A Report on Seraphis

coinstudent2048

October 1, 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), Computational Diffie-Hellman (CDH), 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 $a_{max} \in \mathbb{F}$ be the maximum amount value. Let $\mathcal{H} : \{0,1\}^* \to \mathbb{F}$ be a cryptographic hash function. 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 $k_a^o, k_b^o, a \in \mathbb{F}$ is a tuple (C, K^o, m) such that $C = xG + aH$ for $x \xleftarrow{\$} \mathbb{F}$, $K^o = k_a^o X + k_b^o U$, 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 keys k_a^o and k_b^o (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 $k_a^o, k_b^o, a, x \in \mathbb{F}$.

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*

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

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 $K^s = k_a^s X + k_b^s U$.*

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 $k^v, k_a^s, k_b^s \in \mathbb{F}$.

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 $\{(C_i, K_i^o, m_i)\}_{i=1}^n$ with a total amount of $\left(\sum_{i=1}^n a_i\right) \geq a_t$, all *connected* to a receiver address $(K_{ali}^{dh}, K_{ali}^v, K_{ali}^s)$. This "connection" will be elaborated later on. On the other hand, Bob owns a receiver address $(K_{bob}^{dh}, K_{bob}^v, K_{bob}^s)$. 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" $a_c = \sum_{i=1}^n a_i - a_t$ *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 $R_{ali} = r_{ali} K_{ali}^{dh}$ and $R_{bob} = r_{bob} K_{bob}^{dh}$, 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 $q_{ali} = \mathcal{H}_1(r_{ali}K^v_{ali})$ and $q_{bob} = \mathcal{H}_1(r_{bob}K^v_{bob})$.

4. Compute the one-time addresses $K^o_{ali} = \mathcal{H}_2(q_{ali})X + K^s_{ali}$ and $K^o_{bob} = \mathcal{H}_2(q_{bob})X + K^s_{bob}$. It is easy to see that $\mathcal{H}_2(q_{ali})$ and $\mathcal{H}_2(q_{ali})$ are uniformly random in the random oracle model.

5. Compute the amount commitments $C_{ali} = \mathcal{H}_3(q_{ali})G + a_cH$ and $C_{bob} = \mathcal{H}_3(q_{bob})G + a_tH$. It is easy to see that the blinding factors $\mathcal{H}_3(q_{ali})$ and $\mathcal{H}_3(q_{bob})$ are uniformly random in the random oracle model.

6. Encrypt the amounts: $\overline{a_c} = \texttt{enc}[q_{ali}](a_c)$ and $\overline{a_t} = \texttt{enc}[q_{ali}](a_t)$, 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, 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 still be 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. See Subsection 5.5 for a discussion of how this affects the security model.

- A Seraphis transaction can be collaboratively constructed by multiple players. This is the subject of the so-called "proof dependency" (Subsection 4.3). Also see Subsection 5.5 for a discussion of how this affects the security model.

3.1 Ownership and unspentness proofs

For each of Alice's owned e-notes in $\{(C_i, K^o_i, m_i)\}^n_{i=1}$, 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 in $\{(C_i, K_i^o, m_i)\}_{i=1}^n$, 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_i^o, m_i) as per definition, *except* for $i = n$. For the case of $i = n$, set

$$v_{c,n} = \mathcal{H}_3(q_{ali}) + \mathcal{H}_3(q_{bob}) - \sum_{i=1}^{n-1} v_{c,i}.$$

   Note that the value of $v_{c,n}$ is still uniformly random because the values of $t_{c,i}$ for $i \in \{1, \ldots, n-1\}$ are uniformly random.

2. Insert C_i' to $\texttt{enimg}_i$ in T.

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

3.3 Membership proofs

For each of Alice's owned e-notes in $\{(C_i, K_i^o, m_i)\}_{i=1}^n$, 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_i^o, 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 $K_i'^o$ is already in $\texttt{enimg}_i$ in T, then go to next step. Else generate $K_i'^o$ from (C_i, K_i^o, 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_i^o, 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_j^o). Then arrange the s e-notes in random positions. Alice now has an array (of length s) of pairs: $\mathbb{S}_i = \{(C_j, K_j^o)\}_{j=1}^s$, which is called the **ring**. Its elements (C_j, K_j^o) are called the **ring members**.

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

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

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, 1].

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:

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

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 [2] and Bulletproofs+ [3].

3.5 Receipt

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

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

Suppose that the Verifier successfully verified 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 [9].

Subsections 5.1 to 5.3 outline the required security properties of the cryptographic components for Seraphis, then Subsection 5.4 is the main security analysis of Seraphis, and Subsection 5.5 discusses how some instantiations and modifications described in Section 4 affect the security analysis.

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* $\mathcal{A}$*,*

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

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*

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

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 (more than 1) accepted transcripts with the Prover. This 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* $\mathcal{A}$*,*

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

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 has the following properties: indistinguishable against chosen-plaintext attack (IND-CPA), indistinguishable against adaptive chosen-ciphertext attack (IND-CCA2), and key-private under chosen-ciphertext attacks (IK-CCA).

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 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 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.

5.4 Seraphis security properties

The required security properties for Seraphis are based on Omniring's security model [8]. 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.

We first define commitment functions $\mathsf{CommAmount} : \mathbb{F} \times \mathbb{F} \rightarrow \mathbb{G}$ and $\mathsf{CommTag} : \mathbb{F} \setminus \{0\} \times \mathbb{F} \times \mathbb{F} \rightarrow \mathbb{G} \times \mathbb{G}$ to be needed later on, as follows:

$$\mathsf{CommAmount}(a; x) = xG + aH \ , \ x \xleftarrow{\$} \mathbb{F} \text{ if } x \text{ is not specified.}$$

$$\mathsf{CommTag}(k_a, k_b; t_k) = (t_k G + k_a X + k_b U, (k_b/k_a)U) \ , \ t_k \xleftarrow{\$} \mathbb{F} \text{ if } t_k \text{ is not specified.}$$

$\mathsf{CommAmount}$ corresponds to the structure of amount commitment C, while $\mathsf{CommTag}$ corresponds to the combination of structures of masked address K'^o and linking tag $\tilde{K}$.

The first security property is **Completeness** (called *Correctness* in Omniring), which means that if an e-note appears on the ledger, then the user owning it 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 private information related to amounts like the discrete log of linking tags, spent e-note private keys, etc. are *not equal* to what is originally given to him, or 2) 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 $k_v, k_a^s, k_b^s \in \mathbb{F}$ to construct the address $\mathtt{addr}_\mathcal{A} = (K_\mathcal{A}^{dh}, K_\mathcal{A}^v, K_\mathcal{A}^s)$, and $\{(k_{a,i}^o, k_{b,i}^o, a_i, x_i)\}_{i=1}^n$ that makes $\mathcal{A}$ the owner of $\mathtt{enote}_\mathcal{A} = \{(C_i, K_i^o, m_i)\}_{i=1}^n$ in the ledger.
- $\mathcal{A}$ can choose any receiver's address $\mathtt{addr}_\mathcal{B} = (K_\mathcal{B}^{dh}, K_\mathcal{B}^v, K_\mathcal{B}^s)$.
- $\{(k_{a,i}'^o, k_{b,i}'^o, a_i', x_i')\}_{i=1}^n \leftarrow \mathcal{A}(pp, \mathtt{enote}_\mathcal{A}, \{(k_{a,i}^o, k_{b,i}^o, a_i, x_i)\}_{i=1}^n)$.
- $T \leftarrow \mathcal{A}(pp, \mathtt{addr}_\mathcal{A}, \mathtt{addr}_\mathcal{B}, \mathtt{enote}_\mathcal{A}, \{(k_{a,i}'^o, k_{b,i}'^o, a_i', x_i')\}_{i=1}^n)$. $\mathcal{A}$ spends $\mathtt{enote}_\mathcal{A}$ to $\mathtt{addr}_\mathcal{B}$.
- $b_0 := 1$ if Verifier accepts all transcripts in T, else $:= 0$.
- $b_1 := 1$ if $\exists i \in \{1, \ldots, n\} : (k_{a,i}'^o, k_{b,i}'^o, a_i', x_i') \neq (k_{a,i}^o, k_{b,i}^o, a_i, x_i)$, else $:= 0$.
- $b_2 := 1$ if $\sum_{i=1}^n a_i < a_\mathcal{B}$ where $a_\mathcal{B}$ is the amount of the new e-note for $\mathtt{addr}_\mathcal{B}$, else $:= 0$.
- Return $b_0 \wedge (b_1 \vee b_2)$.

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 5.1 (Balance)**.** *If both* $(\mathsf{Setup}, \mathsf{CommAmount})$ *and* $(\mathsf{Setup}, \mathsf{CommTag})$ *are binding, 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 $\mathcal{A}$. The adversary succeeds in the experiment if

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.*

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

- $pp \leftarrow \mathsf{Setup}(1^\lambda)$.
- $\mathcal{A}$ is provided $k_v, k_a^s, k_b^s \in \mathbb{F}$ to construct the address $\mathtt{addr}_\mathcal{A} = (K_\mathcal{A}^{dh}, K_\mathcal{A}^v, K_\mathcal{A}^s)$, $\mathtt{enote}_\mathcal{A} = \{(C_i, K_i^o, m_i)\}_{i=1}^n$ in the ledger which are connected to that address, and a receiver's address $\mathtt{addr}_\mathcal{B} = (K_\mathcal{B}^{dh}, K_\mathcal{B}^v, K_\mathcal{B}^s)$.
- $b' \leftarrow \mathcal{A}(pp, T)$
- Return 1 if $b = b'$, else 0.

Figure 2: Privacy experiment PRV

Lastly, we consider the **Non-slanderability**, which means that an adversary should never be able to spend e-notes on behalf of the owner of the e-notes. Non-slanderability property involves an experiment $\mathsf{NSLAND}(\mathcal{A}, 1^\lambda)$ on a PPT adversary $\mathcal{A}$. The adversary succeeds in the experiment if

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

- $pp \leftarrow \mathsf{Setup}(1^\lambda)$.
- $\mathcal{A}$ is provided $k_v, k_a^s, k_b^s \in \mathbb{F}$ to construct the address $\mathtt{addr}_\mathcal{A} = (K_\mathcal{A}^{dh}, K_\mathcal{A}^v, K_\mathcal{A}^s)$, $\mathtt{enote}_\mathcal{A} = \{(C_i, K_i^o, m_i)\}_{i=1}^n$ in the ledger which are connected to that address, and a receiver's address $\mathtt{addr}_\mathcal{B} = (K_\mathcal{B}^{dh}, K_\mathcal{B}^v, K_\mathcal{B}^s)$.
- $b' \leftarrow \mathcal{A}(pp, T)$
- Return $b_0 \wedge b_1 \wedge b2$.

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*

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

where NSLAND *is described in Figure 3.*

5.5 Discussions

6 Efficiency

References

[1] 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.

[2] 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.

[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] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography: Principles and Protocols*. Chapman and Hall/CRC, 2007.

[8] 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`.

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

[10] 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.

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

$$r_a = \alpha_a - c\sum_{j=1}^{n} \mathcal{H}_7(\{K_{t2,i}\}_{i=1}^n, j)(x_j/y_j)$$

$$r_b = \alpha_b - c\sum_{j=1}^{n} \mathcal{H}_8(\{\tilde{K}_i\}_{i=1}^n, j)(z_j/y_j)$$

$$r_i = \alpha_i - c(1/y_i) \ , \ \forall i \in \{1, \dots, n\}$$

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.

$$A_a = r_a G + c\sum_{j=1}^{n} \mathcal{H}_7(\{K_{t2,i}\}_{i=1}^n, j) K_{t2,j}$$
$$A_b = r_b U + c\sum_{j=1}^{n} \mathcal{H}_8(\{\tilde{K}_i\}_{i=1}^n, j) \tilde{K}_j$$
$$A_i = r_i K_i + c K_{t1,i} \ , \ \forall i \in \{1, \ldots, n\}$$

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 and $\mathcal{H}$ is a random oracle.

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 $c, r_a, r_b, r_i \xleftarrow{\$} \mathbb{F}\ ,\ \forall i \in \{1, \ldots, n\}$. 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 and $\mathcal{H}$ is a random oracle. Observe that the simulated transcript is uniformly random because $c, r_a, r_b, \{r_i\}_{i=1}^n$ 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 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

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

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

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

Hence solving the discrete log (DL) problem for $K_{t1,i} = (1/y_i)K_i$ with non-negligible probability. □

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.

B Security proofs for Seraphis

We first present a lemma which is helpful in proofs by reduction.

Definition B.1. *A function $f : \mathbb{N} \to \mathbb{R}$ is* **negligible** *if for all polynomial $p(\cdot)$ there exists an $N \in \mathbb{N}$ such that for all integers $n > N$ it holds that $f(n) < \frac{1}{p(n)}$.*

Definition B.1 is from Katz & Lindell [7]. We prove the following lemma:

Lemma B.1. *If $f : \mathbb{N} \to \mathbb{R}$ is non-negligible, then $g(\cdot) = f(\cdot)^m$ for any $m \in \mathbb{N}$ and $m > 1$ is non-negligible.*

Proof. The non-negligibility of f means that there exists a polynomial $p(\cdot)$ such that for all $N \in \mathbb{N}$, there exists an $n > N$ such that $f(n) \geq \frac{1}{p(n)}$. Let $p_f(\cdot)$ be such polynomial and n_f be such $n > N$. Then setting $p_g(\cdot) = p_f(\cdot)^m$ and $n_g = n_f$ suffices for non-negligibility of g because $f(n_f) \geq \frac{1}{p_f(n_f)} \implies f(n_f)^m \geq \frac{1}{p_f(n_f)^m}$. □

Lemma B.1 justifies the usage of finite number of "breaks" in proofs by reduction. For a start, the probability for breaking a hardness assumption HA is a function of λ. For now, we denote this function as $\mathsf{Pr}[\mathsf{HA}(\lambda)]$. Hence, for $m > 1$, the probability for breaking HA m-times $\mathsf{Pr}[\wedge_{i=1}^{m} \mathsf{HA}_i(\lambda)] \geq \mathsf{Pr}[\mathsf{HA}(\lambda)]^m$. Now Lemma B.1 says that if $\mathsf{Pr}[\mathsf{HA}(\lambda)]$ is non-negligible (or equivalently, for all negligible function $\mathsf{negl}(\lambda)$, $\mathsf{Pr}[\mathsf{HA}(\lambda)] \geq \mathsf{negl}(\lambda)$), then $\mathsf{Pr}[\mathsf{HA}(\lambda)]^m$ must also be non-negligible and hence $\mathsf{Pr}[\wedge_{i=1}^{m} \mathsf{HA}_i(\lambda)]$ is also non-negligible. Nevertheless, the number of breaks should still be documented in proofs.

Now we present theorems that will be directly applied to the proofs of security properties of Seraphis:

Theorem B.2. $(\mathsf{Setup}, \mathsf{CommAmount})$ *and generally, Pedersen commitment, is perfectly hiding, and computationally binding under the DL assumption* [10].