A Report on Seraphis

coinstudent2048

July 2, 2023

Abstract

This document contains a concise description of koe's Seraphis [11], a novel privacy-preserving transaction protocol abstraction, and its security model. Note that this does not cover all of it, but a rather simple variant of it for easier security analysis. Because of this, extensions/modifications presented there, with some mentioned here, would not be analyzed here.

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 $H_0, H_1, G_0, G_1, G_2, J$ be generators of $\mathbb{G}$ with unknown DL relationship to each other (see Definition A.1 for the formalization); these 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 $\mathcal{H}_0$, 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.

1.2 E-notes and e-note images

Definition 1.1. An **e-note** *for scalars* $k_1^o, k_2^o, a \in \mathbb{F}$ *is a tuple* (C, K^o, m) *such that* $C = xH_0 + aH_1$ *for* $x \xleftarrow{\$} \mathbb{F}$, $K^o = k_0^o G_0 + k_1^o G_1 + k_2^o G_2$, *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 keys k_0^o, k_1^o, k_2^o (the o superscript indicates "one-time"), and m is the **memo field** which includes information helping wallet owners to know if they own the e-note. We say that someone *owns* an e-note if they know the corresponding scalars $k_0^o, k_1^o, k_2^o, a, x \in \mathbb{F}$. 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*

$$
\begin{aligned}
C' &= t_c H_0 + C \\
&= (t_c + x)H_0 + aH_1 \\
&= v_c H_0 + aH_1 \ , \\
K'^o &= t_k G_0 + K^o \\
&= (t_k + k_0^o)G_0 + k_1^o G_1 + k_2^o G_2 \\
&= v_k G_0 + k_1^o G_1 + k_2^o G_2 \ , \ and \\
\tilde{K} &= (k_2^o/k_1^o)J
\end{aligned}
$$

for $t_c, t_k \overset{\$}{\leftarrow} \mathbb{F}$ 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.

1.3 Sender-receiver shared secret scheme

Although Diffie-Hellman key exchange would be the most common implementation for this, any scheme that satisfies the following can be used:

Definition 1.3. *A **sender-receiver shared secret scheme** is a tuple of functions $(\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2)$ such that:*

- *If $X = \mathcal{S}_0(x)$ and $(R, S) = \mathcal{S}_1(r, X)$, then $S = \mathcal{S}_2(x, R)$.*

- *Given X, R, and the tuple, the problem of determining S satisfying the previous condition is hard.*

Definition 1.4. *A **receiver address** is a tuple (K^r, M^r) such that $K^r = k_0^r G_0 + k_1^r G_1 + k_2^r G_2$ and $M^r = \mathcal{S}_0(m^r)$ is arbitrary data.*

We say that someone *owns* a receiver address if they know the corresponding scalars $k_0^r, k_1^r, k_2^r, m^r \in \mathbb{F}$. Moreover, M^r will be used by senders to generate shared secrets.

1.4 Authenticated symmetric encryption scheme

We require the use of an authenticated symmetric encryption scheme. The shared secret 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 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 $\left(\sum_{i=1}^n a_i\right) \geq a_t$, all *connected* to a receiver address (K_{ali}^r, M_{ali}^r). This "connection" will be elaborated later on. On the other hand, Bob owns a receiver address (K_{bob}^r, M_{bob}^r). 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 $r_{ali}, r_{bob} \overset{\$}{\leftarrow} \mathbb{F}$ and independent to each other.

2. Compute $(S_{ali}, R_{ali}) = \mathcal{S}_1(r_{bob}, (K_{ali}^r, M_{ali}^r))$ and $(S_{bob}, R_{bob}) = \mathcal{S}_1(r_{bob}, (K_{bob}^r, M_{bob}^r))$ and store R_{ali} and R_{bob} to new empty memos m_{ali} and m_{bob}, respectively. The two secrets must be independent to each other.

3. Compute the one-time addresses

$$K_{ali}^o = \mathcal{H}_0(S_{ali})G_0 + \mathcal{H}_1(S_{ali})G_1 + \mathcal{H}_2(S_{ali})G_2 + K_{ali}^r$$
$$K_{bob}^o = \mathcal{H}_0(S_{bob})G_0 + \mathcal{H}_1(S_{bob})G_1 + \mathcal{H}_2(S_{bob})G_2 + K_{bob}^r.$$

4. Compute the amount commitments $C_{ali} = \mathcal{H}_3(S_{ali})G + a_c H$ and $C_{bob} = \mathcal{H}_3(S_{bob})G + a_t H$.

5. Encrypt the amounts: $\overline{a_c} = \texttt{enc}[\mathcal{H}_4(S_{ali})](a_c)$ and $\overline{a_t} = \texttt{enc}[\mathcal{H}_4(S_{bob})](a_t)$, and store $\overline{a_c}$ and $\overline{a_t}$ to memos m_{ali} and m_{bob}, respectively.

Alice now has two new e-notes: $\texttt{enote}_{ali} = (C_{ali}, K_{ali}^o, m_{ali})$ and $\texttt{enote}_{bob} = (C_{bob}, K_{bob}^o, m_{bob})$. 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 and the required security properties for application to Seraphis are described in Subsection 3.1.

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:

- 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 "Implementing Seraphis" [11], Section 10.

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 $\texttt{enimg}_i$ 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.

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_i^o, 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:

$$\{(K_i'^o, \tilde{K}_i \in \mathbb{G}; t_{k,i}, v_{k,i}, k_{1,i}^o, k_{2,i}^o \in \mathbb{F}) : K_i'^o = v_{k,i}G_0 + k_{1,i}^o G_1 + k_{2,i}^o G_2 \wedge k_{1,i}^o \neq 0 \wedge \tilde{K}_i = (k_{2,i}^o/k_{1,i}^o)J\}$$

4. Insert $\Pi_{\text{o\&u},i}$ 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.

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 $\texttt{enimg}_i$ 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:

$$D = d_a H_0 + d_b H_1 = C_{ali} + C_{bob} - \sum_{i=1}^n C_i'$$

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 $\texttt{enimg}_i$ 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 $\mathtt{enimg}_i$ 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 $\mathtt{enimg}_i$ in T.

2. If the masked address $K_i'^o$ is already in $\mathtt{enimg}_i$ 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 $\mathtt{enimg}_i$ 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 $\Pi_{\mathrm{mem},i}$ for a non-interactive proving system for the following relation:

$$\{(C_i', K_i'^o \in \mathbb{G}, \mathbb{S}_i \subset \mathbb{G}^2; \pi_i \in \mathbb{N}, t_{c,i}, t_{k,i} \in \mathbb{F}) : 1 \leq \pi_i \leq s \wedge C_i' - C_{\pi_i} = t_{c,i} H_0 \wedge K_i'^o - K_{\pi_i}^o = t_{k,i} G_0\}$$

6. Insert $\mathbb{S}_i$, $\Pi_{\mathrm{mem},i}$ to $\{\mathtt{enimg}_i, \ldots\}$ in T.

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

Specific proving systems satisfying the relation include CSAG (CLSAG [5] without linking) and One-out-of-Many proving system adapted from Groth and Bootle *et al.* [6, 1].

2.4 Range proofs

Range proof guarantees to everyone that the committed amount a lies in a range. For the new e-notes $\mathtt{enote}_{ali}$ and $\mathtt{enote}_{bob}$, Alice must do the following:

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

$$\{(C \in \mathbb{G}, a_{max} \in \mathbb{F}; x, a \in \mathbb{F}) : C = xH_0 + aH_1 \wedge 0 \leq a \leq a_{max}\}$$

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

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

Specific proving systems satisfying the relation 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

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

We denote the full construction of T as the function $\mathtt{tx}(\cdot)$. 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).

4

2. Compute the nominal sender-receiver shared secret: $S_{nom} = \mathcal{S}_2((k_0^r, k_1^r, k_2^r, m^r), R)$.

3. Compute the nominal spend public key: $K_{nom}^s = K^o - \mathcal{H}_0(S_{nom})G_0 - \mathcal{H}_1(S_{nom})G_1 - \mathcal{H}_2(S_{nom})G_2$. 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: $a = \mathsf{dec}[\mathcal{H}_4(S_{nom})](\bar{a})$.

5. Compute the nominal amount commitment: $C_{nom} = \mathcal{H}_3(S_{nom})H_0 + aH_1$. 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:

$$\tilde{K}_{nom} = \frac{k_2^r + \mathcal{H}_2(S_{nom})}{k_1^r + \mathcal{H}_1(S_{nom})} J.$$

   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: $(k_1^o, k_2^o, a, x) = (k_{a,bob}^s + \mathcal{H}_2(q_{nom}), k_{b,bob}^s, a, \mathcal{H}_3(q_{nom}))$. 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 transcation is complete for Alice. This finishes a Seraphis trancation.

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 [8].

Subsections 3.1 to 3.3 outline the required security properties of the cryptographic components for Seraphis, and Subsection 3.4 is the main security analysis of Seraphis. Proofs for these properties are found in Appendix A.

3.1   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 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 "$(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ 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 3.1 (Perfect Completeness). $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *is perfectly complete for* $\mathcal{R}$ *if for all* PPT *adversary* $\mathcal{A}$,

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

Definition 3.2 (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*

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

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

Definition 3.3 (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*

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

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

The well-known property stronger than witness indistinguishability is called *Special Honest-Verifier Zero Knowledge (SHVZK)* [6]. 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* 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 more desirable 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). The definitions of these properties can be found in [7], Appendix A.4.

3.3 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 3.4 (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*

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

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

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

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

A commitment scheme is *perfectly binding* if negl(λ) is replaced by 0.

The first kind of commitment we define is commonly known as **Pedersen commitments** [9]. We define two instances, PedersenC : $\mathbb{F}^2 \rightarrow \mathbb{G}$ and PedersenK : $\mathbb{F}^3 \rightarrow \mathbb{G}$ as follows:

$$\mathsf{PedersenC}(a; x) = xH_0 + aH_1$$
$$\mathsf{PedersenK}(k_1^o, k_2^o; k_0^o) = k_0^o G_0 + k_1^o G_1 + k_2^o G_2$$

PedersenC corresponds to the formulas of amount commitment C and masked amount commitment C', while PedersenK corresponds to the formulas of one-time address K^o.

Theorem 3.1 (From [9]). *Pedersen commitment is perfectly hiding and computationally binding under the DL assumption.*

We now define a custom commitment LinkTag : $\mathbb{F} \times \mathbb{F} \setminus \{0\} \times \mathbb{F} \times \mathbb{F} \rightarrow \mathbb{G}^2$ as follows:

$$\mathsf{LinkTag}(k_0^o, k_1^o, k_2^o; t_k) = ((t_k + k_0^o)G_0 + k_1^o G_1 + k_2^o G_2, (k_2^o/k_1^o)J)$$

with $t_k \xleftarrow{\$} \mathcal{F}$ being the blinding factor. LinkTag corresponds to the combination of formulas of masked address K'^o and linking tag $\tilde{K}$.

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

3.4 Seraphis security properties

The required security properties for Seraphis are loosely based on Omniring's security model, with modifications to fit Seraphis. The Omniring paper presents a rigorous treatment of RingCT constructions, providing precision for security analysis against several realistic attacks.

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 $\mathsf{BAL}(\mathcal{A}, 1^\lambda)$ on a PPT adversary $\mathcal{A}$. The adversary succeeds in the experiment (i.e. $\mathsf{BAL}(\mathcal{A}, 1^\lambda) = 1$) if 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 not equal to 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* negl(λ) *such that*

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

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.*

```
BAL(𝒜, 1^λ)

  • pp ← Setup(1^λ).

  • 𝒜 is provided random $k_0^r, k_1^r, k_2^r, m^r \in \mathbb{F}$ to construct the address $\text{addr}_\mathcal{A} = (K_\mathcal{A}^r, M_\mathcal{A}^r)$,
    $\{k_{0,i}^o, k_{1,i}^o, k_{2,i}^o, a_i, x_i\}_{i=1}^n$ 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 $\text{addr}_\mathcal{A}$.

  • 𝒜 chooses any receiver address $\text{addr}_\mathcal{B}$.

  • $\{k_{0,i}^{\prime o}, k_{1,i}^{\prime o}, k_{2,i}^{\prime o}, a_i', x_i'\}_{i=1}^n \leftarrow \mathcal{A}(pp, \text{enote}_\mathcal{A}, \text{enote}_{\neg\mathcal{A}}, \{k_{0,i}^o, k_{1,i}^o, k_{2,i}^o, a_i, x_i\}_{i=1}^n)$.

  • $T \leftarrow \text{tx}(pp, \text{addr}_\mathcal{A}, \text{addr}_\mathcal{B}, \text{enote}_\mathcal{A}, \text{enote}_{\neg\mathcal{A}}, \{k_{0,i}^{\prime o}, k_{1,i}^{\prime o}, k_{2,i}^{\prime o}, a_i', x_i'\}_{i=1}^n)$. 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 $\exists i \in \{1, \ldots, n\} : k_{2,i}^{\prime o}/k_{1,i}^{\prime o} \neq k_{2,i}^o/k_{1,i}^o$, else $:= 0$.

  • $b_4 := 1$ if $\sum_{i=1}^n a_i \neq a_t$, 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 $\mathsf{PRV}(\mathcal{A}, 1^\lambda)$ 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* $\mathsf{negl}(\lambda)$ *such that*

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

where PRV *is described in Figure 2.*

Theorem 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 $\mathsf{NSLAND}(\mathcal{A}, 1^\lambda)$ on a PPT adversary $\mathcal{A}$. This property prevents the following attack known as *denial-of-spending attack* [10]: 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.

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

- $pp \leftarrow \mathsf{Setup}(1^\lambda)$.

- $\mathcal{A}$ is provided two random potential sender addresses send_0 and send_1, sets of e-notes enote_0 and enote_1 (with $|\mathsf{enote}_0| = |\mathsf{enote}_1| = n$) 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 only one e-note in enote_0 and only 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.

- $b \stackrel{\$}{\leftarrow} \{0,1\}$.

- $T \leftarrow \mathsf{tx}(pp, \mathsf{send}_b, \mathsf{recv}_b, \mathsf{enote}_b, \{\mathbb{S}_i\}_{i=1}^n, a_{\mathcal{A},b})$. 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.

- $b' \leftarrow \mathcal{A}(pp, T, \{\mathsf{send}_j, \mathsf{recv}_j, \{x_{j,i}, a_{j,i}\}_{i=1}^n, a_{\mathcal{A},j}\}_{j \in \{0,1\}})$.

- Return 1 if $b = b'$, else 0.

Figure 2: Privacy experiment PRV

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

- $pp \leftarrow \mathsf{Setup}(1^\lambda)$.

- $\mathcal{A}$ is provided random $k_0^r, k_1^r, k_2^r, m^r \in \mathbb{F}$ to construct the address $\mathsf{addr}_\mathcal{A} = (K_\mathcal{A}^r, M_\mathcal{A}^r)$, $\{k_{0,i}^o, k_{1,i}^o, k_{2,i}^o, a_i, x_i\}_{i=1}^n$ that makes $\mathsf{addr}_\mathcal{A}$ connect to $\mathsf{enote}_\mathcal{A} = \{(C_i, K_i^o, m_i)\}_{i=1}^n$ in the ledger, all the other addresses $\mathsf{addr}_{\neg\mathcal{A}}$, all the other e-notes $\mathsf{enote}_{\neg\mathcal{A}}$ in the ledger, and all the connections between every e-note to every address.

- $\mathcal{A}$ chooses any receiver address $\mathsf{addr}_\mathcal{B}$, the victim address $\mathsf{addr}_\mathcal{C}$, and another address $\mathsf{addr}_\mathcal{D}$.

- $T \leftarrow \mathsf{tx}(pp, \mathsf{addr}_\mathcal{C}, \mathsf{addr}_\mathcal{D}, \mathsf{enote}_\mathcal{C})$. The owner of $\mathsf{addr}_\mathcal{C}$ honestly spends all e-notes in $\mathsf{enote}_\mathcal{C}$ to send some amounts to $\mathsf{addr}_\mathcal{D}$. 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.

- $\{k_{0,i}'^o, k_{1,i}'^o, k_{2,i}'^o\}_{i=1}^n \leftarrow \mathcal{A}(pp, \mathsf{enote}_\mathcal{A}, T)$.

- $T' \leftarrow \mathsf{tx}(pp, \mathsf{addr}_\mathcal{A}, \mathsf{addr}_\mathcal{B}, \mathsf{enote}_\mathcal{A}, \{k_{0,i}'^o, k_{1,i}'^o, k_{2,i}'^o\}_{i=1}^n)$. $\mathcal{A}$ spends all e-notes in $\mathsf{enote}_\mathcal{A}$ to send some amounts to $\mathsf{addr}_\mathcal{B}$. 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* negl(λ) *such that*

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

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] Marc Fischlin and Arno Mittelbach. An overview of the hybrid argument. Cryptology ePrint Archive, Paper 2021/088, 2021. https://eprint.iacr.org/2021/088.

[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   Proofs of theorems in Section 3 [WIP!]

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* $\mathsf{negl}(\lambda)$ *such that*

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

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 J, which is (k_2^o/k_1^o). However, it still cannot find the inputted k_0^o, k_1^o and k_2^o because $K'^o = (t_k + k_0^o)G_0 + k_1^o G_1 + (k_1^o)(k_2^o/k_1^o)G_2 = (t_k + k_0^o)G_0 + k_1^o(G_1 + (k_2^o/k_1^o)G_2)$ is a Pedersen commitment for k_1^o with $t_k + k_0^o$ as blinding factor.

For computational binding, by breaking the binding of LinkTag, $\mathcal{A}$ finds $(k_0^o, k_1^o, k_2^o; t_k)$ and $(k_0'^o, k_1'^o, k_2'^o; t_k')$ such that the two are *not* equal but LinkTag evaluates to the same commitment value. This implies that

$$(t_k + k_0^o - t_k' - k_0'^o)G_0 + (k_1^o - k_1'^o)G_1 + (k_2^o - k_2'^o)G_2 = 0$$

and this breaks the DL relation assumption of $G_0, G_1, G_2 \in \mathbb{G}$. $\qquad\square$

Proof of Theorem 3.3. Assume that $\mathcal{A}$ succeeds in the BAL experiment with non-negligible probability. $\quad\square$

Proof of Theorem 3.4. We prove by hybrid arguments [4]. $\qquad\square$

Proof of Theorem 3.5. Assume that $\mathcal{A}$ succeeds in the NSLAND experiment with non-negligible probability. $\qquad\square$