

A Report on Seraphis

coinstudent2048

December 31, 2022

Abstract

This document contains a concise description of koe’s Seraphis [12], a novel privacy-preserving transaction protocol abstraction, and its security model. Note that this does not cover all of [12], but a rather simple variant of it for easier security analysis. Because of this, extensions/modifications presented in [12], with some mentioned here, would not be analyzed here for now.

1 Preliminaries

1.1 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; these may be produced using public randomness (see Definition A.1 for the formalization). Let $a_{max} \in \mathbb{F}$ (to be used in range proofs) and $s \in \mathbb{N}$ (to be used in membership proofs). Let $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{F}$ 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: $pp \leftarrow \text{Setup}(1^\lambda)$. 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.

1.2 E-notes and e-note images

Definition 1.1. 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 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_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 $k_a^o, k_b^o, a, x \in \mathbb{F}$. Basically, e-notes are *exactly* what the (hidden) wallet owners currently own and can currently spend.

Definition 1.2. An **e-note image** for an e-note (C, K^o, m) is a tuple (C', K'^o, \tilde{K}) such that

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

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**. Basically, e-note images are “commitments” to e-notes for wallet owners to communicate that those e-notes are just spent in a transaction.

Definition 1.3. 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 reason for its name will be clear in the next subsection. The v superscript indicates “view”, while the s superscript indicates “spend”. The reason for these names is irrelevant here, but see “Addressing Schemes” in [12], Subsection 4.6. We say that someone *owns* a receiver address if they know the corresponding scalars $k^v, k_a^s, k_b^s \in \mathbb{F}$.

1.3 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 $\text{enc}[k](x)$ and $\text{dec}[k](x)$, respectively. We put overlines (e.g. \bar{x}) to indicate encrypted data.

The required security properties for application to Seraphis are described in Subsection 3.2.

2 A Simple Seraphis transaction

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 $(\sum_{i=1}^n a_i) \geq a_t$, 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 $q_c = \mathcal{H}_1(r_{ali} K_{ali}^v)$ and $q_t = \mathcal{H}_1(r_{bob} K_{bob}^v)$.
4. Compute the one-time addresses $K_{ali}^o = \mathcal{H}_2(q_c)X + K_{ali}^s$ and $K_{bob}^o = \mathcal{H}_2(q_t)X + K_{bob}^s$. 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: $\bar{a}_c = \text{enc}[q_c](a_c)$ and $\bar{a}_t = \text{enc}[q_t](a_t)$, and store \bar{a}_c and \bar{a}_t to memos m_{ali} and m_{bob} , respectively.

Alice now has two new e-notes: $\text{enote}_{ali} = (C_{ali}, K_{ali}^o, m_{ali})$ and $\text{enote}_{bob} = (C_{bob}, K_{bob}^o, m_{bob})$. 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 as a consequence, in reflected 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 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. This technically breaks the privacy property described in Subsection 3.4. However, the same property guarantees that the number of receivers would be the *only* break.
- It may be possible that a Seraphis transaction can be collaboratively constructed by multiple players. This is the subject of the so-called “Modular Transaction Building” in [12], Subsection 4.8. For now, formal description and security analysis of it is not documented here.

2.1 Ownership and unspentness proofs

Ownership and unspentness proof guarantees to verifiers that Alice the sender truly owns her set of e-notes and she doesn't already spent it. For each of Alice's owned e-notes $\{(C_i, K_i^o, m_i)\}_{i=1}^n$, Alice must do the following:

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

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

4. Append $\Pi_{o\&u,i}$ to $(\mathbf{enimg}_i, \dots)$ 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 Subsection 3.1. In [12], Appendix B, a *composition proving system* is introduced, in which instead of one $\Pi_{o\&u,i}$ per i (as presented above), Alice only needs to produce one proof transcript for all i .

2.2 Amount balance

Amount balance guarantees to verifiers that the sum of input amounts always equals the sum of output amounts in every transaction. For each of Alice's owned e-notes $\{(C_i, K_i^o, m_i)\}_{i=1}^n$, Alice must do the following:

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

$$D = d_aG + d_bH = C_{ali} + C_{bob} - \sum_{i=1}^n C_i'$$

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

2. Insert C_i' to \mathbf{enimg}_i in T , and store (d_a, d_b) to T .

The verifier must verify the amount balance $\sum_{i=1}^n C_i' + D = C_{ali} + C_{bob}$.

2.3 Membership proofs

Membership proof guarantees to verifiers that each owned e-note of Alice is in a set of e-notes in the ledger. For each of Alice's owned e-notes $\{(C_i, K_i^o, m_i)\}_{i=1}^n$, Alice must do the following:

1. If the masked amount commitment C_i' is already in \mathbf{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 2.2, and insert it to \mathbf{enimg}_i in T .

2. If the masked address $K_i'^o$ is already in \mathbf{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 \mathbf{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 $(\mathbf{enimg}_i, \dots)$ in T .

Aside from verifying the proof transcript, the verifier must confirm that all the collected e-notes in rings appear in the ledger.

The required security properties for application to Seraphis are described in Subsection 3.1. Specific proving systems satisfying the requirement include CSAG (CLSAG [6] without linking) and One-out-of-Many proving system adapted from Groth and Bootle *et al.* [7, 1].

In [12], Appendix A, the *squashed e-note model* is introduced. This is an alternative to the above proving relation that allows for relatively simpler and more efficient proof structures.

2.4 Range proofs

Range proof guarantees to everyone that the committed amount a lies in a range. For the new e-notes \mathbf{enote}_{ali} and \mathbf{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 Subsection 3.1. Specific proving systems satisfying the requirement include Bulletproofs [2] and Bulletproofs+ [3].

2.5 Receipt

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

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

We denote the full construction of T as the function $\mathbf{tx}(\cdot)$. This function would be used for describing Seraphis security properties (Subsection 3.4).

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, \bar{a}) (see the beginning of this whole section).
2. Compute the nominal sender-receiver shared secret: $q_{nom} = \mathcal{H}_1(k_{bob}^v R)$.

3. Compute the nominal spend public key: $K_{nom}^s = K^o - \mathcal{H}_2(q_{nom})X$. If $K_{nom}^s = K_{bob}^s$, 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: $a = \text{dec}[q_{nom}](\bar{a})$.
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. The balance property described in Subsection 3.4 must prevent Bob from spending it successfully.
6. Compute the nominal linking tag: $\tilde{K}_{nom} = (k_{b,bob}^s / (k_{a,bob}^s + \mathcal{H}_2(q_{nom})))U$. If he finds a copy of \tilde{K}_{nom} in the ledger, then the e-note has already been spent. The balance property described in Subsection 3.4 and the verifier checking that new linking tags do not yet appear in the ledger (see Subsection 2.1) must prevent Bob from spending it successfully.

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: $(k_a^o, k_b^o, a, x) = (k_{a,bob}^s + \mathcal{H}_2(q_{nom}), k_{b,bob}^s, a, \mathcal{H}_3(q_{nom}))$. 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 transaction is complete for Alice. This finishes a Seraphis transaction.

3 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 3.1 to 3.3 outline the required security properties of the cryptographic components for Seraphis, and then Subsection 3.4 is the main security analysis of Seraphis.

3.1 Proving systems security properties

We define a proving system as a tuple $(\text{Setup}, \mathcal{P}, \mathcal{V})$. Setup is the setup algorithm: $pp \leftarrow \text{Setup}(1^\lambda)$, 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 as $tr = 1$ if accepted and $tr = 0$ if 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 " $(\text{Setup}, \mathcal{P}, \mathcal{V})$ 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 3.1 (Perfect Completeness). $(\text{Setup}, \mathcal{P}, \mathcal{V})$ is *perfectly complete* for \mathcal{R} if for all PPT adversary \mathcal{A} ,

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

Definition 3.2 (Computational Soundness). $(\text{Setup}, \mathcal{P}, \mathcal{V})$ is *computationally sound* for \mathcal{R} if for all PPT adversary \mathcal{A} , there exists a negligible function $\text{negl}(\lambda)$ such that

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

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. Special soundness is a stronger notion of soundness, hence this already implies computational soundness.

Definition 3.3 (Witness Indistinguishability). *(Setup, P, V) is witness indistinguishable for R if for all PPT adversary A, there exists a negligible function negl(λ) such that*

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

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

The well-known property stronger than witness indistinguishability is called *Special Honest-Verifier Zero Knowledge (SHVZK)* [7]. For a proving system to be perfect SHVZK, there must exist a *simulator* producing accepted transcripts with probability distribution identical to the probability distribution of legitimate accepted transcripts by the prover and honest verifier. *Fiat-Shamir heuristic* [4] is applied to make interactive protocols non-interactive. Moreover, it transforms interactive protocols satisfying SHVZK into non-interactive (fully) zero-knowledge (NIZK) protocols in the random oracle model.

All proving systems for application to Seraphis must *at least* have perfect completeness, computational soundness, and witness indistinguishability. However, it is better for proving systems to satisfy the stronger properties special soundness and SHVZK.

3.2 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 [8], Appendix A.4.

3.3 Commitment schemes

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

Definition 3.4 (Hiding Property). *A commitment scheme (Setup, Comm) is computationally hiding if for all PPT adversary A, there exists a negligible function negl(λ) such that*

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

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

Definition 3.5 (Binding Property). *A commitment scheme (Setup, Comm) is computationally binding if for all PPT adversary A, there exists a negligible function negl(λ) such that*

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

A commitment scheme is *perfectly binding* if $\text{negl}(\lambda)$ is replaced by 0. The first kind of commitment we define is commonly known as **Pedersen commitments** [10]. We define two instances, $\text{PedersenC} : \mathbb{F} \times \mathbb{F} \rightarrow \mathbb{G}$ and $\text{PedersenK} : \mathbb{F} \times \mathbb{F} \rightarrow \mathbb{G}$ as follows:

$$\begin{aligned}\text{PedersenC}(a; x) &= xG + aH \\ \text{PedersenK}(k_b^o; k_a^o) &= k_a^o X + k_b^o U\end{aligned}$$

PedersenC corresponds to the structure of amount commitment C and masked amount commitment C' , while PedersenK corresponds to the structure of one-time address K^o . Both are for rings.

Theorem 3.1 (From [10]). *Pedersen commitment is perfectly hiding and computationally binding under the DL assumption.*

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

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

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

Theorem 3.2. *LinkTag is perfectly hiding and computationally binding under the DL assumption.*

We prove this in the Appendix A.

3.4 Seraphis security properties

The required security properties for Seraphis are loosely based on Omniring’s security model [8], with modifications to fit Seraphis. The Omniring paper presents a rigorous treatment of RingCT constructions, providing precision for security analysis against several realistic attacks. However, we will not be fully formal in this presentation: we may treat algorithms such as \mathcal{A} like *agents*. For instance, we may simply write “ \mathcal{A} outputs x_1 and x_2 ” instead of “we run a subroutine in \mathcal{A} twice to get outputs x_1 and x_2 ”.

It is important to note that the properties in Subsection 3.1 are the *minimal* requirements, thus the properties presented in this subsection may be weaker than in [8]. Proofs for these properties are found in the Appendix A. Because of Seraphis’s modularity, stronger properties for proving systems must yield stronger properties for Seraphis.

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 it truly owns, hence preventing double-spending. Balance property involves an experiment $\text{BAL}(\mathcal{A}, 1^\lambda)$ on a PPT adversary \mathcal{A} . The adversary succeeds in the experiment (i.e. $\text{BAL}(\mathcal{A}, 1^\lambda) = 1$) if it managed to generate an accepted transaction such that 1) some of “spent e-notes” are just made up and not in the ledger, 2) all spent e-notes are in the ledger, but some are supposedly owned by others, 3) 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 4) the amount of the new e-note for the receiver is *larger* than the supposed total amount of e-notes it owns.

Definition 3.6 (Balance). *Seraphis is balanced if for all PPT adversary \mathcal{A} , there exists a negligible function $\text{negl}(\lambda)$ such that*

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

where BAL is described in Figure 1.

Theorem 3.3 (Balance). *If PedersenC, PedersenK, and LinkTag are all binding, and all proving systems are (computational) sound, then Seraphis is balanced.*

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

- $pp \leftarrow \text{Setup}(1^\lambda)$.
- \mathcal{A} is provided random $k^v, k_a^s, k_b^s \in \mathbb{F}$ to construct the address $\text{addr}_{\mathcal{A}} = (K_{\mathcal{A}}^{dh}, K_{\mathcal{A}}^v, K_{\mathcal{A}}^s)$, $\{(k_{a,i}^o, k_{b,i}^o, a_i, x_i)\}_{i=1}^n$ that makes $\text{addr}_{\mathcal{A}}$ connect to $\text{enote}_{\mathcal{A}} = \{(C_i, K_i^o, m_i)\}_{i=1}^n$ in the ledger, and *all* other e-notes $\text{enote}_{\neg \mathcal{A}}$ in the ledger *not* connected to $\text{addr}_{\mathcal{A}}$.
- \mathcal{A} chooses any receiver address $\text{addr}_{\mathcal{B}}$ (which can be $\text{addr}_{\mathcal{A}}$).
- $\{(k_{a,i}'^o, k_{b,i}'^o, a_i', x_i')\}_{i=1}^n \leftarrow \mathcal{A}(pp, \text{enote}_{\mathcal{A}}, \text{enote}_{\neg \mathcal{A}}, \{(k_{a,i}^o, k_{b,i}^o, a_i, x_i)\}_{i=1}^n)$.
- $T \leftarrow \text{tx}(pp, \text{addr}_{\mathcal{A}}, \text{addr}_{\mathcal{B}}, \text{enote}_{\mathcal{A}}, \text{enote}_{\neg \mathcal{A}}, \{(k_{a,i}'^o, k_{b,i}'^o, a_i', x_i')\}_{i=1}^n)$. \mathcal{A} spends all the n e-notes to send the amount $a_t = \sum_{i=1}^n a_i'$ to $\text{addr}_{\mathcal{B}}$.
- $b_0 := 1$ if Verifier accepts T , else $:= 0$.
- $b_1 := 1$ if some “spent e-notes” in T are not in ledger, else $:= 0$.
- $b_2 := 1$ if some spent e-notes in T are from $\text{enote}_{\neg \mathcal{A}}$, else $:= 0$.
- $b_3 := 1$ if $\exists i \in \{1, \dots, n\} : k_{b,i}'^o/k_{a,i}'^o \neq k_{b,i}^o/k_{a,i}^o$, else $:= 0$.
- $b_4 := 1$ if $\sum_{i=1}^n a_i < a_t$, else $:= 0$.
- Return $b_0 \wedge (b_1 \vee b_2 \vee b_3 \vee b_4)$.

Figure 1: Balance experiment BAL

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 $\text{PRV}(\mathcal{A}, 1^\lambda)$ on a PPT adversary \mathcal{A} . In the experiment, it is as if \mathcal{A} itself “sent” amounts to the two potential senders, hence \mathcal{A} 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 it can guess the sender, the receiver, or the amount of the new e-note for the receiver, hence breaking the privacy of T .

Definition 3.7 (Privacy). *Seraphis is private if for all PPT adversary \mathcal{A} , there exist a negligible function $\text{negl}(\lambda)$ such that*

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

where PRV is described in Figure 2.

Theorem 3.4 (Privacy). *If PedersenC, PedersenK, and LinkTag are all hiding, all proving systems are 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 $\text{NSLAND}(\mathcal{A}, 1^\lambda)$ on a PPT adversary \mathcal{A} . This property prevents the following attack known as *denial-of-spending attack* [11]: the adversary is in a remote node that can receive transactions, and also acts 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 it successfully accomplished a denial-of-spending attack.

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

- $pp \leftarrow \text{Setup}(1^\lambda)$.
- \mathcal{A} is provided two random potential sender addresses send_0 and send_1 , sets of e-notes enote_0 and enote_1 (with $|\text{enote}_0| = |\text{enote}_1| = n$) connected to send_0 and 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 enote_0 and enote_1 , respectively, and two random potential receiver addresses recv_0 and recv_1 .
- \mathcal{A} constructs $\{\mathbb{S}_i\}_{i=1}^n$ such that each \mathbb{S}_i contains one e-note in enote_0 and one e-note in 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 send_0 and send_1 , respectively.
- $b \xleftarrow{\$} \{0, 1\}$.
- $T \leftarrow \text{tx}(pp, \text{send}_b, \text{recv}_b, \text{enote}_b, \{\mathbb{S}_i\}_{i=1}^n, a_{\mathcal{A},b})$. The owner of send_b honestly spends all e-notes in enote_b (which are also in $\{\mathbb{S}_i\}_{i=1}^n$) to send the amount $a_{\mathcal{A},b}$ to recv_b .
- If Verifier rejects T , then return 0.
- $b' \leftarrow \mathcal{A}(pp, T, \{(\text{send}_j, \text{recv}_j, \{(x_{j,i}, a_{j,i})\}_{i=1}^n, a_{\mathcal{A},j})\}_{j \in \{0,1\}})$.
- Return 1 if $b = b'$, else 0.

Figure 2: Privacy experiment PRV

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

- $pp \leftarrow \text{Setup}(1^\lambda)$.
- \mathcal{A} is provided random $k^v, k_a^s, k_b^s \in \mathbb{F}$ to construct the address $\text{addr}_{\mathcal{A}} = (K_{\mathcal{A}}^{dh}, K_{\mathcal{A}}^v, K_{\mathcal{A}}^s)$, and $\text{enote}_{\mathcal{A}} = \{(C_i, K_i^o, m_i)\}_{i=1}^n$ in the ledger connected to $\text{addr}_{\mathcal{A}}$.
- \mathcal{A} chooses any receiver address $\text{addr}_{\mathcal{B}}$ (which can be $\text{addr}_{\mathcal{A}}$).
- Let $\text{addr}_{\mathcal{C}}$ be the random victim address, $\text{addr}_{\mathcal{D}}$ be another random address, and $\text{enote}_{\mathcal{C}}$ be all the e-notes in the ledger connected to $\text{addr}_{\mathcal{C}}$.
- $T \leftarrow \text{tx}(pp, \text{addr}_{\mathcal{C}}, \text{addr}_{\mathcal{D}}, \text{enote}_{\mathcal{C}})$. The owner of $\text{addr}_{\mathcal{C}}$ honestly spends all e-notes in $\text{enote}_{\mathcal{C}}$ to send some amounts to $\text{addr}_{\mathcal{D}}$. 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.
- $\{(k'_{a,i}, k'_{b,i})\}_{i=1}^n \leftarrow \mathcal{A}(pp, \text{enote}_{\mathcal{A}}, T)$.
- $T' \leftarrow \text{tx}(pp, \text{addr}_{\mathcal{A}}, \text{addr}_{\mathcal{B}}, \text{enote}_{\mathcal{A}}, \{(k'_{a,i}, k'_{b,i})\}_{i=1}^n)$. \mathcal{A} spends all e-notes in $\text{enote}_{\mathcal{A}}$ to send some amounts to $\text{addr}_{\mathcal{B}}$. 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 3.8 (Non-slanderability). *Seraphis is non-slanderable if for all PPT adversary \mathcal{A} , there exist a negligible function $\text{negl}(\lambda)$ such that*

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

where NSLAND is described in Figure 3.

Theorem 3.5 (Non-slanderability). *If PedersenK is hiding, LinkTag is hiding and binding, ownership and unspentness proof and membership proof are sound and witness indistinguishable, then Seraphis is non-slanderable.*

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] Marc Fischlin and Arno Mittelbach. An overview of the hybrid argument. Cryptology ePrint Archive, Paper 2021/088, 2021. <https://eprint.iacr.org/2021/088>.
- [6] 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>.
- [7] 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>.
- [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] 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>.
- [12] UkoeHB. Seraphis: Privacy-focused tx protocol. <https://github.com/UkoeHB/Seraphis>.

A Proofs of theorems in Section 3

We first present another hardness assumption which will be helpful for the next proof. This assumption is used in Bulletproofs [2] and Bulletproofs+ [3]:

Definition A.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 $\text{negl}(\lambda)$ such that*

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

It is said in [2] and [3] that this assumption and the DL assumption are equivalent.

Proof of Theorem 3.2. For perfect hiding, assume an adversary with unlimited computational power. It can easily find the DL of \tilde{K} base U , which is (k_b^o/k_a^o) . However, it still cannot find the inputted k_a^o and k_b^o because $K^o = t_k G + k_a^o X + (k_a^o)(k_b^o/k_a^o)U = t_k G + k_a^o(X + (k_b^o/k_a^o)U)$ is a Pedersen commitment for k_a^o with t_k as blinding factor.

Now we prove computational binding by contraposition. By breaking the binding of `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 $\text{LinkTag}(k_a^o, k_b^o; t_k) = \text{LinkTag}(k_a'^o, k_b'^o; t_k')$. 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 $G, X, U \in \mathbb{G}$. \square

Proof of Theorem 3.3. We prove by contraposition. Assume that \mathcal{A} succeeds in the BAL experiment with non-negligible probability. There are four cases as to *why* \mathcal{A} succeeded, each with sub-cases: $b_0 \wedge b_1 = 1$, $b_0 \wedge b_2 = 1$, $b_0 \wedge b_3 = 1$, or $b_0 \wedge b_4 = 1$.

For the first case, assume that $b_0 \wedge b_1 = 1$. [UNFINISHED]

For the second case, assume that $b_0 \wedge b_2 = 1$. One sub-case is that \mathcal{A} finds correct private scalars of C and K^o in some e-notes in `enote` _{$\neg \mathcal{A}$} , but this breaks the binding property of `PedersenC` and `PedersenK`. Another sub-case is that \mathcal{A} doesn't have correct private scalars of C , yet \mathcal{A} can non-negligibly create accepted proofs. [UNFINISHED]

For the third case, assume that $b_0 \wedge b_3 = 1$. One sub-case is that for some $i \in \{1, \dots, n\}$, \mathcal{A} finds $(k_{b,i}'^o, k_{a,i}'^o)$ which is not equal to $(k_{b,i}^o, k_{a,i}^o)$, [UNFINISHED]. 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.

For the fourth case, assume that $b_0 \wedge b_4 = 1$. 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 $i \in \{1, \dots, n\}$ such that $C'_i = v_{c,i}G + a_iH = v'_{c,i}G + a'_iH$ (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 `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. \square

Proof of Theorem 3.4. We prove by hybrid arguments [5]. [UNFINISHED] \square

Proof of Theorem 3.5. 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 $\tilde{K}_i \in \{\tilde{K}_C\}$ (base U), and the second step of \mathcal{A} is to find $(k_{a,i}'^o, k_{b,i}'^o, t_{k,i}')$ (for e-note image in T') such that $k_{b,i}'^o/k_{a,i}'^o = k_i$, making $\tilde{K}_i \in \{\tilde{K}_A\}$.

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_i^o 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 $K_{\pi_i}^o$ 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 witness indistinguishability of ownership and unspentness proof and membership proof.

For the second step, one case is that in T' , $K_i'^o = t'_{k,i}G + K^o$, with K^o the one-time address of an e-note in $\mathbf{enote}_{\mathcal{A}}$, but this breaks the binding property of **LinkTag**. The other case is the negation of the previous case: $K_i'^o \neq t'_{k,i}G + K^o$, yet \mathcal{A} can non-negligibly create accepted proofs, but this breaks the soundness property of ownership and unspentness proof. \square