randomness space $\mathbb{R}$, a type space $\mathbb{T}$ and message space $\mathbb{M}$.

$\underline{\mathsf{ty} = \mathsf{ComTypeGen}(i)}$ takes an identifier $i \in \{0,1\}^*$ and outputs a base type $\mathsf{ty} \in \mathbb{T}$

$\underline{\mathsf{com} = \mathsf{Commit}(\mathsf{ty}, v; r)}$ takes a type $\mathsf{ty} \in \mathbb{T}$ and a value $v \in \mathbb{M}$ with randomness $r \in \mathbb{R}$ and outputs a commitment com. We require the following homomorphic property, that $\mathsf{Commit}(\mathsf{ty}, v; r) \odot \mathsf{Commit}(\mathsf{ty}, v'; r') = \mathsf{Commit}(\mathsf{ty}, v + v'; s)$ with $s \in \mathbb{R}$ efficiently computable.

The security requirements to a typed homomorphic commitment are similar to Pedersen commitments and we therefore require binding and hiding properties, detailed in Appendix A. (4) We define a Tagged Ring Signature as a Signature of Knowledge (SoK) to anonymously authorize spending. (5) We define a Seal signature to confidentially prove the balance of an offer, and (6) we specify an anonymously aggregatable signature scheme, which is parameterized with the tagged ring signature and binds outputs to authorized inputs.

### 7.1 Signatures of Knowledge

For a generalized description of the tagged ring signature and the seal signature, we use signatures of knowledge (SoK) which are efficiently constructable from arguments of knowledge (AoK) [5, 15] by including a message in the Fiat-Shamir transformation [11]. They consist of the algorithms $\mathsf{pp} \leftarrow \mathsf{SoK}[\mathcal{L}]\mathsf{Setup}$, $\sigma \leftarrow \mathsf{SoK}[\mathcal{L}]\mathsf{Sign}(\mathsf{stmt}, \mathsf{wit}, m)$, $b \leftarrow \mathsf{SoK}[\mathcal{L}]\mathsf{Verify}(\sigma, \mathsf{stmt}, m)$, $\sigma \leftarrow \mathsf{SoK}[\mathcal{L}]\mathsf{Sim}(\mathsf{stmt}, m))$ for any NP language $\mathcal{L}$. They fulfill the properties completeness, simulatability and extractability according to [5].

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 $\mathsf{pk}_j$. 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 [1]. 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 $\mathsf{com}'$ commits to a type ty and amount $a$, which is the same as the amount and type in $\mathsf{com}_j$, referenced by tag but has a different coin key $\mathsf{ck}'$ to hide the link to $\mathsf{com}_j$. 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 $\mathsf{com}_j$ and $\mathsf{com}'$.

The seal signature is a SoK convincing a verifier in zero knowledge that an offer is balanced. Given a set of temporary commitments $\{\mathsf{com}'_i\}_{i=1}^{|\mathcal{S}|}$ and a set of output commitments $\{\mathsf{com}_j\}_{j=1}^{|\mathcal{T}|}$, a valid signature convinces an honest verifier, that for each type ty present in the transaction outputs $\{\mathsf{ty}_j^{\mathcal{T}}\}_{j=1}^{|\mathcal{T}|}$ the sum of amounts in the inputs $\sum\{a_i^{\mathcal{S}} | \mathsf{ty}_i^{\mathcal{S}} = \mathsf{ty}\}_{i=1}^{|\mathcal{S}|}$ is equal to the sum of amounts in the outputs $\sum\{a_i^{\mathcal{T}} | \mathsf{ty}_i^{\mathcal{T}} = \mathsf{ty}\}_{i=1}^{|\mathcal{T}|}$. We specify the seal signature with the following language $\mathcal{L}^{\mathsf{seal}}$

$$:= \begin{cases} \mathsf{stmt} = (\{\mathsf{com}'_i\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{com}_j^{\mathcal{T}}\}_{j=1}^{|\mathcal{T}|}) : \\ \exists \mathsf{wit} = (\{\mathsf{ty}_i^{\mathcal{S}}, a_i^{\mathcal{S}}, \mathsf{ck}'_i\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{ty}_j^{\mathcal{T}}, a_j^{\mathcal{T}}, \mathsf{ck}_j^{\mathcal{T}}\}_{j=1}^{|\mathcal{T}|}) \text{ s.t.} \\ \forall i \in [|\mathcal{S}|] : \mathsf{com}'_i = \mathsf{Commit}(\mathsf{ty}_i^{\mathcal{S}}, a_i^{\mathcal{S}}; \mathsf{ck}'_i) \\ \forall j \in [|\mathcal{T}|] : \mathsf{com}_j^{\mathcal{T}} = \mathsf{Commit}(\mathsf{ty}_j^{\mathcal{T}}, a_j^{\mathcal{T}}; \mathsf{ck}_j^{\mathcal{T}}) \\ \forall \mathsf{ty} \in \{\mathsf{ty}_j^{\mathcal{T}}\}_{j=1}^{|\mathcal{T}|} : \\ \qquad \sum\{a_i^{\mathcal{S}} | \mathsf{ty}_i^{\mathcal{S}} = \mathsf{ty}\}_{i=1}^{|\mathcal{S}|} = \sum\{a_i^{\mathcal{T}} | \mathsf{ty}_i^{\mathcal{T}} = \mathsf{ty}\}_{i=1}^{|\mathcal{T}|} \end{cases}$$

From $\mathcal{L}^{\mathsf{seal}}$, we see that with a binding THC scheme, the SoK signature is secure and assures the balance of two sets of commitments.

## 7.2 Aggregatable Anonymous Signature

To create offers independently and merge multiple offers into a transaction, we require a high level of privacy such that the final transaction does not leak the individual parties' inputs. However, we also require an authorization, such that signatures of offers cannot be abused to authorize spending of the tokens to different recipients.

Without the privacy requirement, a solution to achieve authorization of inputs spending to a fixed set of outputs is simple. Each authorization signature signs a message that is the subset of outputs required to be in the final transaction. On verification, the signature is invalid, if an output is missing or changed. However, this reveals the mapping between the signers and outputs.

The generalized problem is a set of signers, each with a set of messages $\{m_j\}_{j=1}^{|\mathcal{T}|}$ (representing transaction outputs) and a set of statements and witnesses $\{(\mathsf{stmt}_i, \mathsf{wit}_i)\}_{i=1}^{|\mathcal{S}|}$ for a SoK (representing authorization signatures on inputs). 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.

As the aggregation is agnostic to the signature scheme, we describe it with the notation of a SoK as defined above. An independent use-case are e.g. authors each writing and signing a chapter. They may then later claim to have participated in the whole book without revealing which part of it they wrote and signed.

An aggregatable signature scheme consists of $\mathsf{AS} = (\mathsf{AsSetup}, \mathsf{AsSign}, \mathsf{AsVerify}, \mathsf{AsMerge}, \mathsf{AsSim})$ parametrized with an NP Language $\mathcal{L}$ and its relation $\mathbf{R}_{\mathcal{L}}$.

$\underline{\mathsf{pp} \leftarrow \mathsf{AsSetup}(1^\lambda, \mathcal{L})}$ takes the security parameter $\lambda$ and the language $\mathcal{L}$ which parametrizes the SoK and outputs the public parameters $\mathsf{pp}$.

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

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

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

$\underline{(\{\mathfrak{r}_i\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a}) \leftarrow \mathsf{AsSim}(\{\mathsf{stmt}_i\}_{i=1}^{|\mathcal{S}|}, \{m_j\}_{j=1}^{|\mathcal{T}|})}$ takes statements $\mathsf{stmt}_i$ and messages $m_j$ and outputs a set of simulated signatures $\{\mathfrak{r}_i\}_{i=1}^{|\mathcal{S}|}$ and a simulated proof $\mathfrak{a}$.

**Definition 7** (Correctness). *(1) For all* $(\mathsf{stmt}_i, \mathsf{wit}_i) \in \mathbf{R}_{\mathcal{L}}$ *and all messages* $\{m_j \in \{0,1\}^*\}_{j=1}^{|\mathcal{T}|}$ *with* $(\{\mathfrak{r}_i\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a}) \leftarrow \mathsf{AsSign}(\{(\mathsf{stmt}_i, \mathsf{wit}_i)\}_{i=1}^{|\mathcal{S}|}, \{m_j\}_{j=1}^{|\mathcal{T}|})$ *it holds, that* $\mathsf{AsVerify}(\{(\mathfrak{r}_i, \mathsf{stmt}_i)\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a}, \{m_j\}_{j=1}^{|\mathcal{T}|}) = 1$ *(2) Regarding merging with* $t \in \{1,2\}$, *for any* $(\mathsf{stmt}_{t,i}, \mathsf{wit}_{t,i}) \in \mathbf{R}_{\mathcal{L}}$ *and all messages* $M_t := \{m_{t,j} \in \{0,1\}^*\}_{j=1}^{|\mathcal{T}_t|}$ *with* $(\{\mathfrak{r}_{t,i}\}_{i=1}^{|\mathcal{S}_t|}, \mathfrak{a}_t) \leftarrow \mathsf{AsSign}(\{\mathsf{stmt}_{t,i}, \mathsf{wit}_{t,i}\}_{i=1}^{|\mathcal{S}_t|}, M_t)$ *and* $\mathfrak{a} \leftarrow \mathsf{AsMerge}(\mathfrak{a}_1, \mathfrak{a}_2)$ *it holds that* $\mathsf{AsVerify}(\bigcup_{t\in\{1,2\}}\{(\mathfrak{r}_{t,i}, \mathsf{stmt}_{t,i})\}_{i=1}^{|\mathcal{S}_t|}, \mathfrak{a}, \bigcup_{t\in\{1,2\}} M_t) = 1$

The formal security describing that each signature permanently binds a subset of messages is done by reformulating the problem more rigorously. If a SoK $\mathfrak{r}_i$ is reused, then the new set of messages must be a superset of the signed messages, such that all previously signed messages are included.

**Definition 8** (Security). *Given a secure SoK scheme, an 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* $\{m_j \in \{0,1\}^*\}_{j=1}^{|\mathcal{T}|}$ *with* $(\{\mathfrak{r}_i\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a}) \leftarrow \mathsf{AsSign}(\{(\mathsf{stmt}_i, \mathsf{wit}_i)\}_{i=1}^{|\mathcal{S}|}, \{m_j\}_{j=1}^{|\mathcal{T}|})$ *it must hold, that*

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

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.

```
$\text{ASPrivacy}^b_{\mathcal{A}}(1^\lambda)$
  $\mathsf{pp} \leftarrow \mathsf{AsSetup}(1^\lambda)$
  $(I, J, \Sigma, \mathsf{STMT}, \mathcal{M}, \mathfrak{a}) \leftarrow \mathcal{A}(\mathsf{pp})$
  $\Sigma_0 := \Sigma_1 := \Sigma, \mathcal{M}_0 := \mathcal{M}_1 = \mathcal{M}, \mathfrak{a}_0 := \mathfrak{a}_1 := \mathfrak{a}$
  parse $\Sigma$ as $\{\mathfrak{r}_i\}_{i=1}^{|\Sigma|}$ and $\mathsf{STMT}$ as $\{\mathsf{stmt}_i\}_{i=1}^{|\Sigma|}$
  if $\mathsf{AsVerify}(\{(\mathfrak{r}_i, \mathsf{stmt}_i)\}_{i=1}^{|\Sigma|}, \mathfrak{a}, \{m_i\}_{i=1}^{|\mathcal{M}|}) = 0$ then return $0$
  parse $I$ as $\{(\{u^{\mathcal{S}}_{t,i}\}_{t=0}^{1}, \mathsf{stmt}_i, \mathsf{wit}_i)\}_{i=1}^{|I|}$
  parse $J$ as $\{(\{u^{\mathcal{T}}_{t,i}\}_{t=0}^{1}, m_i)\}_{i=1}^{|J|}$
  for all $t \in \{0,1\}$ do
    $\mathfrak{U}^I_t := \{u^{\mathcal{S}}_{t,i}\}_{i=1}^{|I|}, \mathfrak{U}^J_t = \{u^{\mathcal{T}}_{t,j}\}_{j=1}^{|J|}$
    if $\mathfrak{U}^I_t \neq \mathfrak{U}^J_t$ then return $0$
    for all $k \in \mathfrak{U}^I_t$ do
      $\mathsf{STMT}^k_t := \{\mathsf{stmt}_i | u^{\mathcal{S}}_{t,i} = k\}_{i=1}^{|I|}$
      $\mathsf{WIT}^k_t := \{(\mathsf{stmt}_i, \mathsf{wit}_i) | u^{\mathcal{S}}_{t,i} = k\}_{i=1}^{|I|}$
      $\mathcal{M}^k_t := \{m_i | u^{\mathcal{T}}_{t,i} = k\}_{i=1}^{|J|}$
      $(\Sigma^k_t, \mathfrak{a}^k_t) \leftarrow \mathsf{AsSign}(\mathsf{WIT}^k_t, \mathcal{M}^k_t)$
      // % zips: $|A| = |B|$ and $A\%B := \{(a_1, b_1), \ldots, (a_{|A|}, b_{|B|})\}$
      if $\mathsf{AsVerify}(\Sigma^k_t \% \mathsf{STMT}^k_t, \mathfrak{a}^k_t, \mathcal{M}^k_t) = 0$ then return $0$
      $\mathcal{M}_t := \mathcal{M}_t \cup \mathcal{M}^k_t,\ \Sigma_t := \Sigma_t \cup \Sigma^k_t,\ \mathfrak{a}_t \leftarrow \mathsf{Merge}(\mathfrak{a}_t, \mathfrak{a}^k_t)$
  $b' \leftarrow \mathcal{A}(\Sigma_b, \mathfrak{a}_b)$ return $b'$
```

**Fig. 7.** AS privacy experiment

**Definition 9** (Simulatability)**.** AS *is simulatable if*

$$\left\{ x \,\middle|\, \begin{matrix} \mathsf{pp} \leftarrow \mathsf{AsSetup}(1^\lambda, \mathcal{L}) \\ x \leftarrow \mathsf{AsSign}(\{(\mathsf{stmt}_i, \mathsf{wit}_i)\}_{i=1}^{|\mathcal{S}|}, \{m_j\}_{j=1}^{|\mathcal{T}|}) \end{matrix} \right\} = \left\{ x \,\middle|\, \begin{matrix} \mathsf{pp} \leftarrow \mathsf{AsSetup}(1^\lambda, \mathcal{L}) \\ x \leftarrow \mathsf{AsSim}(\{\mathsf{stmt}_i\}_{i=1}^{|\mathcal{S}|}, \{m_j\}_{j=1}^{|\mathcal{T}|}) \end{matrix} \right\}$$

The privacy of the AS scheme is expressed by the security experiment in Figure 7 similarly to the privacy of the SwapCT offer privacy. The mapping of inputs to outputs and the number of participants must remain hidden. An Adversary $\mathcal{A}$ generates a valid set of signatures $\Sigma$, 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 $\mathsf{wit}_i | u^{\mathcal{S}}_{t,i} = k$ and messages $m_i | u^{\mathcal{T}}_{t,i} = k$. $\mathcal{A}$ wins by calculating $b$ correctly.

**Definition 10** (Privacy)**.** *An* AS *scheme is private, if for all PPT adversaries $\mathcal{A}$ it holds that* $\left|\Pr[\text{ASPrivacy}^0_{\mathcal{A}}(1^\lambda) = 1] - \Pr[\text{ASPrivacy}^1_{\mathcal{A}}(1^\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$ *with* $\text{ASPrivacy}^b_{\mathcal{A}}(1^\lambda)$ *defined in Figure 7.*

## 8 SwapCT Construction

With all the components specified in the previous section, the SwapCT construction $\Xi$ is the interaction of the algorithms shown in Figure 8. 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 $\chi$ 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 generates the public parameters of each component by calling their setup functions.

To participate, each entity generates a long term key $(\mathsf{ltp}, \mathsf{lts})$ with KeyGen 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.

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. 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. 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.

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 $\mathsf{ek}, \mathsf{ty}, a, \mathsf{ck}$ are encrypted under the recipient's keys $\mathsf{apk}, \mathsf{vpk}$ to be decrypted with $\mathsf{ask}, \mathsf{vsk}$ recovering the tokens. The structure of the one-time accounts is very similar to the Omniring construction apart from using a typed homomorphic commitment.

The owner of the long term secret key lts is able to receive an account acc by Receive. 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 $a$,

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

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

$\mathsf{TypeGen}(\mathsf{name})$
    $\mathsf{ty} := \mathsf{ComTypeGen}(\mathsf{name})$ return $\mathsf{ty}$

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

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

$\mathsf{VfOffer}(\mathsf{off}, \mathfrak{o})$
    parse $\mathsf{off}$ 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 $\perp$
        $\mathsf{stmt}_i := (\{(\mathsf{pk}_{i,j}^{\mathcal{R}}, \mathsf{com}_{i,j}^{\mathcal{R}})\}_{j=1}^{|\mathcal{R}_i|}, \mathsf{tag}_i, \mathsf{com}'_i)$
    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 $\perp$
    return $\mathsf{AsVerify}_{\mathsf{SoK}[\mathcal{L}^{\mathsf{ring}}]}(\{(\mathfrak{r}_i, \mathsf{stmt}_i)\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a}, \{\mathsf{acc}_j^{\mathcal{T}}\}_{j=1}^{|\mathcal{T}|})$

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

$\mathsf{Receive}(\mathsf{acc}, \mathsf{lts})$
    parse $\mathsf{acc}$ as $(\mathsf{pk}, \mathsf{com}, \tilde{\mathsf{ek}}, \tilde{\mathsf{ck}})$
    parse $\mathsf{lts}$ 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 $(\mathsf{tag}, \mathsf{sk}, a, \mathsf{ty}, \mathsf{ck})$

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

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

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

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

**Fig. 8.** SwapCT Construction

type ty and ck and then derives the tag for this account from the tagging scheme.

With all accounts set up, Offer ensures sender anonymity by creating a temporary commitment $\mathsf{com}'_i$ 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 $\mathsf{com}'_i$ as well as the ring accounts as statement, which is the input for the AsSign 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 (1), and $\mathfrak{o} = (\{\mathfrak{r}_i, \mathsf{com}'_i, \mathsf{ck}'_i\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a})$.

With the underlying aggregatable signature scheme AS, merging offers $\mathfrak{o}_1, \mathfrak{o}_2$ with Merge 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.

Offers are verifiable by VfOffer 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 AsVerify.

Once an offer is balanced and valid, it can be sealed. Seal uses a SoK with the seal language $\mathcal{L}^{\mathsf{seal}}$ 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 intermediate 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′.

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 $(\mathsf{acc}_{\mathsf{reward}}, \mathsf{ck}) \leftarrow \mathsf{OTGen}(\mathsf{ltp}_{\mathrm{miner}}, a_{\mathrm{reward}}, \mathsf{ty}_{\mathrm{native}})$ and $\mathsf{acc}_{\mathsf{reward}}$ 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 $a_{\mathrm{fee}}, r$. A verifier checks the commitment and then appends it to the $\mathcal{L}^{\mathsf{seal}}$ 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 transaction size, this requires one additional input and output for each merged transaction by the exchange.

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

The construction is correct and adheres to the following security properties. The proofs thereof are presented in Appendix B.

**Theorem 1** (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* $\Xi$ *is non-slanderable.*

**Theorem 2** (Theft Prevention). *If* THC *is binding and* AS *is secure and* $\Xi$ *is non-slanderable, it prevents theft.*

**Theorem 3** (Balance). *If* THC *is binding and* $\mathsf{SoK}[\mathcal{L}^{seal}]$ *is extractable and* $\Xi$ *is non-slanderable the construction is balanced.*

**Theorem 4** (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,* $\Xi$ *has offer privacy.*

**Theorem 5** (Transaction Privacy). *If* $\Xi$ *has offer privacy and* $\mathsf{SoK}[\mathcal{L}^{seal}]$ *is simulatable, the construction* $\Xi$ *has transaction privacy.*

```
$\mathsf{AsSign}(\{(\mathsf{stmt}_i, \mathsf{wit}_i)\}_{i=1}^{|\mathcal{S}|}, \{m_j\}_{j=1}^{|\mathcal{T}|})$
  $\vec{s}, \vec{r} \xleftarrow{\$} \mathbb{Z}_q^{|\mathcal{T}|}, C := D := \emptyset$
  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 $(\{\mathfrak{r}_i\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a})$
$\mathsf{AsVerify}(\{(\mathfrak{r}_i, \mathsf{stmt}_i)\}_{i=1}^{|\mathcal{S}|}, \mathfrak{a}, \{m_j\}_{j=1}^{|\mathcal{T}|})$
  parse $\mathfrak{a}$ as $(\{(\pi_i^{\mathcal{S}}, D_i)\}_{i=1}^{|\mathcal{S}|}, \{(\pi_j^{\mathcal{T}}, C_j)\}_{j=1}^{|\mathcal{T}|}, r)$
  $\Gamma := H^{-r}$, $\Delta = I$
  for all $j \in [|\mathcal{T}|]$ do
      if $\mathsf{SoK}[\mathcal{L}^{\mathsf{ped}}]\mathsf{Verify}(\mathsf{stmt} := C_j, \pi_j^{\mathcal{T}}, m_j) = 0$ then return $0$
      $\Gamma := \Gamma C_j G^{\mathfrak{h}(m_j || C_j)}$
  for all $i \in [|\mathcal{S}|]$ do
      if $\mathsf{SoK}[\mathcal{L}^{\mathsf{com}}]\mathsf{Verify}(\mathsf{stmt} := D_i, \pi_i^{\mathcal{S}}, 42) = 0$ then return $0$
      if $\mathsf{SoK}[\mathcal{L}]\mathsf{Verify}(\mathfrak{r}_i, \mathsf{stmt}_i, D_i) = 0$ then return $0$
      $\Delta := \Delta D_i$
  return $b := \Gamma = \Delta$
$\mathsf{AsMerge}(\mathfrak{a}_1, \mathfrak{a}_2)$
  parse $\mathfrak{a}_1$ as $(\{(\pi_{i,1}^{\mathcal{S}}, D_{i,1})\}_{i=1}^{|\mathcal{S}_1|}, \{(\pi_{j,1}^{\mathcal{T}}, C_{j,1})\}_{j=1}^{|\mathcal{T}_1|}, r_1)$
  parse $\mathfrak{a}_2$ as $(\{(\pi_{i,2}^{\mathcal{S}}, D_{i,2})\}_{i=1}^{|\mathcal{S}_2|}, \{(\pi_{j,2}^{\mathcal{T}}, C_{j,2})\}_{j=1}^{|\mathcal{T}_2|}, r_2)$
  $\mathfrak{a} = (\{(\pi_{i,1}^{\mathcal{S}}, D_{i,1})\}_{i=1}^{|\mathcal{S}_1|} \cup \{(\pi_{i,2}^{\mathcal{S}}, D_{i,2})\}_{i=1}^{|\mathcal{S}_2|},$
  $\{(\pi_{j,1}^{\mathcal{T}}, C_{j,1})\}_{j=1}^{|\mathcal{T}_1|} \cup \{(\pi_{j,2}^{\mathcal{T}}, C_{j,2})\}_{j=1}^{|\mathcal{T}_2|}, r_1 + r_2)$ return $\mathfrak{a}$
```

**Fig. 9.** Instantiation of AS

## 9 Component Instantiation

In this section, we provide the instantiation for our aggregatable anonymous signature scheme. For detailed examples of instantiations of the tagging scheme and the labeled public-key encryption scheme, we refer to the Omniring paper [15]. The instantiation of the other components are presented in Appendix A.

The main challenge of non-interactive privacy-preserving swap transactions is to decouple authorization signatures from the signed messages (spending outputs). We, therefore, present a novel aggregatable signature scheme AS which allows the non-interactive merging of offers, as related aggregatable signature schemes [3, 7] are not applicable in our setting.

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

Signing (Figure 9) 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, 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 $r_j \in \mathbb{Z}_q$ is generated 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 $\mathsf{SoK}[\mathcal{L}^{\mathsf{ped}}]$ $\pi_j^{\mathcal{T}}$ 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 $h_j = \mathfrak{h}(m_j \| C_j)$. With the commitment $C_j$ and the correct message, a verifier can calculate $G^{h_j} \cdot C_j = G^{h_j + s_j} \cdot H^{r_j}$ to verify if the messages belong to the signatures.

The signature $\pi_j^{\mathcal{T}}$ is necessary, proving knowledge about the values in the commitment. Without $\pi_j^{\mathcal{T}}$, 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 hiding 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 $\sum_{j=1}^{|\mathcal{T}|}(h_j + s_j)$ and a public sum $r = \sum_{j=1}^{|\mathcal{T}|} r_j$. 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 $\sum_{j=1}^{|\mathcal{T}|}(h_j + s_j)$ stays constant without knowing $s_j$. Original signers can always replace their offer from a merged set and add different offer.

We use this property to distribute the value of $\sum_{j=1}^{|\mathcal{T}|}(h_j + s_j)$ 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 $\sum_{i=1}^{|\mathcal{S}|} x_i = \sum_{j=1}^{|\mathcal{T}|}(h_j + s_j)$. A pragmatic approach is to use $|\mathcal{S}| - 1$ random values and calculate the last as $x_{|\mathcal{S}|} := \sum_{j=1}^{|\mathcal{T}|}(h_j + s_j) - \sum_{i=1}^{|\mathcal{S}|-1} x_i$. To assure the honest creation of these commitments, a valid $\mathsf{SoK}[\mathcal{L}^{\mathsf{com}}]$ $\pi_i^{\mathcal{S}}$ must be attached. An Argument of Knowledge is sufficient here, so we use a constant message. 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 $\prod_{i=1}^{|\mathcal{S}|} D_i = H^{-r} \cdot \prod_{j=1}^{|\mathcal{T}|}(G^{h_j} \cdot C_j)$ 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 $\mathsf{stmt}_i$ and witnesses $\mathsf{wit}_i$ and the messages $D_i$. The aggregatable signature then consists of the commitments $C_j, D_i$ with their $\mathsf{SoK}$ signatures $\pi_j^{\mathcal{T}}, \pi_i^{\mathcal{S}}$, the $\mathsf{SoK}[\mathcal{L}]$ signatures $\mathfrak{r}_i$ and the sum of blinding factors $r$.

Merging 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 $\mathfrak{a}$.

Verification If the publicly calculable product holds, and all SoKs $(\pi_j^{\mathcal{T}}, \pi_i^{\mathcal{S}}, \mathfrak{r}_i)$ are valid, 1 is returned.

Our construction fulfills all required definitions which are proven in Appendix B.

**Theorem 6** (Secure). *The construction for the aggregatable signature is secure according to Definition 8.*

**Theorem 7** (Simulatable). *The* $\mathsf{AS}$ *construction is simulatable according to Definition 9.*

**Theorem 8** (Private). *The construction for* $\mathsf{AS}$ *is private according to Definition 10.*

## 10 Evaluation

The offer and transaction sizes of our SwapCT system are competitive. 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 ($\approx 800\,\text{B}$) which is independent of $r$. Both together with outputs 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:** $+5\tilde{\mathbb{Z}} + (4 + 2\lceil \log_2(2 + m + n + m * n + n * 64) \rceil)\tilde{\mathbb{G}}$
**outputs** $+3\tilde{\mathbb{G}}$

While significantly better than Monero's proofs ($\mathcal{O}(m \cdot r + \log(n))$), we observe our size to be asymptotically linear in $m$ and $n$ while the single type Omniring transaction has no linear components: $\mathcal{O}(\log(r \cdot m + n))$. 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 (10a) 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` [14]. 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

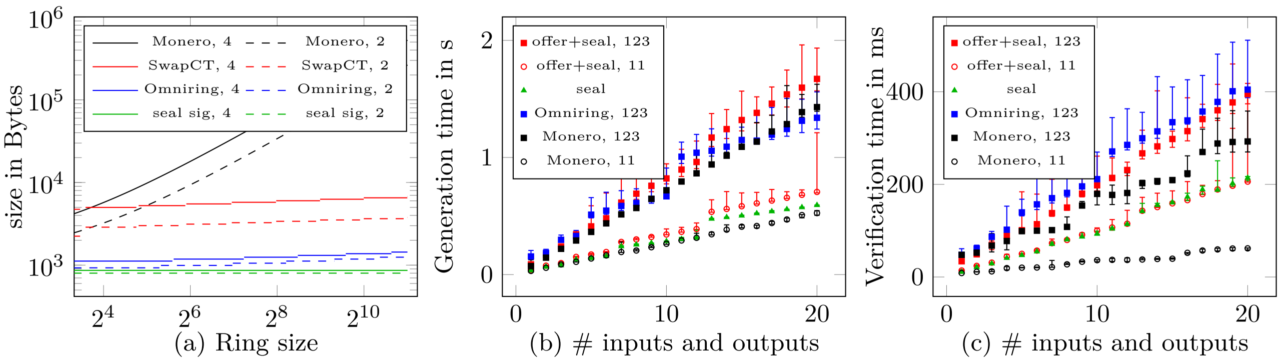

**Fig. 10.** a)Transaction size for 4 and 2 inputs and outputs depending on the ring size. For Omniring, we assumed a common ring size of $r$. Run time for (b) transaction generation with ring size independent sealing time part and (c) transaction verification with a ring independent seal verification 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.

5.8.0-43. Timings for your hardware are easily generated by running our published code at https://github.com/SwapCT/SwapCT. 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 10b). 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 10c). 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.

While our prototype is slightly slower than Monero, it is comparable to a 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. 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.

## 11 Conclusion

With our SwapCT, we present a novel decentralized transaction system which supports both privacy-preserving transactions and non-interactive atomic swaps. We formalize the system and provide an efficient instantiation which offers logarithmically sized transactions for large anonymity sets. For this, we propose our novel aggregatable anonymous signature, a new scheme for non-interactive merging of partial transactions. 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.

## Acknowledgements

We thank the anonymous reviewers and our shepherd, Pedro Moreno-Sanchez, for their helpful comments in improving this work. This work was partially supported by a fellowship within the IFI programme of the German Academic Exchange Service (DAAD) and partially funded by the Carlsberg Foundation under the Semper Ardens Research Project CF18-112 (BCM).

## References

[1] K. M. Alonso and J. Herrera-Joancomartí. Monero - privacy in the blockchain. *IACR Cryptology ePrint Archive*, 2018.

[2] O. Andreev, B. Glickstein, V. Niu, T. Rinearson, D. Sur, and C. Yun. Zkvm: fast, private, flexible blockchain contracts. Technical report, 2019.

[3] A. Bagherzandi and S. Jarecki. Identity-based aggregate and multi-signature schemes based on rsa. In P. Q. Nguyen and D. Pointcheval, editors, *Public Key Cryptography – PKC 2010*. Springer, 2010.

[4] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018.

[5] M. Chase and A. Lysyanskaya. On signatures of knowledge. In *International Cryptology Conference*. Springer, 2006.

[6] U. W. Chohan. The problems of cryptocurrency thefts and exchange shutdowns. *Available at SSRN 3131702*, 2018.

[7] J. M. de Fuentes, L. González-Manzano, J. Tapiador, and P. Peris-Lopez. Pracis: Privacy-preserving and aggregatable cybersecurity information sharing. *Computers & Security*, 2017. Security Data Science and Cyber Threat Mgnt.

[8] A. Deshpande and M. Herlihy. Privacy-preserving cross-chain atomic swaps. In *International Conference on Financial Cryptography and Data Security*. Springer, 2020.

[9] 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*, pages 602–631. Springer, 2020.

[10] S. Eskandari, S. Moosavi, and J. Clark. Sok: Transparent dishonesty: front-running attacks on blockchain. 2019.

[11] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *Advances in Cryptology — CRYPTO' 86*.

[12] 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.

[13] 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. *arXiv preprint arXiv:1909.06535*, 2019.

[14] Isis Agora Lovecruft and Henry de Valence. curve25519_dalek https://doc.dalek.rs/curve25519_dalek/.

[15] 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.

[16] 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. *PoPETs*, 2018.

[17] A. Poelstra, A. Back, M. Friedenbach, G. Maxwell, and P. Wuille. Confidential assets. In *Financial Cryptography Bitcoin Workshop*, 2017.

[18] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *Security and Privacy (SP), 2014 IEEE Symposium on*. IEEE, 2014.

[19] F. Vogelsteller and V. Buterin. Erc-20 token standard. *Ethereum Foundation, Switzerland*, 2015.

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

[21] A. Zamyatin, M. Al-Bassam, D. Zindros, E. Kokoris-Kogias, P. Moreno-Sanchez, A. Kiayias, and W. J. Knottenbelt. Sok: Communication across distributed ledgers. 2019. https://eprint.iacr.org/2019/1128.

[22] Y. Zheng, H. Ye, P. Dai, T. Sun, and V. Gelfer. Confidential assets on mimblewimble. *rin*, 1000:1, 2020.

## A Components

We construct THC like confidential assets which satisfies the following properties.

**Definition 11** (Binding). *A THC scheme is value binding, if for any adversary $\mathcal{A}$ and any $\lambda$, it holds that*

$$\Pr\left[\begin{array}{c|}\mathsf{ty}=\mathsf{ComTypeGen}(i),\mathsf{ty}'=\mathsf{ComTypeGen}(i')\\ \mathsf{Commit}(\mathsf{ty},v;r)=\mathsf{Commit}(\mathsf{ty}',v';r')\\ i\neq i'\vee(i=i'\wedge v\neq v')\end{array}\right.$$
$$\left.\mathsf{pp}\leftarrow\mathsf{ComSetup}(1^\lambda),(i,i',v,v',r,r')\leftarrow\mathcal{A}(\mathsf{pp})\right]<\mathsf{negl}(\lambda)$$

**Definition 12** (Hiding). *A THC scheme is hiding, if for any $\lambda$ and any adversary $\mathcal{A}$ it holds that*

$$\left|\Pr\left[b'\leftarrow\mathcal{A}(\mathsf{pp},\mathsf{com}),b=b'\,\middle|\,\begin{array}{c}\mathsf{pp}\leftarrow\mathsf{ComSetup}(1^\lambda),r\xleftarrow{\$}\mathbb{R},b\xleftarrow{\$}\{0,1\}\\(i_0,i_1,v_0,v_1)\leftarrow\mathcal{A}(\mathsf{pp})\\\mathsf{com}=\mathsf{Commit}(\mathsf{ComTypeGen}(i_b),v_b;r)\end{array}\right]-\frac{1}{2}\right|<\mathsf{negl}(\lambda)$$

Let $\mathcal{G}=(\mathbb{G},q,G)$ be a secure group and $\mathsf{H}:\{0,1\}^*\to\mathbb{G}$ a hash function. The randomness space is $\mathbb{R}:=\mathbb{Z}_q^2$, $\mathbb{M}:=\mathbb{Z}_q$. A type is the hash of its name $\mathsf{ty}=\mathsf{H}(\mathsf{name})$ and thereby $\mathbb{T}:=\mathbb{G}$. To commit, calculate $(\mathsf{ty}\cdot G^r,\mathsf{ty}^a\cdot G^{ra+s}):=\mathsf{Commit}(\mathsf{ty},a;(r,s))$.

**Theorem 9** (Updatable). *For any $i,i'\in\{0,1\}^*$ and $\mathsf{ty}=\mathsf{ComTypeGen}(i),\mathsf{ty}'=\mathsf{ComTypeGen}(i')$ with $(T,V)=\mathsf{Commit}(\mathsf{ty},a;(r,s))$ and $(T',V')=\mathsf{Commit}(\mathsf{ty}',a';(r',s'))$ it holds that $i=i'$, if there exists a PPT algorithm to compute $\phi_1$ in $T\cdot T'^{-1}=G^{\phi_1}$. If additionally $\phi_2$ in $V\cdot V'^{-1}=G^{\phi_2}$ is PPT computable, $\mathsf{ty}=\mathsf{ty}'$ and $a=a'$ holds. Proof in Appendix B.*

The Omniring tagging scheme is used as an example: Let $\mathsf{TagKGen}(x)=H^x$ with $H\in\mathbb{G}$ and $\mathsf{TagEval}(x)=G^{\frac{1}{x}}$. We denote the vector exponentiation as $G^{\circ\vec{v}}:=(G^{v_1},\dots,G^{v_n})$. For a scalar $s$ we define

$\vec{s}^n := (s^0, s^1, \ldots, s^{n-1})$. An ordered set $\{A_i\}_{i=1}^k$ may be interchangeably written as vector $\vec{A} = (A_1, \ldots, A_k)$.

For the languages $\mathcal{L}^{\mathsf{com}}$ and $\mathcal{L}^{\mathsf{ped}}$ we use a standard sigma protocol and transform it to a SoK. As both the Tagged Ring Signature and the Seal Signature are based on a Signature of Knowledge, we first describe a generic efficient SoK based on the Bulletproof and Omniring structure and then parametrize it for each of the purposes. Let again be $\mathcal{G} = (\mathbb{G}, q, G, H)$ a cyclic group $\mathbb{G}$ of prime order $q$ with generators $G$ and $H$ where the discrete log assumption holds. The goal of this signature is to prove the knowledge of the vectors $\vec{c}_L, \vec{c}_R$ which satisfy a set of conditions relative to publicly known group elements $\vec{K}$ and inner product relations. The parameters of the system consist of two witness vectors $\vec{c}_L \in \mathbb{Z}_q^m$ and $\vec{c}_R \in \mathbb{Z}_q^m$ of length $m$. They may be dependent on the challenge variables $u, v$. Each vector is composed of two parts $\vec{c}_L = (\vec{c}_{L,1} \| \vec{c}_{L_2})$ and $\vec{c}_R = (\vec{c}_{R,1} \| \vec{c}_{R_2})$ where $\vec{c}_{L,1}$ and $\vec{c}_{R,1}$ are both of length $n$ with $n \leq m$. The second parameter is a vector of public group elements $\vec{K} \in \mathbb{G}^n$, which may depend on $u, v$ and fulfills the condition that the product of element-wise exponentiation by $\vec{c}_{L,1}$ results in the identity element $I = \prod \vec{K}^{\circ \vec{c}_{L,1}}$. Additionally it accepts constraints in a very specific form. Each constraint $\vec{\mathrm{v}}_i, \vec{\mathrm{v}}'_i \in \mathbb{Z}_q^m$ may be parameterized by $u, v, y$. For each $i$, the structure of the constraint can fall into one of the following four constraints classes, where $c_i$ is efficiently computable by the verifier: $\mathfrak{mul}$: $\langle \vec{c}_L, \vec{c}_R \circ \vec{\mathrm{v}}_i \rangle = c_i$, $\mathfrak{dir}$: $\langle \vec{c}_L, \vec{\mathrm{v}}_i \rangle = c_i$, $\mathfrak{sum}$: $\langle \vec{c}_L, \vec{\mathrm{v}}_i \rangle + \langle \vec{c}_R, \vec{\mathrm{v}}'_i \rangle = c_i$, and $\mathfrak{one}$: $\langle \vec{c}_L - \vec{c}_R - \vec{1}^m, \vec{\mathrm{v}}_i \rangle = c_i$. The prover $\mathcal{P}$ and verifier $\mathcal{V}$ engage in the following interaction:

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

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

Now the prover and the verifier compress the constraints of the parametrization. Each constraint $\vec{\mathrm{v}}_i$ has an index $i$ and $\mathsf{cls}(\vec{\mathrm{v}}_i)$ returns the class $\{\mathfrak{mul}, \mathfrak{dir}, \mathfrak{sum}, \mathfrak{one}\}$ of the constraint. Define:

$$\begin{array}{ll}
\vec{\Theta} := \sum_{i:\mathsf{cls}(\vec{\mathrm{v}}_i)=\mathfrak{mul}} z^i \vec{\mathrm{v}}_i & \vec{\mu} := \sum_{i:\mathsf{cls}(\vec{\mathrm{v}}_i)\neq\mathfrak{mul}} z^i \vec{\mathrm{v}}_i \\
\vec{\nu} := \sum_{i:\mathsf{cls}(\vec{\mathrm{v}}_i)=\mathfrak{one}} z^i \vec{\mathrm{v}}_i & \vec{\omega} := \sum_{i:\vec{\mathrm{v}}'_i \neq 0} z^i \vec{\mathrm{v}}'_i \\
\vec{\alpha} := \vec{\Theta}^{\circ -1} \circ (\vec{\omega} - \vec{\nu}) & \vec{\beta} := \vec{\Theta}^{\circ -1} \circ \mu \\
\delta := \langle \vec{\alpha}, \vec{\mu} \rangle + \langle \vec{1}^m, \vec{\nu} \rangle + \sum_i z^i c_i &
\end{array}$$

$\mathcal{P}$: 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 $\tau_1, \tau_2 \xleftarrow{\$} \mathbb{Z}_q$, $T_1 := G^{t_1} F^{\tau_1}, T_2 := G^{t_2} F^{\tau_2}$
$\mathcal{P} \rightarrow \mathcal{V}$: $T_1, T_2$ and $\mathcal{V}$: $x \xleftarrow{\$} \mathbb{Z}_q, Q \xleftarrow{\$} \mathbb{G}$ and $\mathcal{P} \leftarrow \mathcal{V}$: $x, Q$
$\mathcal{P}$:
1. $\tau := \tau_1 x + \tau_2 x^2$, $r := r_A + r_S x$
2. $(\vec{l}, \vec{r}, t) := (l(x), r(x), t(x))$
3. padd $\vec{l}$ and $\vec{r}$ with 0 to the next power 2 length.
4. $\pi_{\mathsf{IP}} \leftarrow \mathbf{IPprove}(\vec{l}, \vec{r}, \vec{G}_w, \vec{H}^{\vec{\Theta}^{\circ -1}}, Q)$

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

We use the inner product protocol $(\mathbf{IPprove}, \mathbf{IPvf})$ from Bulletproofs which satisfies the language $\mathcal{L}^{\mathsf{IP}} := \left((P, Q, G, H) \in \mathbb{G} : \exists \vec{l}, \vec{r} \in \mathbb{Z}_q^m \text{ s.t. } P = \vec{G}^{\circ \vec{l}} \vec{H}^{\circ \vec{r}} Q^{\langle \vec{l}, \vec{r} \rangle}\right)$

We instantiate the tagged ring signature by specifying the parameters of the previously defined efficient SoK. Using the concrete instantiations for $\mathsf{THC}$ ($\mathsf{com} := (T, V)$) with Theorem 9 and $\mathsf{TAG}$, we get the following language $\mathcal{L}^{\mathsf{ring}}_{\mathsf{THC},\mathsf{TAG}}$

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

Given the challenge variables $u, v$ from the SoK 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 $\mathsf{tag}$, we extend $\vec{K}'$ with $\mathsf{tag}, G, H$. 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

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

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 $y$:

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

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.

The seal signature uses the same efficient generic SoK. Here we describe the preparation of the parameters. Using THC and Theorem 9, we get $\mathcal{L}^{\mathsf{seal}}_{\mathsf{THC}}$

$$:= \begin{cases} \mathsf{stmt} = (\{(T'_i, V'_i)\}_{i=1}^{|\mathcal{S}|}, \{(T_i^{\mathcal{T}}, V_i^{\mathcal{T}})\}_{j=1}^{|\mathcal{T}|}) : \\ \exists \mathsf{wit} = (\{\mathsf{ty}'_i, a'_i, (r'_i, s'_i)\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{ty}_j^{\mathcal{T}}, a_j^{\mathcal{T}}, (r_j^{\mathcal{T}}, s_j^{\mathcal{T}})\}_{j=1}^{|\mathcal{T}|}) : \\ \forall j \in [|\mathcal{T}|] : \begin{cases} \prod T_j^{\mathcal{T}} \cdot \vec{T'}^{\circ - e_{i_j}} = G^{\phi_{1,j}} \\ V_j^{\mathcal{T}} = T_j^{\mathcal{T}\, a_j^{\mathcal{T}}} G^{s_j^{\mathcal{T}}}, a_i^{\mathcal{T}} \in \{0, \dots, 2^\beta - 1\} \end{cases} \\ \prod_{i=1}^{|\mathcal{S}|} V'_i \cdot \prod_{j=1}^{|\mathcal{T}|} V_j^{\mathcal{T}^{-1}} = G^{\phi_2} \end{cases}$$

To compress this into a compact form with the least group elements possible, we require $G$, $\vec{T}_{\mathcal{T}}$, $\vec{T}'$ and one publicly computable element $\hat{V}$ which contribute to $\vec{K}$. The secret exponent $\vec{c}_{L,1}$ of $\vec{K}$ to enforce the constraints and result in the identity, are constructed as follows:

$$\begin{array}{rl} \boldsymbol{K} := & \left(\boldsymbol{G} \| \vec{\boldsymbol{T}}_{\mathcal{S}} \| \vec{\boldsymbol{T}}_{\mathcal{T}}^{\circ u \cdot \vec{v}^{|\mathcal{T}|}} \| \hat{\boldsymbol{V}}\right) \\ \vec{\boldsymbol{c}}_{L,1} := & (\boldsymbol{\xi} \| \;\vec{\hat{\boldsymbol{e}}}\; \| \;\vec{\boldsymbol{a}}_{\mathcal{T}}\; \| \mathbf{1} \| \;\mathbf{v}(\mathbf{E})\; \| \;\mathbf{v}(\mathbf{B})\; ) \\ \vec{\boldsymbol{c}}_{L,1} := & (\mathbf{0} \| \vec{\mathbf{0}}^{|\mathcal{S}|} \| \vec{\mathbf{0}}^{|\mathcal{T}|} \| \mathbf{0} \| \mathbf{v}(\mathbf{E}) - \vec{\mathbf{1}}^{|\mathcal{T}|\cdot|\mathcal{S}|} \| \mathbf{v}(\mathbf{B}) - \vec{\mathbf{1}}^{|\mathcal{T}|\beta}) \end{array}$$

$$\hat{V} = \underbrace{\prod \vec{T}_{\mathcal{T}}^{\circ - \vec{v}^{|\mathcal{T}|}}}_{\text{Surjection}} \cdot \underbrace{\prod \vec{V}_{\mathcal{T}}^{\circ - u \cdot \vec{v}^{|\mathcal{T}|}}}_{\text{Commitments}} \cdot \underbrace{\prod \vec{V}_{\mathcal{S}}^{\circ u^2} \cdot \prod V_{\mathcal{T}}^{\circ - u^2}}_{\text{Equality}}$$

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

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

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

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

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

To enforce the two constraints above on the encoded witness, we again define inner product relations similar to Bulletproofs. They are parameterized by $u, v, y$:

$$\begin{array}{rl} \vec{\mathbf{v}}_0 := & (\mathbf{0} \| \vec{\mathbf{0}}^{|\mathcal{S}|} \| \vec{\mathbf{0}}^{|\mathcal{T}|} \| \mathbf{0} \| \vec{\boldsymbol{y}}^{|\mathcal{T}|\cdot|\mathcal{S}|+|\mathcal{T}|\beta}) \\ \vec{\mathbf{v}}_1 := & (\mathbf{0} \| \vec{\mathbf{0}}^{|\mathcal{S}|} \| \vec{\mathbf{0}}^{|\mathcal{T}|} \| \mathbf{0} \| \vec{\boldsymbol{y}}^{|\mathcal{T}|\cdot|\mathcal{S}|+|\mathcal{T}|\beta}) \\ \vec{\mathbf{v}}_2 := & (\mathbf{0} \| \vec{\mathbf{0}}^{|\mathcal{S}|} \| \vec{\mathbf{0}}^{|\mathcal{T}|} \| \boldsymbol{y}^{|\mathcal{T}|} \| \vec{\boldsymbol{y}}^{|\mathcal{T}|} \otimes \vec{\mathbf{1}}^{|\mathcal{S}|} \| \vec{\mathbf{0}}^{|\mathcal{T}|\beta}) \\ \vec{\mathbf{v}}_3 := & (\mathbf{0} \| \vec{\mathbf{0}}^{|\mathcal{S}|} \| -\vec{\boldsymbol{y}}^{|\mathcal{T}|} \| \mathbf{0} \| \vec{\mathbf{0}}^{|\mathcal{T}|\cdot|\mathcal{S}|} \| \vec{\boldsymbol{y}}^{|\mathcal{T}|} \otimes \vec{\mathbf{2}}^{\beta}) \\ \vec{\mathbf{v}}_4 := & (\mathbf{0} \| -\vec{\boldsymbol{y}}^{|\mathcal{S}|} \| \vec{\mathbf{0}}^{|\mathcal{T}|} \| \mathbf{0} \| \vec{\boldsymbol{v}}^{|\mathcal{T}|} \otimes \vec{\boldsymbol{y}}^{|\mathcal{S}|} \| \vec{\mathbf{0}}^{|\mathcal{T}|\beta}) \end{array}$$

with $\langle \vec{c}_L, \vec{c}_R \circ \vec{\mathrm{v}}_0 \rangle = 0$, $\langle \vec{c}_L - \vec{c}_R - \vec{1}^m, \vec{\mathrm{v}}_1 \rangle = 0$, $\langle \vec{c}_L, \vec{\mathrm{v}}_2 \rangle = \langle \vec{1}^{|\mathcal{T}|+1}, \vec{y}^{|\mathcal{T}|+1} \rangle$, $\langle \vec{c}_L, \vec{\mathrm{v}}_3 \rangle = 0$, and $\langle \vec{c}_L, \vec{\mathrm{v}}_4 \rangle = y$.

**Theorem 10** (SoK Signatures)**.** *Given the parameters above, the resulting protocols are perfectly complete, perfectly special honest-verifier zero-knowledge and logarithmic round arguments of knowledge schemes for $\mathcal{L}^{ring}$ and $\mathcal{L}^{seal}$. Given witness-extended emulation and computationally unique responses, they are transformable to perfectly complete, extractable, perfectly simulatable signatures of knowledge for the languages and any message $m \in \{0,1\}^*$ using Fiat-Shamir [11] which holds for multiple rounds [9, Thm. 23]. As the simulator and an extraction proofs follow the same structure as Omniring proofs we refer the reader to Lai et al. [15].*

## B Construction Security Proofs

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 1 holds in our setting, as our new transaction signature is simulatable by $\mathsf{AsSim}_{\mathcal{L}^{\mathsf{ring}}}$ (Theorem 7) and $\mathsf{SoK}[\mathcal{L}^{\mathsf{seal}}]\mathsf{Sim}$.

*Proof of Theorem 2 (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 $\mathsf{off}' := (\{\mathsf{tag}', \cdot\}_{i=1}^{|\mathcal{S}'|}, \mathcal{R}', \{\mathsf{acc'}_i^{\mathcal{T}}, \cdot\}_{i=1}^{|\mathcal{T}'|}), \mathfrak{o}' := (\{\mathfrak{r}'_i, \cdot\}_{i=1}^{|\mathcal{S}'|}, \mathfrak{a}')$, 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 ($\mathsf{acc}_*^{\mathcal{T}} \notin \{\mathsf{acc'}_i^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}'|}$). As the offer off′ is valid, this implies that $\mathfrak{a}'$ is valid by AsVerify in VfOffer. The security of the AS scheme from Definition 8 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 2 can be used to construct an efficient adversary against the security of $\mathsf{SoK}[\mathcal{L}^{\mathsf{ring}}]$ as the verification in $\mathsf{AsVerify}_{\mathsf{SoK}[\mathcal{L}^{\mathsf{ring}}]}$ 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 1 prevents exactly this. □

*Proof of Theorem 3 (Balance).* To show the balance property, we proceed by constructing an efficient extractor $\mathcal{E}$. As $\mathsf{VfTx}(\mathsf{tx}, \mathfrak{t}) = 1$ implies that $\mathsf{SoK}[\mathcal{L}^{\mathsf{seal}}]\mathsf{Verify}(\mathfrak{t}, \mathsf{stmt}(\mathsf{tx}), \mathsf{tx}) = 1$. Then there exists an efficient extractor $\mathsf{SoK}[\mathcal{L}^{\mathsf{seal}}]\mathcal{E}_{\mathcal{A}}$ extracting a wit for stmt(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 $\forall i \in [|\mathcal{S}|]$ :

$\mathsf{com}'_i = \mathsf{Commit}(\mathsf{ty}^{\mathcal{S}}_i, a^{\mathcal{S}}_i; \mathsf{ck}'_i)$ ,$\forall j \in [|\mathcal{T}|] : \mathsf{com}^{\mathcal{T}}_j = \mathsf{Commit}(\mathsf{ty}^{\mathcal{T}}_j, a^{\mathcal{T}}_j; \mathsf{ck}^{\mathcal{T}}_j)$ ,$\forall \mathsf{ty} \in \{\mathsf{ty}^{\mathcal{T}}_j\}^{|\mathcal{T}|}_{j=1} : \sum\{a^{\mathcal{S}}_i | \mathsf{ty}^{\mathcal{S}}_i = \mathsf{ty}\}^{|\mathcal{S}|}_{i=1} = \sum\{a^{\mathcal{T}}_i | \mathsf{ty}^{\mathcal{T}}_i = \mathsf{ty}\}^{|\mathcal{T}|}_{i=1}$ holds. This directly implies the following conditions of the balance experiment: $\forall i \in [|\mathcal{T}|], \mathsf{ChkAcc}(\mathsf{acc}^{\mathcal{T}}_i, a^{\mathcal{T}}_i, \mathsf{ty}^{\mathcal{T}}_i, \mathsf{ck}^{\mathcal{T}}_i) = 1$ , $\forall i \in [|\mathcal{T}|], \mathsf{ty}^{\mathcal{T}}_i \in \{\mathsf{ty}^{\mathcal{S}}_j\}^{|\mathcal{S}|}_{j=1}$ and $\forall \mathsf{ty} \in \{\mathsf{ty}^{\mathcal{T}}_i\}^{|\mathcal{T}|}_{i=1}$ :
$\sum\{a^{\mathcal{S}}_i | \mathsf{ty}^{\mathcal{S}}_i = \mathsf{ty}\}^{|\mathcal{S}|}_{i=1} = \sum\{a^{\mathcal{T}}_i | \mathsf{ty}^{\mathcal{T}}_i = \mathsf{ty}\}^{|\mathcal{T}|}_{i=1}$. The validity of $\mathsf{VfTx}(\mathsf{tx}, \mathfrak{t}) = 1$ additionaly requires $\mathsf{AsVerify}_{\mathsf{SoK}[\mathcal{L}^{\mathsf{ring}}]}(\{(\mathfrak{r}_i, \mathsf{stmt}_i)\}^{|\mathcal{S}|}_{i=1}, \mathfrak{a}, \{\mathsf{acc}^{\mathcal{T}}_j\}^{|\mathcal{T}|}_{j=1}) = 1$. 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 $\mathsf{wit}_i$ for $\mathsf{stmt}_i$. Parse the statements as $\mathsf{stmt}_i = (\{(\mathsf{pk}_{i,k}, \mathsf{com}_{i,k})\}^{|\mathcal{R}|}_{k=1}, \mathsf{tag}_i, \mathsf{com}'_i)$ and the witnesses as $\mathsf{wit}_i = (j_i, \mathsf{sk}_i, a^{\mathcal{S}}_i, \mathsf{ty}^{\mathcal{S}}_i, \mathsf{ck}^{\mathcal{S}}_i, \mathsf{ck}'_i)$. $\mathcal{L}^{\mathsf{ring}}$ then enforces $\mathsf{pk}_{j_i} = \mathsf{TagKGen}(\mathsf{sk}_i)$, $\mathsf{tag}_i = \mathsf{TagEval}(\mathsf{sk}_i)$, $\mathsf{com}_{i,j_i} = \mathsf{Commit}(\mathsf{ty}_i, a_i; \mathsf{ck}_i)$ and $\mathsf{com}'_i = \mathsf{Commit}(\mathsf{ty}_i, a_i; \mathsf{ck}'_i)$ which implies the remaining conditions of the balance experiment, namely for each $i$, $\mathsf{ChkTag}(\mathsf{acc}^{\mathcal{R}}_{i,j_i}, \mathsf{sk}_i, \mathsf{tag}_i) = 1$ and $\mathsf{ChkAcc}(\mathsf{acc}^{\mathcal{R}}_{i,j_i}, a^{\mathcal{S}}_i, \mathsf{ty}^{\mathcal{S}}_i, \mathsf{ck}^{\mathcal{S}}_i) = 1$ hold. With a binding THC, intermediate commitments $\mathsf{com}'_i$ commit to the same witnesses $(a^{\mathcal{S}}, \mathsf{ty}^{\mathcal{S}})$ in Offer and Seal. □

*Proof of Theorem 4 (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:

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

$\underline{\mathsf{OHyb}_2}$ differs in that the aggregatable signature AS in Offer is simulated by $\mathsf{AsSim}_{\mathcal{L}^{\mathsf{TRS}}}()$. The information available to the Adversary are $\mathsf{off}_0$ which includes $\mathsf{off}_0 = \left(\{\mathsf{tag}_{0,i}, a^{\mathcal{S}}_{0,i}, \mathsf{ty}^{\mathcal{S}}_{0,i}\}^{|\mathcal{S}|}_{i=1}, \{\{\mathsf{acc}^{\mathcal{R}}_{i,j}\}^{|\mathcal{R}_i|}_{j=1}\}^{|\mathcal{S}|}_{i=1}, \{\mathsf{acc}^{\mathcal{T}}_{0,i}, a^{\mathcal{T}}_i, \mathsf{ty}^{\mathcal{T}}_i, \mathsf{ck}^{\mathcal{T}}_{0,i}\}^{|\mathcal{T}|}_{i=1}\right)$ and the intermediate commitment with coin key $\{\mathsf{com}'_{0,i}, \mathsf{ck}'_{0,i}\}^{|\mathcal{S}|}_{i=1}$. Everything else in the signature $\mathfrak{o}_0 = (\{(\mathfrak{r}_{0,i}, \mathsf{com}'_{0,i}, \mathsf{ck}'_{0,i})\}^{|\mathcal{S}|}_{i=1}, \mathfrak{a}_0)$ is simulated. Values indepent of $b$ are ignored, reducing the data to $\left(\{\mathsf{tag}_{0,i}\}^{|\mathcal{S}|}_{i=1}, \{\mathsf{acc}^{\mathcal{T}}_{0,i}, \mathsf{ck}^{\mathcal{T}}_{0,i}\}^{|\mathcal{T}|}_{i=1}\right)$ and $\{\mathsf{com}'_{0,i}, \mathsf{ck}'_{0,i}\}^{|\mathcal{S}|}_{i=1}$

$\underline{\mathsf{OHyb}_3}$ changes the intermediate commitment to $\{\mathsf{com}'_{1,i}, \mathsf{ck}'_{1,i}\}^{|\mathcal{S}|}_{i=1}$, leaving $\left(\{\mathsf{tag}_{0,i}\}^{|\mathcal{S}|}_{i=1}, \{\mathsf{acc}^{\mathcal{T}}_{0,i}, \mathsf{ck}^{\mathcal{T}}_{0,i}\}^{|\mathcal{T}|}_{i=1}\right)$

$\underline{\mathsf{OHyb}_4}$ changes the tags to the experiment with $b = 1$, resulting in $\left(\{\mathsf{tag}_{1,i}\}^{|\mathcal{S}|}_{i=1}, \{\mathsf{acc}^{\mathcal{T}}_{0,i}, \mathsf{ck}^{\mathcal{T}}_{0,i}\}^{|\mathcal{T}|}_{i=1}\right)$ and $\{\mathsf{com}'_{1,i}, \mathsf{ck}'_{1,i}\}^{|\mathcal{S}|}_{i=1}$

$\underline{\mathsf{OHyb}_5}$ : The output accounts consist of a public key and a commitment $\mathsf{acc}^{\mathcal{T}}_{0,i} = (\mathsf{pk}^{\mathcal{T}}_{0,i}, \mathsf{com}^{\mathcal{T}}_{0,i})$. First we change the public keys to $\left(\{\mathsf{tag}_{1,i}\}^{|\mathcal{S}|}_{i=1}, \{\mathsf{pk}^{\mathcal{T}}_{1,i}, \mathsf{com}^{\mathcal{T}}_{0,i}, \mathsf{ck}^{\mathcal{T}}_{0,i}\}^{|\mathcal{T}|}_{i=1}\right)$ and $\{\mathsf{com}'_{1,i}, \mathsf{ck}'_{1,i}\}^{|\mathcal{S}|}_{i=1}$

$\underline{\mathsf{OHyb}_6}$ changes output commitments and coin keys to $\left(\{\mathsf{tag}_{1,i}\}^{|\mathcal{S}|}_{i=1}, \{\mathsf{pk}^{\mathcal{T}}_{1,i}, \mathsf{com}^{\mathcal{T}}_{1,i}, \mathsf{ck}^{\mathcal{T}}_{1,i}\}^{|\mathcal{T}|}_{i=1}\right)$ and $\{\mathsf{com}'_{1,i}, \mathsf{ck}'_{1,i}\}^{|\mathcal{S}|}_{i=1}$

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

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 $\left(\{\mathsf{tag}_{0,i}\}^{|\mathcal{S}|}_{i=1}, \{\mathsf{acc}^{\mathcal{T}}_{0,i}, \mathsf{ck}^{\mathcal{T}}_{0,i}\}^{|\mathcal{T}|}_{i=1}\right)$ and $\{\mathsf{com}'_{1,i}, \mathsf{ck}'_{1,i}\}^{l}_{i=1} \cup \{\mathsf{com}'_{0,i}, \mathsf{ck}'_{0,i}\}^{|\mathcal{S}|}_{i=l+1}$. To show that $\mathsf{OHyb}_{2,l-1} \equiv \mathsf{OHyb}_{2,l}$ we know that the amount $a^{\mathcal{S}}_l$ and type $\mathsf{ty}^{\mathcal{S}}_l$ 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 in $\mathsf{OHyb}_{3,l}$ is $\left(\{\mathsf{tag}_{0,i}\}^{l}_{i=1} \cup \{\mathsf{tag}_{1,i}\}^{|\mathcal{S}|}_{i=l+1}, \{\mathsf{acc}^{\mathcal{T}}_{0,i}, \mathsf{ck}^{\mathcal{T}}_{0,i}\}^{|\mathcal{T}|}_{i=1}\right)$ and $\{\mathsf{com}'_{1,i}, \mathsf{ck}'_{1,i}\}^{|\mathcal{S}|}_{i=1}$. 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 indistinguishable.

$\underline{\mathsf{OHyb}_4 \equiv \mathsf{OHyb}_5}$ : To show the equivalence, we define $|\mathcal{T}| + 1$ 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 $\mathsf{OHyb}_{4,|\mathcal{T}|} = \mathsf{OHyb}_5$. In $\mathsf{OHyb}_{4,l}$ the information available to the adversary is $\left(\{\mathsf{pk}^{\mathcal{T}}_{1,i}, \mathsf{com}^{\mathcal{T}}_{0,i}, \mathsf{ck}^{\mathcal{T}}_{0,i}\}^{l}_{i=1} \cup \{\mathsf{pk}^{\mathcal{T}}_{0,i}, \mathsf{com}^{\mathcal{T}}_{0,i}, \mathsf{ck}^{\mathcal{T}}_{0,i}\}^{|\mathcal{T}|}_{i=l+1}\right)$, $\{\mathsf{tag}_{1,i}\}^{|\mathcal{S}|}_{i=1}$ and $\{\mathsf{com}'_{1,i}, \mathsf{ck}'_{1,i}\}^{|\mathcal{S}|}_{i=1}$. As $\mathsf{pk}^{\mathcal{T}}_{0,l}$ and $\mathsf{pk}^{\mathcal{T}}_{1,l}$ 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 $|\mathcal{T}|+1$ 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.

□

*Proof of Theorem 5 (Transactions).* Similarly to the offer privacy, we prove the transaction privacy with a set of hybrids. First, simulate with $\mathsf{SoK}[\mathcal{L}^{\mathsf{seal}}]\mathsf{Sim}(\mathsf{stmt}(tx), tx)$ in Seal and $\mathsf{AsSim}_{\mathcal{L}^{\mathsf{TRS}}}$ in Offer. Then gradually change the information to the adversary from $\left(\{\mathsf{tag}_{0,i}\}^{|\mathcal{S}|}_{i=1}, \{\mathsf{acc}^{\mathcal{T}}_{0,i}\}^{|\mathcal{T}|}_{i=1}\right)$ and $\{\mathsf{com}'_{0,i}\}^{|\mathcal{S}|}_{i=1}$

to the hybrid with $b = 1$: $\left(\{\mathsf{tag}_{1,i}\}_{i=1}^{|\mathcal{S}|}, \{\mathsf{acc}_{1,i}^{\mathcal{T}}\}_{i=1}^{|\mathcal{T}|}\right)$ and $\{\mathsf{com}'_{1,i}\}_{i=1}^{|\mathcal{S}|}$. Finally change the simulated proofs back to real ones. The hybrids are indistinguishable due to the existence of simulators, hiding commitments and tags as in the previous proof. □

## C Component Security Proofs

*Proof of Theorem 9 (Update).* We show that THC is binding according to Definition 11. Given a discrete log challenge $\mathsf{chl} = (G, G^\gamma) \in \mathbb{G}^2$, we define an oracle $\mathsf{H}^{\mathcal{O}}$ 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 $b = \mathfrak{h}(i\|\mathsf{sk}) \mod 2$ 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 $i, i', v, v', (r, s), (r', s') \leftarrow \mathcal{A}_{\mathsf{H}^{\mathcal{O}}}(\mathsf{pp})$ 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 $\mathsf{H}^{\mathcal{O}}(i)G^r = \mathsf{H}^{\mathcal{O}}(i')G^{r'}$ and $\mathsf{H}^{\mathcal{O}}(i)^v G^{rv+s} = \mathsf{H}^{\mathcal{O}}(i')^{v'} G^{r'v'+s'}$. We have two cases:
<u>For $i = i'$</u> the pre-image is equal, so $v \neq v'$ is true and $v - v' \neq 0$. Then $H^{\mathcal{O}}(i)^{v-v'} = G^{r'v'+s'-rv-s}$. 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 $\mathsf{chl}$.
<u>For $i \neq i'$</u> 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 $\mathsf{H}^{\mathcal{O}}(i) = G^{\mathfrak{h}(i\|\mathsf{sk})}$ and $\mathsf{H}^{\mathcal{O}}(i') = G^{\gamma\mathfrak{h}(i'\|\mathsf{sk})}$. 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 $\gamma$ to solve $\mathsf{chl}$. In both cases, independent of the adversary's choice of $i, i'$, we have $\frac{1}{2} > \mathsf{negl}(\lambda)$ chance to solve the dlog challenge. Therefore we conclude that no efficient adversary against the binding property exists.

We show that from an efficient adversary $\mathcal{A}$ against the update Theorem 9, we can derive an efficient Adversary for the binding property of Def. 11 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 $\phi_2$ 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 6 (Security).* 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 efficiently construct an adversary against the discrete logarithm problem. Given a 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 $\pi_1^{\mathcal{S}} = \mathsf{SoK}[\mathcal{L}^{\mathsf{com}}]\mathsf{Sim}(\mathsf{stmt} = D_1, 42)$. **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 $\mathsf{stmt}_1$ 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 secure 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 $\mathsf{SoK}[\mathcal{L}^{\mathsf{com}}]\mathcal{E}$ which exists due to $\mathsf{SoK}[\mathcal{L}^{\mathsf{com}}]\mathsf{Verify}(\pi'^{\mathcal{S}}_i, D'_i, 42) = 1$ 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 $\gamma = \sum_{j=1}^{|\mathcal{T}|} s'_j - h_1 - \sum_{i=2}^{|\mathcal{S}|} x'_i$. □

*Proof of Theorem 7 (Simulatability).* The witnesses $\mathsf{wit}_i$ are used only in $\mathsf{SoK}[\mathcal{L}]\mathsf{Sign}$, for which an efficient simulator exists. An efficient simulator $\mathsf{AsSim}$ is defined by replacing $\mathfrak{r}_i$ with $\mathfrak{r}_i \leftarrow \mathsf{SoK}[\mathcal{L}]\mathsf{Sim}(\mathsf{stmt}, D_i)$. □

*Proof of Theorem 8 (Privacy).* The information given to the adversary in the experiment about $b$ is $\Sigma_b, \mathfrak{a}_b$. Let $\Sigma_b$ be simulated by $\mathsf{AsSim}$. Sets are closed under the union operation and thereby reveal nothing about $b$. For both $b \in \{0,1\}$ it holds that $\mathfrak{a}_b = (\{\pi_{b,i}^{\mathcal{S}}, D_{b,i}\}_{i=1}^{|I|}, \{\pi_{b,i}^{\mathcal{T}}, C_{b,i}\}_{i=1}^{|J|}, r_b)$. 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 8 holds. □