A Report on Seraphis

coinstudent2048

November 16, 2021

Abstract

This document contains a concise description of Seraphis [11], a novel privacy-preserving transaction protocol abstraction, and its security model. This document will also serve as a suggestion to the organization of the contents of the final Seraphis paper.

1 Introduction

In a p2p (peer-to-peer) electronic cash system, the entire supply of currency exists as a digital record that can be stored by any person, and transactions (attempts to transfer money to new owners) are mediated by a network of *peers* (usually called *nodes*)... ...An unfortunate consequence of cryptocurrencies being decentralized is that the ledger is *public*, implying that all e-notes and transaction events are public knowledge. If amounts are in cleartext, addresses are trivially traceable, and e-notes to be spent are referenced directly, then observers can discern many details about users' finances.

Hence, several confidential transaction protocols have been proposed to ensure financial privacy and currency fungibility. [List privacy technologies and limitations]

1.1 Our contribution

We introduce Seraphis, privacy-preserving transaction protocol abstraction, which means the unlike previous... Moreover...

1.2 Acknowledgments

Lelantus Spark

2 Preliminaries

2.1 Public parameters

Let λ be the security parameter. Let $\mathbb{G}$ be a prime order group based on λ where the Discrete Logarithm (DL) and Decisional Diffie-Hellman (DDH) problems are hard, and let $\mathbb{F}$ be its scalar field. Let G, H, X, U be generators of $\mathbb{G}$ with unknown DL relationship to each other. Note that these generators may be produced using public randomness. Let $a_{max} \in \mathbb{F}$ (to be used in range proofs) and $s \in \mathbb{N}$ (to be used in membership proofs). Let $\mathcal{H} : \{0,1\}^* \to \mathbb{F}$ be a cryptographic hash function. We assume that $\mathcal{H}$ is a random oracle, hence we work in the random oracle model. We add a subscript to $\mathcal{H}$, such as $\mathcal{H}_1$, in lieu of domain-separating the hash function explicitly; any domain-separation method may be used in practice. All these public parameters are collected as pp, and we now define the setup algorithm: $pp \leftarrow \mathsf{Setup}(1^\lambda)$. Setup is implicitly executed by all players involved in the beginning, hence it can be omitted in protocol descriptions.

The notation $\xleftarrow{\$}$ will be used to denote for a uniformly randomly chosen element, and $(1/x)$ for the modular inverse of $x \in \mathbb{F}$. Lastly, we use additive notation for group operations.

2.2 E-notes and e-note images

Definition 2.1. *An* **e-note** *for scalars $k^o_a, k^o_b, a \in \mathbb{F}$ is a tuple (C, K^o, m) such that $C = xG + aH$ for $x \xleftarrow{\$} \mathbb{F}$, $K^o = k^o_a X + k^o_b U$, and m is an arbitrary data.*

C is called the **amount commitment** for the amount a with blinding factor x, K^o is called the **one-time address** for (one-time) private key k^o_a and k^o_b (the o superscript indicates "one-time"), and m is the **memo field**. We say that someone *owns* an e-note if they know the corresponding scalars $k^o_a, k^o_b, a, x \in \mathbb{F}$.

Definition 2.2. *An* **e-note image** *for an e-note (C, K^o, m) is a tuple $(C', K'^o, \tilde{K})$ such that*

$$\begin{aligned} C' &= t_c G + C \\ &= (t_c + x)G + aH \\ &= v_c G + aH \ , \\ K'^o &= t_k G + K^o \\ &= t_k G + k^o_a X + k^o_b U \ , \ \textit{and} \\ \tilde{K} &= (k^o_b / k^o_a) U \end{aligned}$$

for $t_c, t_k \xleftarrow{\$} \mathbb{F}$ and independent to each other.

C' is called the **masked amount commitment**, K'^o is called the **masked address**, and $\tilde{K}$ is called the **linking tag**.

Definition 2.3. *A* **receiver address** *is a tuple (K^{dh}, K^v, K^s) such that $K^{dh} \in \mathbb{G}$, $K^v = k^v K^{dh}$, and $K^s = k^s_a X + k^s_b U$.*

K^{dh} is called the **Diffie-Hellman base public key**, the v superscript indicates "view", and the s superscript indicates "spend". The reason for the name of K^{dh} will be clear in the next subsection, while the reason for the names of superscripts will be discussed in the addressing schemes (Subsection 4.1). We say that someone *owns* a receiver address if they know the corresponding scalars $k^v, k^s_a, k^s_b \in \mathbb{F}$.

2.3 Authenticated symmetric encryption scheme

We require the use of an authenticated symmetric encryption scheme. The Diffie-Hellman base public key enables shared secrets between the sender and the receiver, which can be used to produce the key for encryption and the authentication tag. We denote the encryption and decryption of data x with the input k for Key Derivation Function (KDF) as $\texttt{enc}[k](x)$ and $\texttt{dec}[k](x)$, respectively. We put overlines (e.g. $\overline{x}$) to indicate encrypted data.

The required security properties for application to Seraphis are described in Section 5.2.

3 A Seraphis transaction

We now describe a simple Seraphis transaction. This will be used as the basis for further instantiations and modifications (Section 4) and for the security model (Section 5).

Suppose that Alice would send $a_t \in \mathbb{F}$ amount of funds to Bob. Alice owns a set of e-notes $\{(C_i, K^o_i, m_i)\}^n_{i=1}$ with a total amount of $\left(\sum^n_{i=1} a_i\right) \geq a_t$, all *connected* to a receiver address $(K^{dh}_{ali}, K^v_{ali}, K^s_{ali})$. This "connection" will be elaborated later on. On the other hand, Bob owns a receiver address $(K^{dh}_{bob}, K^v_{bob}, K^s_{bob})$. For Bob to receive the funds, he will now send his receiver address to Alice. Alice will actually send funds to two addresses: to Bob's and to herself (for the "change" $a_c = \sum^n_{i=1} a_i - a_t$ *even if* $a_c = 0$). Hence, Alice must create 2 new e-notes. She starts the transaction by doing the following:

1. Generate $r_{ali}, r_{bob} \xleftarrow{\$} \mathbb{F}$ and independent to each other.
2. Compute $R_{ali} = r_{ali} K^{dh}_{ali}$ and $R_{bob} = r_{bob} K^{dh}_{bob}$, then store R_{ali} and R_{bob} to new (empty) memos m_{ali} and m_{bob}, respectively. The name for K^{dh} should now be clear.

3. Compute the sender-receiver shared secrets $q_c = \mathcal{H}_1(r_{ali} K^v_{ali})$ and $q_t = \mathcal{H}_1(r_{bob} K^v_{bob})$.

4. Compute the one-time addresses $K^o_{ali} = \mathcal{H}_2(q_c)X + K^s_{ali}$ and $K^o_{bob} = \mathcal{H}_2(q_t)X + K^s_{bob}$. The $\mathcal{H}_2(q_c)$ and $\mathcal{H}_2(q_t)$ are uniformly random because of r_{ali}, r_{bob}, and random oracle $\mathcal{H}$.

5. Compute the amount commitments $C_{ali} = \mathcal{H}_3(q_c)G + a_c H$ and $C_{bob} = \mathcal{H}_3(q_t)G + a_t H$. The blinding factors $\mathcal{H}_3(q_c)$ and $\mathcal{H}_3(q_t)$ are uniformly random because of r_{ali}, r_{bob}, and random oracle $\mathcal{H}$.

6. Encrypt the amounts: $\overline{a_c} = \texttt{enc}[q_c](a_c)$ and $\overline{a_t} = \texttt{enc}[q_t](a_t)$, and store $\overline{a_c}$ and $\overline{a_t}$ to memos m_{ali} and m_{bob}, respectively.

Alice now has two new e-notes: $\texttt{enote}_{ali} = (C_{ali}, K^o_{ali}, m_{ali})$ and $\texttt{enote}_{bob} = (C_{bob}, K^o_{bob}, m_{bob})$. These will then be stored to a new (empty) *whole transaction* T. Other objects that will be stored to the whole transaction are from proving systems, which can be executed in *any* order. Proving systems are discussed in the next subsections.

For specific instances of Seraphis, there might be changes in some parts of the above steps, and by reflection, in some parts of the Receipt. Here are some notable changes:

- For some addressing schemes (Subsection 4.1), the input to $\mathcal{H}_2$, the input to $\mathcal{H}_3$, and the key for both `enc` and `dec` may be constructed differently and different to each other. Nevertheless, these inputs and key must be random sender-receiver shared secrets.
- A Seraphis transaction can easily have multiple receivers aside from Bob, which implies that Alice will create more than 2 new e-notes.
- A Seraphis transaction can be collaboratively constructed by multiple players. This is the subject of the so-called "proof dependency", which will be discussed in Subsection 4.3).

3.1 Ownership and unspentness proofs

For each of Alice's owned e-notes $\{(C_i, K^o_i, m_i)\}_{i=1}^n$, Alice must do the following:

1. If the masked address K'^o_i is already in the e-note image $\texttt{enimg}_i$ in T, then go to next step. Else generate K'^o_i from (C_i, K^o_i, m_i) as per definition, and insert it to $\texttt{enimg}_i$ in T.
2. If the linking tag $\tilde{K}_i$ is already in $\texttt{enimg}_i$ in T, then go to next step. Else generate $\tilde{K}_i$ from (C_i, K^o_i, m_i) as per definition, and insert it to $\texttt{enimg}_i$ in T.
3. Prepare the proof transcript $\Pi_{\text{o\&u},i}$ for a non-interactive proving system for the following relation:

$$\{(G, X, U, K'^o_i, \tilde{K}_i \in \mathbb{G}; t_{k,i}, k^o_{a,i}, k^o_{b,i} \in \mathbb{F}) : k^o_{a,i} \neq 0 \wedge K'^o_i = t_{k,i}G + k^o_{a,i}X + k^o_{b,i}U \wedge \tilde{K}_i = (k^o_{b,i}/k^o_{a,i})U\}$$

4. Append $\Pi_{\text{o\&u},i}$ to $(\texttt{enimg}_i, \ldots)$ in T.

Aside from verifying the proof transcript, the Verifier must confirm that the linking tags do not yet appear in the ledger.

The required security properties for application to Seraphis are described in Section 5.1. Unlike what's presented above, we recommend a composition proving system in which instead of one $\Pi_{\text{o\&u},i}$ per i, Alice only needs to produce one proof transcript for all i's. We present this in Appendix A, and provide proof that it satisfies the required security requirements.

3.2 Amount balance

For each of Alice's owned e-notes $\{(C_i, K^o_i, m_i)\}_{i=1}^n$, Alice must do the following:

1. If the masked amount commitment C'_i is already in $\texttt{enimg}_i$ in T, then exit this subsection. Else generate C'_i from (C_i, K^o_i, m_i) as per definition. Then compute the difference:

$$D = d_a G + d_b H = C_{ali} + C_{bob} - \sum_{i=1}^{n} C'_i$$

   Note that d_a is uniformly random because of t_c and random oracle $\mathcal{H}$, while d_b is a publicly known extra amount (e.g. transaction fee).

2. Insert C'_i to $\texttt{enimg}_i$ in T, and store (d_a, d_b) to T.

The generation of $v_{c,n}$ is as such so that the Verifier can verify the amount balance $\sum_{i=1}^{n} C'_i + D = C_{ali} + C_{bob}$.

3.3 Membership proofs

For each of Alice's owned e-notes $\{(C_i, K^o_i, m_i)\}_{i=1}^{n}$, Alice must do the following:

1. If the masked amount commitment C'_i is already in $\texttt{enimg}_i$ in T, then go to next step. Else generate C'_i from (C_i, K^o_i, m_i) exactly like in Step 1 of Subsection 3.2, and insert it to $\texttt{enimg}_i$ in T.

2. If the masked address K'^o_i is already in $\texttt{enimg}_i$ in T, then go to next step. Else generate K'^o_i from (C_i, K^o_i, m_i) as per definition, and insert it to $\texttt{enimg}_i$ in T.

3. Collect $s-1$ number of random e-notes from the ledger and add her owned (C_i, K^o_i, m_i), for a total of s e-notes. The number s is called the **anonymity size**.

4. For each e-note in the collection (of size s), extract only the amount commitment and one-time address like this: (C_j, K^o_j). Then arrange the s e-notes in random positions. Alice now has an array (of length s) of pairs: $\mathbb{S}_i = \{(C_j, K^o_j)\}_{j=1}^{s}$, which is called the **ring**. Its elements (C_j, K^o_j) are called the **ring members**.

5. Prepare the proof transcript $\Pi_{\text{mem},i}$ for a non-interactive proving system for the following relation:

$$\{(G, C'_i, K'^o_i \in \mathbb{G}, \mathbb{S}_i \subset \mathbb{G}^2; \pi_i \in \mathbb{N}, t_{c,i}, t_{k,i} \in \mathbb{F}) : 1 \leq \pi_i \leq s \wedge C'_i - C_{\pi_i} = t_{c,i} G \wedge K'^o_i - K^o_{\pi_i} = t_{k,i} G\}$$

6. Append $(\mathbb{S}_i, \Pi_{\text{mem},i})$ to $(\texttt{enimg}_i, \ldots)$ in T.

The required security properties for application to Seraphis are described in Section 5.1. Specific proving systems satisfying the requirement include CSAG (CLSAG [5] without linking) and One-out-of-Many proving system adapted from Groth and Bootle *et al.* [6, 2].

3.4 Range proofs

For the new e-notes $\texttt{enote}_{ali}$ and $\texttt{enote}_{bob}$, Alice must do the following:

1. Prepare the respective proof transcript $\Pi_{\text{ran},ali}$ and $\Pi_{\text{ran},bob}$ for a non-interactive proving system for the following relation:

$$\{(G, H, C \in \mathbb{G}, a_{max} \in \mathbb{F}; x, a \in \mathbb{F}) : C = xG + aH \wedge 0 \leq a \leq a_{max}\}$$

   where a_{max} is the maximum e-note amount.

2. Store $\Pi_{\text{ran},ali}$ and $\Pi_{\text{ran},bob}$ to T.

The required security properties for application to Seraphis are described in Section 5.1. Specific proving systems satisfying the requirement include Bulletproofs [1] and Bulletproofs+ [3].

3.5 Receipt

Once the construction of T is completed, Alice sends it to the network. Its contents must now be

$$T = (\texttt{enote}_{ali}, \texttt{enote}_{bob}, \Pi_{\text{ran},ali}, \Pi_{\text{ran},bob}, d_a, d_b, \{(\texttt{enimg}_i, \Pi_{\text{o\&u},i}, \mathbb{S}_i, \Pi_{\text{mem},i})\}_{i=1}^{n}).$$

We denote the full construction of T as the function $\texttt{tx}(\cdot)$. This function would be used for describing Seraphis security properties (Section 5.4).

Suppose that the Verifier accepts T, hence T is now stored in the ledger. When Bob scans the ledger for new transactions, he must do the following for every T he encounters:

1. Get a new e-note (C, K^o, m) in T. Note that m contains $(R, \overline{a})$ (see the beginning of this whole section).
2. Compute the nominal sender-receiver shared secret: $q_{nom} = \mathcal{H}_1(k^v_{bob} R)$.
3. Compute the nominal spend public key: $K^s_{nom} = K^o - \mathcal{H}_2(q_{nom})X$. If $K^s_{nom} = K^s_{bob}$, then the e-note is *connected* to Bob's receiver address, and proceed to the next step (this is the "connection" hinted at the beginning of this whole section). Otherwise (if not equal), the e-note is not connected, and hence go to Step 1.
4. Decrypt the amount: $a = \texttt{dec}[q_{nom}](\overline{a})$.
5. Compute the nominal amount commitment: $C_{nom} = \mathcal{H}_3(q_{nom})G + aH$. If $C_{nom} \neq C$, then the e-note is malformed and cannot be spent.
6. Compute the nominal linking tag: $\tilde{K}_{nom} = (k^s_{b,bob}/(k^s_{a,bob} + \mathcal{H}_2(q_{nom})))U$. If he finds a copy of $\tilde{K}_{nom}$ in the ledger, then the e-note has already been spent.

If an e-note (C, K^o, m) is connected to Bob's receiver address, then he knows the corresponding scalars of that e-note: $(k^o_a, k^o_b, a, x) = (k^s_{a,bob} + \mathcal{H}_2(q_{nom}), k^s_{b,bob}, a, \mathcal{H}_3(q_{nom}))$. Hence, "connection" implies e-note ownership. The transaction is complete for Bob.

For Alice to receive the change e-note, she must do the same above steps. After that, the transcation is complete for Alice. This finishes a Seraphis trancation.

4 Instantiations and Modifications

There are a number of details to consider when implementing Seraphis in a real cryptocurrency. Aside from instances of proving systems mentioned already in the previous section, this section is comprised of 'recommendations' for instantiations and modifications of other parts of Seraphis, which are inspired by historical privacy-focused cryptocurrency implementations.

4.1 Addressing schemes

4.2 Multisignature operations

4.3 Proof dependency

Transaction Chaining

4.4 Transaction fees

4.5 Coinbase transactions

4.6 Squashed e-note model

4.7 ?????

5 Security model

For a start, we assume that the distributed ledger is immutable. Therefore, the adversary in our analysis will never be able to modify transactions already stored in the ledger. This ledger immutability can be actualized

through, for instance, the Nakamoto consensus protocol [8].

Subsections 5.1 to 5.3 outline the required security properties of the cryptographic components for Seraphis, and then Subsection 5.4 is the main security analysis of Seraphis.

5.1 Proving systems security properties

We define a proving system as a tuple $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$. Setup is the setup algorithm: $pp \leftarrow \mathsf{Setup}(1^\lambda)$, and $\mathcal{P}$ and $\mathcal{V}$ are PPT algorithms called **Prover** and **Verifier**, respectively. We denote the *transcript* (all data being sent and received in the protocol) produced by $\mathcal{P}$ and $\mathcal{V}$ when dealing with inputs x and y as $tr \leftarrow \langle \mathcal{P}(x), \mathcal{V}(y) \rangle$. Once the transcript is produced, we denote the final transcript verification done by an algorithm $\mathcal{X}$ as $\mathcal{X}(tr) = 1$ for accepted transcript and $\mathcal{X}(tr) = 0$ for rejected.

Let $\mathcal{R}$ be an NP (polynomial-time verifiable) relation of the form $\{(x, w) : P(x, w)\}$ where x is the *statement*, w is the *witness*, and P is a predicate of x and w. Then "$(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ is a proving system for the relation $\mathcal{R}$" informally means that when $\mathcal{P}$ gives an x to $\mathcal{V}$, $\mathcal{P}$ must convince $\mathcal{V}$ that it knows a w such that $(x, w) \in \mathcal{R}$ by generating $tr \leftarrow \langle \mathcal{P}(pp, x, w), \mathcal{V}(pp, x) \rangle$ that $\mathcal{V}$ accepts.

Here are the *minimal* needed security properties of proving systems for Seraphis:

Definition 5.1 (Perfect Completeness)**.** $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *is perfectly complete for* $\mathcal{R}$ *if for all* PPT *adversary* $\mathcal{A}$*,*

$$\mathsf{Pr}\left[\begin{array}{c} (x, w) \in \mathcal{R} \\ \wedge\ \mathcal{V}(tr) = 0 \end{array} \middle| \begin{array}{c} pp \leftarrow \mathsf{Setup}(1^\lambda); (x, w) \leftarrow \mathcal{A}(pp); \\ tr \leftarrow \langle \mathcal{P}(pp, x, w), \mathcal{V}(pp, x) \rangle \end{array} \right] = 0.$$

Definition 5.2 (Computational Soundness)**.** $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *is computationally sound for* $\mathcal{R}$ *if for all* PPT *adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\mathsf{Pr}\left[\begin{array}{c} (x, w) \notin \mathcal{R} \\ \wedge\ \mathcal{V}(tr) = 1 \end{array} \middle| \begin{array}{c} pp \leftarrow \mathsf{Setup}(1^\lambda); (x, w) \leftarrow \mathcal{A}(pp); \\ tr \leftarrow \langle \mathcal{A}(pp, x, w), \mathcal{V}(pp, x) \rangle \end{array} \right] \leq \mathsf{negl}(\lambda).$$

There is another notion of soundness called *Special Soundness*. For a proving system to be special sound, there must exist a *witness extractor* that has an ability to "rewind time" and make the Prover answer several different challenges, and it must be able to extract a witness given the several accepted transcripts with the Prover. Because of this, proving systems satisfying special soundness are called *Proofs of Knowledge*. Special soundness is a stronger notion of soundness, hence this already implies computational soundness.

Definition 5.3 (Perfect Special HVZK)**.** $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *is perfectly special honest-verifier zero knowledge (HVZK) for* $\mathcal{R}$ *if there exists a* PPT *simulator* $\mathcal{S}$ *such that for all* PPT *adversary* $\mathcal{A}$*,*

$$\begin{aligned} &\mathsf{Pr}\left[\begin{array}{c} (x, w) \in \mathcal{R} \\ \wedge\ \mathcal{A}(tr) = 1 \end{array} \middle| \begin{array}{c} pp \leftarrow \mathsf{Setup}(1^\lambda); (x, w, \rho) \leftarrow \mathcal{A}(pp); \\ tr \leftarrow \langle \mathcal{P}(pp, x, w), \mathcal{V}(pp, x; \rho) \rangle \end{array} \right] \\ = &\mathsf{Pr}\left[\begin{array}{c} (x, w) \in \mathcal{R} \\ \wedge\ \mathcal{A}(tr) = 1 \end{array} \middle| \begin{array}{c} pp \leftarrow \mathsf{Setup}(1^\lambda); (x, w, \rho) \leftarrow \mathcal{A}(pp); \\ tr \leftarrow \mathcal{S}(pp, x, \rho) \end{array} \right] \end{aligned}$$

where ρ *is the public randomness used by* $\mathcal{V}$*.*

Fiat-Shamir heuristic [4] is applied to make interactive proving systems non-interactive. Moreover, Fiat-Shamir heuristic transforms interactive protocols satisfying HVZK into non-interactive (fully) zero-knowledge (NIZK) protocols in the random oracle model. Hence, all proving systems for application to Seraphis should be transformed via Fiat-Shamir heuristic.

For membership proofs, we need the following property which is weaker than perfect special HVZK [6]:

Definition 5.4 (Witness Indistinguishability)**.** $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *is witness indistinguishable for* $\mathcal{R}$ *if for all* PPT *adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\left| \frac{1}{2} - \mathsf{Pr}\left[b' = b \middle| \begin{array}{c} pp \leftarrow \mathsf{Setup}(1^\lambda); (x, w_0, w_1) \leftarrow \mathcal{A}(pp); \\ b \xleftarrow{\$} \{0, 1\}; tr \leftarrow \langle \mathcal{P}(pp, x, w_b), \mathcal{V}(pp, x) \rangle; \\ b' \leftarrow \mathcal{A}(tr) \end{array} \right] \right| \leq \mathsf{negl}(\lambda)$$

where $\mathcal{A}(pp)$ *always outputs* (x, w_0, w_1) *such that* $(x, w_0) \in \mathcal{R} \wedge (x, w_1) \in \mathcal{R}$*.*

All proving systems for application to Seraphis must *at least* have perfect completeness and computational soundness. Moreover, ownership and unspentness proofs and range proofs must at least have perfect special HVZK, while membership proofs must at least have witness indistinguishability.

5.2 Authenticated symmetric encryption scheme

We require that the authenticated symmetric encryption scheme must at least have the following properties: indistinguishable against adaptive chosen-ciphertext attack (IND-CCA2) and key-private under chosen-ciphertext attacks (IK-CCA). These properties are defined in the Appendix A.4 of [7].

5.3 Commitment schemes

We define a commitment scheme as a tuple $(\mathsf{Setup}, \mathsf{Comm})$. Setup is the setup algorithm: $pp \leftarrow \mathsf{Setup}(1^\lambda)$, and $\mathsf{Comm} : \mathcal{M} \times \chi \rightarrow \mathcal{C}$ is the *commitment function*, where $\mathcal{M}$ is the message space, χ is the randomness space, and $\mathcal{C}$ is the commitment space. Note that $\mathcal{M}, \chi$ and $\mathcal{C}$ are all constructed from pp. To commit to a message $m \in M$, the sender selects $r \xleftarrow{\$} \chi$ and computes the commitment $C = \mathsf{Comm}(m; r)$. We define the required security properties of commitment schemes.

Definition 5.5 (Hiding Property)**.** *A commitment scheme* $(\mathsf{Setup}, \mathsf{Comm})$ *is computationally hiding if for all* PPT *adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\left| \frac{1}{2} - \mathsf{Pr}\left[b' = b \;\middle|\; \begin{array}{c} pp \leftarrow \mathsf{Setup}(1^\lambda); (m_0, m_1) \leftarrow \mathcal{A}(pp); \\ b \xleftarrow{\$} \{0,1\}; r \xleftarrow{\$} \chi; \\ C = \mathsf{Comm}(m_b; r); b' \leftarrow \mathcal{A}(C) \end{array} \right] \right| \leq \mathsf{negl}(\lambda).$$

A commitment scheme is *perfectly hiding* if $\mathsf{negl}(\lambda)$ is replaced by 0.

Definition 5.6 (Binding Property)**.** *A commitment scheme* $(\mathsf{Setup}, \mathsf{Comm})$ *is computationally binding if for all* PPT *adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\mathsf{Pr}\left[\begin{array}{c} \mathsf{Comm}(m_0; r_0) \\ = \mathsf{Comm}(m_1; r_1) \\ \wedge\ m_0 \neq m_1 \end{array} \;\middle|\; \begin{array}{c} pp \leftarrow \mathsf{Setup}(1^\lambda); \\ (m_0, m_1, r_0, r_1) \leftarrow \mathcal{A}(pp) \end{array} \right] \leq \mathsf{negl}(\lambda).$$

A commitment scheme is *perfectly binding* if $\mathsf{negl}(\lambda)$ is replaced by 0.
The first kind of commitment we define is commonly known as **Pedersen commitments** [9]. We define two instances, $\mathsf{PedersenC} : \mathbb{F} \times \mathbb{F} \rightarrow \mathbb{G}$ and $\mathsf{PedersenK} : \mathbb{F} \times \mathbb{F} \rightarrow \mathbb{G}$ as follows:

$$\mathsf{PedersenC}(a; x) = xG + aH$$
$$\mathsf{PedersenK}(k_b^o; k_a^o) = k_a^o X + k_b^o U$$

$\mathsf{PedersenC}$ corresponds to the structure of amount commitment C and masked amount commitment C', while $\mathsf{PedersenK}$ corresponds to the structure of one-time address K^o.

Theorem 5.1 (From [9])**.** *Pedersen commitment is perfectly hiding and computationally binding under the DL assumption.*

Then we define a custom commitment $\mathsf{LinkTag} : \mathbb{F} \setminus \{0\} \times \mathbb{F} \times \mathbb{F} \rightarrow \mathbb{G} \times \mathbb{G}$ as follows:

$$\mathsf{LinkTag}(k_a^o, k_b^o; t_k) = (t_k G + k_a^o X + k_b^o U, (k_b^o / k_a^o) U)$$

$\mathsf{LinkTag}$ corresponds to structure of the combination of masked address K'^o and linking tag $\tilde{K}$.

Theorem 5.2. $\mathsf{LinkTag}$ *is perfectly hiding and computationally binding under the DL assumption.*

We prove this in Appendix B.

5.4 Seraphis security properties

The required security properties for Seraphis are based on Omniring's security model [7]. The Omniring paper presents a rigorous formalization of RingCT constructions, providing precision for security analysis against several realistic attacks.

It is important to note that the properties in subsection 5.1 are the *minimal* requirements, thus the properties presented in this subsection are the *weakest*. Because of Seraphis's modularity, stronger properties for proving systems must yield stronger properties for Seraphis. Proofs for the theorems in this subsection, found in Appendix B, can also serve a guide for proving stronger properties.

Lastly, note that the mentioned proving systems in the theorems are in their "original" interactive versions. As Fiat-Shamir heuristic produces full zero-knowledge which is stronger than HVZK, the theorems should also apply to the non-interactive protocols transformed through the heuristic.

The first security property is **Completeness** (called *Correctness* in Omniring), which means that if an e-note appears on the ledger, then its owner can honestly generate an accepted transaction spending it. Seraphis satisfying completeness immediately follows from the completeness properties of the cryptographic components and by inspection of the protocol description.

Next we consider the **Balance** property, which means that a spender adversary should never be able to spend more amounts than he truly owns, hence preventing double-spending. The one presented here would be weaker than the one in Omniring because the one in Omniring requires special soundness for proving systems, which we do not require. Balance property involves an experiment $\mathsf{BAL}(\mathcal{A}, 1^\lambda)$ on a PPT adversary $\mathcal{A}$. The adversary succeeds in the experiment (i.e. $\mathsf{BAL}(\mathcal{A}, 1^\lambda) = 1$) if he managed to generate an accepted transaction such that 1) some of spent e-notes are supposedly owned by others 2) some of spent e-note one-time private keys are *not equal* to what is originally given to him, leading to a double-spend of same e-notes under different linking tags, or 3) the amount of the new e-note for the receiver is *larger* than the supposed total amount of e-notes he owns.

$\underline{\mathsf{BAL}(\mathcal{A}, 1^\lambda)}$

- $pp \leftarrow \mathsf{Setup}(1^\lambda)$.
- $\mathcal{A}$ is provided random $k^v, k_a^s, k_b^s \in \mathbb{F}$ to construct the address $\mathtt{addr}_\mathcal{A} = (K_\mathcal{A}^{dh}, K_\mathcal{A}^v, K_\mathcal{A}^s)$, $\{(k_{a,i}^o, k_{b,i}^o, a_i, x_i)\}_{i=1}^n$ that makes $\mathtt{addr}_\mathcal{A}$ connect to $\mathtt{enote}_\mathcal{A} = \{(C_i, K_i^o, m_i)\}_{i=1}^n$ in the ledger, and some other e-notes $\mathtt{enote}_{\neg\mathcal{A}}$ *not* connected to $\mathtt{addr}_\mathcal{A}$.
- $\mathcal{A}$ chooses any receiver address $\mathtt{addr}_\mathcal{B}$.
- $\{(k_{a,i}'^o, k_{b,i}'^o, a_i', x_i')\}_{i=1}^n \leftarrow \mathcal{A}(pp, \mathtt{enote}_\mathcal{A}, \mathtt{enote}_{\neg\mathcal{A}}, \{(k_{a,i}^o, k_{b,i}^o, a_i, x_i)\}_{i=1}^n)$.
- $T \leftarrow \mathtt{tx}(pp, \mathtt{addr}_\mathcal{A}, \mathtt{addr}_\mathcal{B}, \mathtt{enote}_\mathcal{A}, \mathtt{enote}_{\neg\mathcal{A}}, \{(k_{a,i}'^o, k_{b,i}'^o, a_i', x_i')\}_{i=1}^n)$. $\mathcal{A}$ spends n e-notes in $\mathtt{enote}_\mathcal{A} \cup \mathtt{enote}_{\neg\mathcal{A}}$ to send an amount a_t to $\mathtt{addr}_\mathcal{B}$.
- $b_0 := 1$ if Verifier accepts T, else $:= 0$.
- $b_1 := 1$ if some spent e-notes in T are from $\mathtt{enote}_{\neg\mathcal{A}}$, else $:= 0$.
- $b_2 := 1$ if $\exists i \in \{1, \ldots, n\} : k_{b,i}'^o / k_{a,i}'^o \neq k_{b,i}^o / k_{a,i}^o$, else $:= 0$.
- $b_3 := 1$ if $\sum_{i=1}^n a_i < a_t$, else $:= 0$.
- Return $b_0 \wedge (b_1 \vee b_2 \vee b_3)$.

Figure 1: Balance experiment BAL

Definition 5.7 (Balance)**.** *Seraphis is balanced if for all* PPT *adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\Pr\left[\mathsf{BAL}(\mathcal{A}, 1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

where BAL *is described in Figure 1.*

Theorem 5.3 (Balance)**.** *If* PedersenC *and* PedersenK *are both binding, and all proving systems are (computational) sound, then Seraphis is balanced.*

Next we consider the **Privacy** property, which means that an adversary should never be able to detect the spender, receiver, and amounts in any transaction, hence providing sender and receiver anonymity, and confidential transfer of amounts. Privacy property involves an experiment $\mathsf{PRV}(\mathcal{A}, 1^\lambda)$ on a PPT adversary $\mathcal{A}$. In the experiment, it is as if $\mathcal{A}$ himself "sent" amounts to the two potential senders, hence she is provided the sender addresses, the e-notes themselves, and the private scalars of the amount commitment C in those e-notes. Given a whole transaction T, the adversary succeeds in the experiment if she can guess the sender, the receiver, or the amount of the new e-note for the receiver, hence breaking the privacy of T.

$\underline{\mathsf{PRV}(\mathcal{A}, 1^\lambda)}$

- $pp \leftarrow \mathsf{Setup}(1^\lambda)$.
- $\mathcal{A}$ is provided two random potential sender addresses $\mathtt{send}_0$ and $\mathtt{send}_1$, sets of e-notes $\mathtt{enote}_0$ and $\mathtt{enote}_1$ (with $|\mathtt{enote}_0| = |\mathtt{enote}_1| = n$) connected to $\mathtt{send}_0$ and $\mathtt{send}_1$ respectively, private scalars of C, $\{(x_{0,i}, a_{0,i})\}_{i=1}^n$ and $\{(x_{1,i}, a_{1,i})\}_{i=1}^n$, of each e-note in $\mathtt{enote}_0$ and $\mathtt{enote}_1$, respectively, and two random potential receiver addresses $\mathtt{recv}_0$ and $\mathtt{recv}_1$.
- $\mathcal{A}$ constructs $\{\mathbb{S}_i\}_{i=1}^n$ such that each $\mathbb{S}_i$ contains one e-note in $\mathtt{enote}_0$ and one e-note in $\mathtt{enote}_1$.
- $\mathcal{A}$ chooses any valid amount the potential senders would send: $0 \leq a_{\mathcal{A},0} \leq \sum_{i=1}^n a_{0,i}$ and $0 \leq a_{\mathcal{A},1} \leq \sum_{i=1}^n a_{1,i}$ for $\mathtt{send}_0$ and $\mathtt{send}_1$, respectively.
- $b \xleftarrow{\$} \{0, 1\}$.
- $T \leftarrow \mathtt{tx}(pp, \mathtt{send}_b, \mathtt{recv}_b, \mathtt{enote}_b, \{\mathbb{S}_i\}_{i=1}^n, a_{\mathcal{A},b})$. The owner of $\mathtt{send}_b$ honestly spends all e-notes in $\mathtt{enote}_b$ (which are also in $\{\mathbb{S}_i\}_{i=1}^n$) to send the amount $a_{\mathcal{A},b}$ to $\mathtt{recv}_b$.
- If Verifier rejects T, then return 0.
- $b' \leftarrow \mathcal{A}(pp, T, \{(\mathtt{send}_j, \mathtt{recv}_j, \{(x_{j,i}, a_{j,i})\}_{i=1}^n, a_{\mathcal{A},j})\}_{j\in\{0,1\}})$.
- Return 1 if $b = b'$, else 0.

Figure 2: Privacy experiment PRV

Definition 5.8 (Privacy)**.** *Seraphis is private if for all* PPT *adversary* $\mathcal{A}$*, there exist a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\Pr\left[\mathsf{PRV}(\mathcal{A}, 1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

where PRV *is described in Figure 2.*

Theorem 5.4 (Privacy)**.** *If* PedersenC*,* PedersenK*, and* LinkTag *are all hiding, ownership and unspentness proof and range proof are both perfect special HVZK, membership proof is witness indistinguishable, and authenticated symmetric encryption scheme is IND-CCA2 and IK-CCA, then Seraphis is private.*

Lastly, we consider the **Non-slanderability** property, which means that an adversary should never be able to forge a linking tag of an honest user's e-notes when those are spent. Non-slanderability property involves an experiment $\mathsf{NSLAND}(\mathcal{A}, 1^\lambda)$ on a PPT adversary $\mathcal{A}$. This property prevents the following attack known as *denial-of-spending attack* [10]: $\mathcal{A}$ acts as a miner and as a verifier of a victim transaction T. This means that $\mathcal{A}$ can see the linking tags in T. $\mathcal{A}$ then temporarily blocks T from entering the ledger, creates a

new transaction T' that matches some linking tags in T, and enters T' first in the ledger before finally entering T. This way T is marked as a double-spend, and some e-notes of the victim sender are now unspendable. Now the adversary succeeds in the experiment if they successfully accomplished a denial-of-spending attack.

$\underline{\mathsf{NSLAND}(\mathcal{A}, 1^\lambda)}$

- $pp \leftarrow \mathsf{Setup}(1^\lambda)$.
- $\mathcal{A}$ is provided random $k^v, k^s_a, k^s_b \in \mathbb{F}$ to construct the address $\mathtt{addr}_\mathcal{A} = (K^{dh}_\mathcal{A}, K^v_\mathcal{A}, K^s_\mathcal{A})$, and $\mathtt{enote}_\mathcal{A} = \{(C_i, K^o_i, m_i)\}^n_{i=1}$ in the ledger connected to $\mathtt{addr}_\mathcal{A}$.
- $\mathcal{A}$ chooses any receiver address $\mathtt{addr}_\mathcal{B}$.
- Let $\mathtt{addr}_\mathcal{C}$ be the random victim address, $\mathtt{addr}_\mathcal{D}$ be another random address, and $\mathtt{enote}_\mathcal{C}$ be all the e-notes in the ledger connected to $\mathtt{addr}_\mathcal{C}$.
- $T \leftarrow \mathtt{tx}(pp, \mathtt{addr}_\mathcal{C}, \mathtt{addr}_\mathcal{D}, \mathtt{enote}_\mathcal{C})$. The owner of $\mathtt{addr}_\mathcal{C}$ honestly spends all e-notes in $\mathtt{enote}_\mathcal{C}$ to send some amounts to $\mathtt{addr}_\mathcal{D}$. Let $\{\tilde{K}_\mathcal{C}\}$ be all the linking tags in T.
- $\mathcal{A}$ is provided T and verifies it honestly. If $\mathcal{A}$ rejects T, then return 0.
- $\{(k'^o_{a,i}, k'^o_{b,i})\}^n_{i=1} \leftarrow \mathcal{A}(pp, \mathtt{enote}_\mathcal{A}, T)$.
- $T' \leftarrow \mathtt{tx}(pp, \mathtt{addr}_\mathcal{A}, \mathtt{addr}_\mathcal{B}, \mathtt{enote}_\mathcal{A}, \{(k'^o_{a,i}, k'^o_{b,i})\}^n_{i=1})$. $\mathcal{A}$ spends all e-notes in $\mathtt{enote}_\mathcal{A}$ to send some amounts to $\mathtt{addr}_\mathcal{B}$. Let $\{\tilde{K}_\mathcal{A}\}$ be all the linking tags in T'.
- $b_0 := 1$ if Verifier accepts T', else $:= 0$
- $b_1 := 1$ if $\{\tilde{K}_\mathcal{A}\} \cap \{\tilde{K}_\mathcal{C}\} \neq \emptyset$, else $:= 0$
- Return $b_0 \wedge b_1$.

Figure 3: Non-slanderability experiment NSLAND

Definition 5.9 (Non-slanderability)**.** *Seraphis is non-slanderable if for all* PPT *adversary* $\mathcal{A}$*, there exist a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\Pr\left[\mathsf{NSLAND}(\mathcal{A}, 1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

where NSLAND *is described in Figure 3.*

Theorem 5.5 (Non-slanderability)**.** *If* PedersenK *is hiding,* LinkTag *is hiding and binding, ownership and unspentness proof is sound and perfect special HVZK, and membership proof is sound and witness indistinguishable, then Seraphis is non-slanderable.*

Because of the binding property of LinkTag (for e-note images in both T and T'), non-slanderability already captures that an adversary should never be able to forge an accepted transaction of another honest user, a property known as **Unforgeability**. Hence there is no need to formally define and prove an unforgeability property.

6 Efficiency

References

[1] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. Cryptology ePrint Archive, Report 2017/1066, 2017. https://ia.cr/2017/1066.

[2] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. Short accountable ring signatures based on ddh. Cryptology ePrint Archive, Report 2015/643, 2015. https://ia.cr/2015/643.

[3] Heewon Chung, Kyoohyung Han, Chanyang Ju, Myungsun Kim, and Jae Hong Seo. Bulletproofs+: Shorter proofs for privacy-enhanced distributed ledger. Cryptology ePrint Archive, Report 2020/735, 2020. https://ia.cr/2020/735.

[4] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[5] Brandon Goodell, Sarang Noether, and RandomRun. Concise linkable ring signatures and forgery against adversarial keys. Cryptology ePrint Archive, Report 2019/654, 2019. https://ia.cr/2019/654.

[6] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. Cryptology ePrint Archive, Report 2014/764, 2014. https://ia.cr/2014/764.

[7] Russell W. F. Lai, Viktoria Ronge, Tim Ruffing, Dominique Schröder, Sri Aravinda Krishnan Thyagarajan, and Jiafan Wang. Omniring: Scaling up private payments without trusted setup - formal foundations and constructions of ring confidential transactions with log-size proofs. Cryptology ePrint Archive, Report 2019/580, 2019. https://ia.cr/2019/580.

[8] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[9] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, pages 129–140, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[10] Tim Ruffing, Sri Aravinda Thyagarajan, Viktoria Ronge, and Dominique Schröder. Burning zerocoins for fun and for profit: A cryptographic denial-of-spending attack on the zerocoin protocol. Cryptology ePrint Archive, Report 2018/612, 2018. https://ia.cr/2018/612.

[11] UkoeHB. Seraphis: Privacy-focused tx protocol. https://github.com/UkoeHB/Seraphis.

A Composition proving system

The composition proving system is a protocol for the relation:

$$\Big\{\big(G, X, U \in \mathbb{G}, \{K_i\}_{i=1}^n, \{K_{t1,i}\}_{i=1}^n, \{\tilde{K}_i\}_{i=1}^n \in \mathbb{G}^n; \{x_i\}_{i=1}^n, \{y_i\}_{i=1}^n, \{z_i\}_{i=1}^n \in \mathbb{F}^n\big) : \bigwedge_{i=1}^{n} \big(y_i \neq 0 \wedge K_i = x_i G + y_i X + z_i U \wedge K_{t1,i} = (1/y_i)K_i \wedge \tilde{K}_i = (z_i/y_i)U\big)\Big\}$$

Now the Prover only needs to produce one proof transcript for all i's instead of one proof transcript for each i. This protocol is based on the concise approach from [5] to reduce proof sizes when constructing multiple proofs in parallel. Notice the extra $\{K_{t1,i}\}_{i=1}^n$ in the relation. This should not affect the proving system's applicability for ownership and unspentness proof because all y_i's are still hidden in $K_{t1,i}$ (by the DL assumption) and the required relationships for $\{K_i\}_{i=1}^n$ and $\{\tilde{K}_i\}_{i=1}^n$ are still proven.

The protocol proceeds as follows:

1. The Prover generates $\alpha_a, \alpha_b, \alpha_i \xleftarrow{\$} \mathbb{F}$, $\forall i \in \{1, \ldots, n\}$. The Prover computes $(A_a, A_b) = (\alpha_a G, \alpha_b U)$ and $A_i = \alpha_i K_i$, $\forall i \in \{1, \ldots, n\}$, and sends $(A_a, A_b, \{A_i\}_{i=1}^n)$ to the verifier.
2. Both the Prover and Verifier compute $K_{t2,i} = K_{t1,i} - X - \tilde{K}_i$, $\forall i \in \{1, \ldots, n\}$.
3. The Verifier sends a challenge $c \xleftarrow{\$} \mathbb{F}$ and random scalars $\mu_a, \mu_b \xleftarrow{\$} \mathbb{F}$ to the Prover.

4. The Prover computes the responses:

$$r_a = \alpha_a - c\sum_{i=1}^{n} \mu_a^{i-1}(x_i/y_i)$$
$$r_b = \alpha_b - c\sum_{i=1}^{n} \mu_b^{i-1}(z_i/y_i)$$
$$r_i = \alpha_i - c(1/y_i)\ ,\ \forall i \in \{1,\ldots,n\}$$

   and sends those values to the Verifier.

5. The Verifier checks the following equalities. If any of them fail, then the Prover has failed to satisfy the composition proof system.

$$A_a = r_a G + c\sum_{i=1}^{n} \mu_a^{i-1} K_{t2,i}$$
$$A_b = r_b U + c\sum_{i=1}^{n} \mu_b^{i-1} \tilde{K}_i$$
$$A_i = r_i K_i + cK_{t1,i}\ ,\ \forall i \in \{1,\ldots,n\}$$

We now prove that the protocol is perfectly complete, computationally sound, and perfectly special honest-verifier zero knowledge, all if G, X, and U are mutually independent.

Proof. For perfect completeness, note that $\forall i \in \{1,\ldots,n\}$, knowledge of $(x_i/y_i,\ z_i/y_i,\ 1/y_i)$ is also enough to satisfy the proving relation, and

$$\begin{aligned} K_{t2,i} &= K_{t1,i} - X - \tilde{K}_i \\ &= (1/y_i)(x_i G + y_i X + z_i U) - X - (z_i/y_i)U \\ &= (x_i/y_i)G + X + (z_i/y_i)U - X - (z_i/y_i)U \\ &= (x_i/y_i)G. \end{aligned}$$

Then the property follows from inspection.

For perfect special HVZK, we construct a simulator producing accepted transcripts with probability distribution identical to the probability distribution of legitimate accepted transcripts by Prover and Verifier. The simulator generates $c, \mu_a, \mu_b, r_a, r_b, r_i \xleftarrow{\$} \mathbb{F}\ ,\ \forall i \in \{1,\ldots,n\}$. Then the simulator computes $A_a, A_b, \{A_i\}_{i=1}^n$ exactly according to Step 5 of the protocol description. Because of this last step, the Verifier will accept the simulated transcript. Now assume that G, X, and U are mutually independent. Observe that the simulated transcript is uniformly random because $c, \mu_a, \mu_b, r_a, r_b, \{r_i\}_{i=1}^n$ are uniformly randomly selected. On the other hand, observe that a legitimate accepted transcript between Prover and Verifier is also uniformly random because the Prover's $\alpha_a, \alpha_b, \{\alpha_i\}_{i=1}^n$ and the Verifier's c, μ_a, μ_b are all uniformly randomly selected. Hence the two probability distributions are identical.

For computational soundness under the DL assumption, the proof is by contraposition. Suppose that a PPT adversarial prover $\mathcal{A}$ *not* knowing a witness, can produce an accepted transcript

$$(A_a, A_b, \{A_i\}_{i=1}^n, c, \mu_a, \mu_b, r_a, r_b, \{r_i\}_{i=1}^n)$$

with an honest verifier with non-negligible probability. From the third equation in Step 5, $\forall i \in \{1,\ldots,n\}$:

$$\begin{aligned} A_i &= r_i K_i + cK_{t1,i} \\ \implies \alpha_i K_i &= r_i K_i + c(1/y_i)K_i \\ \implies \alpha_i &= r_i + c(1/y_i) \\ \implies (1/y_i) &= (\alpha_i - r_i)(1/c) \end{aligned}$$

Hence solving the discrete log (DL) problem for $K_{t1,i} = (1/y_i)K_i$ with non-negligible probability. Note that α_i and $(1/y_i)$ must always exist because $\mathbb{G}$ is of prime order which implies that any non-zero point is a generator. □

Remark. Since only computational soundness is proven for this protocol, this cannot be called a proof of knowledge. As far as the author's knowledge, constructing a composition proving system that satisfies special soundness is an open problem.

Fiat-Shamir heuristic transforms interactive protocols satisfying HVZK into non-interactive (fully) zero-knowledge (NIZK) protocols in the random oracle model. Applying Fiat-Shamir heuristic to the composition proving system should be straightforward. Note that the c, μ_a and μ_b are generated through domain separation of hash function $\mathcal{H}$.

B Proofs of some theorems in Section 5

We first present another hardness assumption which will be helpful for the next proof. This assumption is used in Bulletproofs [1] and Bulletproofs+ [3]:

Definition B.1 (Discrete Logarithm Relation Assumption)**.** *DL Relation assumption holds relative to* Setup *if for all* $n \geq 2$ *and* PPT *adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\mathsf{Pr}\left[\begin{array}{c|c} \exists i \in \{1,\ldots,n\} : x_i \neq 0 & (\mathbb{G},\mathbb{F}) \leftarrow \mathsf{Setup}(1^\lambda); \\ \wedge \sum_{i=1}^{n} x_i G_i = 0 & \{G_i\}_{i=1}^n \xleftarrow{\$} \mathbb{G}; \\ & \{x_j\}_{j=1}^n \leftarrow \mathcal{A}(\mathbb{G},\mathbb{F},\{G_i\}_{i=1}^n) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

It is said in [1] and [3] that this assumption and the DL assumption are equivalent.

Proof of Theorem 5.2. Perfect hiding follows from the proof of perfect hiding of Pedersen commitments because the blinding factor t_k is only used in one of the pair output: the one corresponding to K'^o.

For computational binding, we prove by contraposition. By breaking the binding of $\mathsf{LinkTag}$, $\mathcal{A}$ finds (k_a^o, k_b^o, t_k) and $(k_a'^o, k_b'^o, t_k')$ such that the two are *not* equal and $\mathsf{LinkTag}(k_a^o, k_b^o; t_k) = \mathsf{LinkTag}(k_a'^o, k_b'^o; t_k')$. This implies that

$$(t_k - t_k')G + (k_a^o - k_a'^o)X + (k_b^o - k_b'^o)U = 0$$

and this breaks the DL relation assumption of $G, X, U \in \mathbb{G}$. □

Proof of Theorem 5.3. We prove by contraposition. Assume that $\mathcal{A}$ succeeds in the BAL experiment with non-negligible probability. There are three cases as to *why* $\mathcal{A}$ succeeded, each with sub-cases: $b_0 \wedge b_1 = 1$, $b_0 \wedge b_2 = 1$, or $b_0 \wedge b_3 = 1$.

For the first case, assume that $b_0 \wedge b_1 = 1$. One sub-case is that $\mathcal{A}$ finds correct private scalars of C and K^o in some e-notes in $\texttt{enote}_{\neg\mathcal{A}}$, but this breaks the binding property of $\mathsf{PedersenC}$ and $\mathsf{PedersenK}$. Another sub-case is that $\mathcal{A}$ doesn't have correct private scalars of C and K^o, yet $\mathcal{A}$ can non-negligibly create accepted proofs, but this breaks the soundness of all proving systems. Hence for all sub-cases, at least one supposed property of a cryptographic construction is broken.

For the second case, assume that $b_0 \wedge b_2 = 1$. One sub-case is that there exists $i \in \{1,\ldots,n\}$ such that $(k_{b,i}^o, k_{a,i}) \neq (k_{b,i}^o, k_{a,i})$. The two must produce unequal linking tags because the (k_b^o, k_a^o) that satisfy both the equations for K^o and $\tilde{K}$ is unique. However, the inequality means that $(k_{b,i}^o, k_{a,i}^o)$ and $(k_{b,i}'^o, k_{a,i}'^o)$ break the binding property of $\mathsf{PedersenK}$. Another sub-case is that $(k_{a,i}'^o, k_{b,i}'^o, t_{k,i}')$ does *not* satisfy proving relations involving those values, yet $\mathcal{A}$ can non-negligibly create accepted proofs, but this breaks the soundness property of ownership and unspentness proof and membership proof. Hence for all sub-cases, at least one supposed property of a cryptographic construction is broken.

For the third case, assume that $b_0 \wedge b_3 = 1$. One sub-case is that the new e-notes are constructed honestly and amount balance $\sum_{i=1}^n C_i' = C_c + C_t - D$ (where C_c is the "change" e-note) is satisfied, but $\sum_{i=1}^n a_i < a_t$. This implies that there exists $i \in \{1,\ldots,n\}$ such that $C_i' = v_{c,i}G + a_i H = v_{c,i}'G + a_i' H$ (where $v_{c,i}' \in \mathbb{F}$ is also found by $\mathcal{A}$) *and* $a_i' > a_i$. Thus $(v_{c,i}, a_i)$ and $(v_{c,i}', a_i')$ breaks the binding property of $\mathsf{PedersenC}$. Another

sub-case is that $\mathcal{A}$ can non-negligibly create an accepted range proof for the amount a committed in new e-note such that $C \neq xG + aH$ or $\neg(0 \leq a \leq a_{max})$, but this breaks the soundness property of range proof. Hence for all sub-cases, at least one supposed property of a cryptographic construction is broken. □

Proof of Theorem 5.4. Assume that $\mathcal{A}$ succeeds in the PRV experiment with non-negligible probability. There are three cases, each with sub-cases: the sender is distinguished, the receiver is distinguished, or the sent amount is distinguished.

For the first case, assume that $\mathcal{A}$ distinguished the sender. One sub-case is that $\mathcal{A}$ determined the $k_b^o = k_b^s$ that K'^o and $\tilde{K}$ hide, but this breaks the hiding property of LinkTag. Another sub-case is that the proving systems leak information related to the sender, like π, the index of true spent e-note in a ring, or k_b^o in e-note images, but this breaks the perfect special HVZK of ownership and unspentness proof and witness indistinguishability of membership proof. Hence for all sub-cases, at least one supposed property of a cryptographic construction is broken.

For the second case, assume that $\mathcal{A}$ distinguished the receiver. One sub-case is that $\mathcal{A}$ found q_t. Now q_t can be found in two ways. One is via C_t of new e-note in $\texttt{recv}_b$, but this breaks the random oracle property of $\mathcal{H}$, DL assumption of $\mathbb{G}$, and requires determining $a_{\mathcal{A},b}$, which will be discussed in the third case. Another way is via determining the encryption key of $\overline{a_t}$, but this breaks the IK-CCA property of authenticated symmetric encryption scheme. From just q_t, $\mathcal{A}$ can get K^s because $K^o - \mathcal{H}_2(q_t)X = K^s$ in new e-note. Now with $\mathcal{H}^{-1}(q_t) = rK^v$, $\mathcal{A}$ may find K^v because of r, but this breaks $\mathcal{H}$ and the randomness of r. Still with $\mathcal{H}^{-1}(q_t)$ and now with $R = rK^{dh}$ in the memo of new e-note, $\mathcal{A}$ may also deduce K^v, but this breaks $\mathcal{H}$ and the DDH assumption of $\mathbb{G}$. Another sub-case is that $\mathcal{A}$ determined that the K^o of new e-note hides k_a^s, k_b^s, but this breaks the hiding property of PedersenK. Hence for all sub-cases, at least one supposed property of a cryptographic construction is broken.

For the third case, assume that $\mathcal{A}$ distinguished the sent amount. One sub-case is that $\mathcal{A}$ determined that the C_t of new e-note hides $a_{\mathcal{A},b}$, but this breaks the hiding property of PedersenC. Another sub-case is that $\mathcal{A}$ recognized in T that $C'_i - t_{c,i}G = x_{b,i}G + a_{b,i}H$ for some $i \in \{1, \ldots, n\}$, but this breaks the randomness of t_c. Another subcase is that the proving systems leak information related to amounts, but this breaks the perfect special HVZK of range proof and witness indistinguishability of membership proof. Moreover, leaking t_c from membership proof leads to the previous case. The last subcase is that $\mathcal{A}$ can non-negligibly decrypt $\overline{a_{\mathcal{A},b}}$ without correct key, but this breaks the IND-CCA2 property of authenticated symmetric encryption scheme. Hence for all sub-cases, at least one supposed property of a cryptographic construction is broken. □

Proof of Theorem 5.5. Assume that $\mathcal{A}$ succeeds in the NSLAND experiment with non-negligible probability. The first step of $\mathcal{A}$ is to get the discrete logarithm k_i of some $\tilde{K}_i \in \{\tilde{K}_\mathcal{C}\}$ (base U), and the second step of $\mathcal{A}$ is to find $(k'^o_{a,i}, k'^o_{b,i}, t'_{k,i})$ (for e-note image in T') such that $k'^o_{b,i}/k'^o_{a,i} = k_i$, making $\tilde{K}_i \in \{\tilde{K}_\mathcal{A}\}$.

For the first step, one case is to simply find k_i by breaking the DL assumption. Another case is to get (k_a^o, k_b^o, t_k) that K'^o_i and $\tilde{K}_i$ hides and simply set $k_i = k_b^o/k_a^o$, but this breaks the hiding property of LinkTag. Another case is to get (k_b^o, k_a^o) that the $K^o_{\pi_i}$ in $\mathbb{S}_i$ (i.e. the one-time address of the true spent e-note) hides, but this breaks the hiding property of PedersenK. The last case is that the proving systems may leak the same (k_a^o, k_b^o, t_k) from the second case above, but this breaks the perfect special HVZK of ownership and unspentness proof and witness indistinguishability of membership proof. Hence for all cases, at least one supposed property of a cryptographic construction is broken.

For the second step, one case is that in T', $K'^o_i = t'_{k,i}G + K^o$, with K^o the one-time address of an e-note in $\texttt{enote}_\mathcal{A}$, but this breaks the binding property of LinkTag. The other case is the negation of the previous case: $K'^o_i \neq t'_{k,i}G + K^o$, yet $\mathcal{A}$ can non-negligibly create accepted proofs, but this breaks the soundness property of ownership and unspentness proof and membership proof. Hence for all cases, at least one supposed property of a cryptographic construction is broken. □