A Report on Seraphis

coinstudent2048

July 31, 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 through public randomness. Let $a_{max}, d_{1,max} \in \mathbb{F}$ (to be used in range proofs and account balance respectively) with $d_{1,max} \leq a_{max}$, 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*

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

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.

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* $(K^r, M^r) = \mathcal{S}_0(k_0^r, k_1^r, k_2^r, m^r)$ *such that* $K^r = k_0^r G_0 + k_1^r G_1 + k_2^r G_2$ *and* M^r *and* m^r *are public and private arbitrary data, respectively.*

We say that someone *owns* a receiver address if they know the corresponding scalars $k_0^r, k_1^r, k_2^r, m^r \in \mathbb{F}$.

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 $r_{ali}, r_{bob} \xleftarrow{\$} \mathbb{F}$ 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: $\texttt{enote}_{ali} = (C_{ali}, K^o_{ali}, m_{ali})$ 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 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 (O&U) 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 $\{(C_i, K^o_i, m_i)\}^n_{i=1}$, 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:

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

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 $\{(C_i, K^o_i, m_i)\}^n_{i=1}$, Alice must do the following:

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

$$D = d_0 H_0 + d_1 H_1 = C_{ali} + C_{bob} - \sum_{i=1}^{n} C'_i$$

   Note that d_0 is uniformly random because of t_c inside C'_i and random oracle $\mathcal{H}$ inside C_{ali} and C_{bob}, while d_1 is a publicly known extra amount (e.g. transaction fee).

2. Insert C'_i to $\texttt{enimg}_i$ in T, and store (d_0, d_1) to T.

The verifier must verify the amount balance $\sum^n_{i=1} C'_i + D = C_{ali} + C_{bob}$ and $0 \leq d_1 \leq d_{1,max}$.

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 $\texttt{enimg}_i$ 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 2.2, 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:

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

6. Insert $\mathbb{S}_i$, $\Pi_{\text{mem},i}$ to $\{\texttt{enimg}_i, \ldots\}$ 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 $\texttt{enote}_{ali}$ 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:

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

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

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

$$T = \{\texttt{enote}_{ali}, \texttt{enote}_{bob}, \Pi_{\text{ran},ali}, \Pi_{\text{ran},bob}, d_0, d_1, \{\texttt{enimg}_i, \Pi_{\text{o\&u},i}, \mathbb{S}_i, \Pi_{\text{mem},i}\}_{i=1}^n\}.$$

We denote the full construction of T as the function $\texttt{tx}(\cdot)$. This function would be used for describing Seraphis security properties (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 $\{R, \overline{a}\}$ (see the beginning of this whole section).

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: $K_{nom}^r = K^o - \mathcal{H}_0(S_{nom})G_0 - \mathcal{H}_1(S_{nom})G_1 - \mathcal{H}_2(S_{nom})G_2$. If $K_{nom}^r = K_{bob}^r$, 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: $a = \mathtt{dec}[\mathcal{H}_4(S_{nom})](\overline{a})$.

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 its corresponding scalars $k_0^o, k_1^o, k_2^o, a, x$ (e.g. $k_0^o = k_0^r + \mathcal{H}_0(S_{nom})$). 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 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 [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* $\mathcal{A}$*,*

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

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*

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

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

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

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

Fiat-Shamir heuristic is applied to make interactive protocols non-interactive. Moreover, it transforms interactive protocols satisfying PSHVZK into non-interactive (fully) zero-knowledge (NIZK) protocols in the random oracle model.

All proving systems for application to Seraphis must be non-interactive and at least have perfect completeness, computational soundness, and NIZK.

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*

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

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

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

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

A commitment scheme is *perfectly binding* if $\mathsf{negl}(\lambda)$ is replaced by 0.
The first kind of commitment we define is commonly known as **Pedersen commitments** [9]. We define two instances, $\mathsf{PedersenC} : \mathbb{F}^2 \to \mathbb{G}$ and $\mathsf{PedersenK} : \mathbb{F}^3 \to \mathbb{G}$ as follows:

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

$\mathsf{PedersenC}$ corresponds to the formulas of amount commitment C and masked amount commitment C', while $\mathsf{PedersenK}$ 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 $\mathsf{LinkTag} : \mathbb{F} \times \mathbb{F} \setminus \{0\} \times \mathbb{F} \times \mathbb{F} \to \mathbb{G}^2$ 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. $\mathsf{LinkTag}$ corresponds to the combination of formulas of masked address K'^o and linking tag $\tilde{K}$.

Theorem 3.2. $\mathsf{LinkTag}$ *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 linking tags are not generated from the given one-time private keys, leading to double-spend of owned e-notes, or 4) the amount of the new e-note for the receiver is larger than the supposed total amount of e-notes it owns.

Theorem 3.3 (Balance)**.** *Seraphis is balanced: 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.*

Next we consider the **Privacy** property, which means that an adversary should never be able to distinguish between transactions, 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 and receiver 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 which of the two is used, hence breaking the privacy of T.

Theorem 3.4 (Privacy)**.** *Seraphis is private: 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.*

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

- $pp \leftarrow \mathsf{Setup}(1^\lambda)$.
- $\mathcal{A}$ is provided the whole ledger $\{T_i\}$, scalars k_0^r, k_1^r, k_2^r, m^r to construct the address $\mathtt{addr}_\mathcal{A} = (K_\mathcal{A}^r, M_\mathcal{A}^r)$, scalars $\{k_{0,i}^o, k_{1,i}^o, k_{2,i}^o, a_i, x_i\}_{i=1}^n$ that makes $\mathtt{addr}_\mathcal{A}$ connect to $\{\mathtt{enote}\}_\mathcal{A} = \{(C_i, K_i^o, m_i)\}_{i=1}^n$ in the ledger, all the other addresses $\{\mathtt{addr}\}_{\neg\mathcal{A}}$, and knowledge of all the connections between every e-note to every address. The e-notes not owned by $\mathcal{A}$ are denoted as $\{\mathtt{enote}\}_{\neg\mathcal{A}}$.
- $\mathcal{A}$ chooses any receiver address $\mathtt{addr}_\mathcal{B}$.
- $T \leftarrow \mathcal{A}(pp, \{\mathtt{enote}\}_\mathcal{A}, \{\mathtt{enote}\}_{\neg\mathcal{A}}, \{k_{0,i}^o, k_{1,i}^o, k_{2,i}^o, a_i, x_i\}_{i=1}^n)$, where T is a transaction.
- $b_0 := 1$ 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 $\{\mathtt{enote}\}_{\neg\mathcal{A}}$, else $:= 0$.
- $b_3 := 1$ if all spent e-notes in T are owned by $\mathcal{A}$, but $\exists i : \tilde{K}_i \neq (k_{2,i}^o / k_{1,i}^o)J$, else $:= 0$.
- $b_4 := 1$ if all spent e-notes in T are owned by $\mathcal{A}$, but $a_t > \sum_{i=1}^n a_i$, else $:= 0$.
- Return $b_0 \wedge (b_1 \vee b_2 \vee b_3 \vee b_4)$.

Figure 1: Balance experiment BAL

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

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

Figure 2: Privacy experiment PRV

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.

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

- $pp \leftarrow \mathsf{Setup}(1^\lambda)$.
- $\mathcal{A}$ is provided the whole ledger $\{T_i\}$, scalars k_0^r, k_1^r, k_2^r, m^r to construct the address $\mathtt{addr}_\mathcal{A} = (K_\mathcal{A}^r, M_\mathcal{A}^r)$, scalars $\{k_{0,i}^o, k_{1,i}^o, k_{2,i}^o, a_i, x_i\}_{i=1}^n$ that makes $\mathtt{addr}_\mathcal{A}$ connect to $\{\mathtt{enote}\}_\mathcal{A} = \{(C_i, K_i^o, m_i)\}_{i=1}^n$ in the ledger, all the other addresses $\{\mathtt{addr}\}_{\neg\mathcal{A}}$, and knowledge of all the connections between every e-note to every address. The e-notes not owned by $\mathcal{A}$ are denoted as $\{\mathtt{enote}\}_{\neg\mathcal{A}}$.
- $\mathcal{A}$ chooses any receiver address $\mathtt{addr}_\mathcal{B}$, the victim address $\mathtt{addr}_\mathcal{C}$, and another address $\mathtt{addr}_\mathcal{D}$.
- $T \leftarrow \mathtt{tx}(pp, \mathtt{addr}_\mathcal{C}, \mathtt{addr}_\mathcal{D}, \{\mathtt{enote}\}_\mathcal{C})$. The owner of $\mathtt{addr}_\mathcal{C}$ honestly spends all e-notes in $\{\mathtt{enote}\}_\mathcal{C}$ to send some amounts to $\mathtt{addr}_\mathcal{D}$. Let $\{\tilde{K}_\mathcal{C}\}$ be all the linking tags in T.
- $\mathcal{A}$ is provided T and verifies it honestly. If $\mathcal{A}$ rejects T, then return 0.
- $T' \leftarrow \mathcal{A}(pp, \{\mathtt{enote}\}_\mathcal{A}, T)$, where T' is a transaction. 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

Theorem 3.5 (Non-slanderability)**.** *Seraphis is non-slanderable: for all* PPT *adversary* $\mathcal{A}$*, there exist a negligible function* $\mathsf{negl}(\lambda)$ *such that*

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

where NSLAND *is described in Figure 3.*

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*

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

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 $\mathsf{LinkTag}$, $\mathcal{A}$ 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 $\mathsf{LinkTag}$ 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 $G_0, G_1, G_2 \in \mathbb{G}$. □

Proof of Theorem 3.3. Assume that $\mathcal{A}$ succeeds in the BAL experiment with non-negligible probability. There are four cases to consider:

1. Case 1: $b_0 \wedge b_1 = 1$. $\mathcal{A}$ makes up $\texttt{enote}_* = (C, K^o, m)$, and pretends owning it. Then $\mathcal{A}$ has two possible actions: 1) $\mathcal{A}$ successfully found a set of corresponding scalars $k_0^o, k_1^o, k_2^o, a, x \in \mathbb{F}$. However this breaks the hiding or binding property of $\mathsf{PedersenC}$ and $\mathsf{PedersenK}$. 2) Even without a set of corresponding scalars, $\mathcal{A}$ managed to produce verified proofs for $\texttt{enote}_*$. For the case of C, if C' is generated correctly (i.e. $C' = t_c H_0 + C$), then $\mathcal{A}$ found $d_0, d_1 \in \mathbb{F}$ such that $d_0 H_0 + d_1 H_1 = C_{ali} + C_{bob} - C'$ for a successful amount balance check, but this breaks the DL assumption of $\mathbb{G}$. On the other hand, if $\mathcal{A}$ came up with $d_0, d_1 \in \mathbb{F}$, then $\mathcal{A}$ produced a verified membership proof because t_c is among the witness. But successfully finding t_c breaks the DL assumption of $\mathbb{G}$ and not finding it breaks the soundness of membership proof. For the case of K^o, $\mathcal{A}$ produced a verified o&u proof because k_1^o, k_2^o are among the witnesses, but this breaks the soundness of o&u proof.

2. Case 2: $b_0 \wedge b_2 = 1$. This is the same with Case 1 but with $\mathcal{A}$ knowing the $\texttt{addr}_*$ connected to an $\texttt{enote}_* \in \{\texttt{enote}\}_{\neg\mathcal{A}}$ and the rest of the transaction T_* containing $\texttt{enote}_*$. Then $\mathcal{A}$ has four possible actions: 1) $\mathcal{A}$ successfully found a set of corresponding scalars $k_0^r, k_1^r, k_2^r, m^r \in \mathbb{F}$ of $\texttt{addr}_*$, 2) $\mathcal{A}$ successfully found just S_{nom}, 3) $\mathcal{A}$ successfully found a set of corresponding scalars $k_0^o, k_1^o, k_2^o, a, x \in \mathbb{F}$ with the use of T_*, and 4) even without a set of corresponding scalars, $\mathcal{A}$ managed to produce verified proofs for $\texttt{enote}_*$ with the use of T_*. Actions 1 and 2 both break the shared-secretness of $(\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2)$. For action 1, it is because $\mathcal{A}$ can now easily compute S_{nom} through $\mathcal{S}_2$. Action 3 breaks either the hiding or binding property of $\mathsf{PedersenC}$ (from range proof) or $\mathsf{LinkTag}$ (from o&u proof), or NIZK of proving systems. Lastly, action 4 also breaks NIZK of proving systems, because action 4 implies that $\mathcal{A}$ obtained some information about the witnesses in proofs of T_*.

3. Case 3: $b_0 \wedge b_3 = 1$. This means for some i, $\mathcal{A}$ made up $\tilde{K}_{i,*} \in \mathbb{G}$ and then produced a verified o&u proof with $\tilde{K}_{i,*}$ in statement, but $\tilde{K}_{i,*} \neq (k_{2,i}^o / k_{1,i}^o) J$, but this breaks the soundness of o&u proof.

4. Case 4: $b_0 \wedge b_4 = 1$. For a start, the soundness of range proof prevents $\mathcal{A}$ in producing a valid transaction such that $a_t > a_{max}$. If $\mathcal{A}$ just made a_t to be greater than $\sum_{i=1}^n a_i$ and proceed honestly, then the d_1 in account balance becomes "negative", which implies that account balance is violated. If account balance is satisfied, then $\mathcal{A}$ successfully found $(a', x') \neq (a_i, x_i)$ such that $\mathsf{PedersenC}(a'; x') = C_i$ for some i, but this breaks the binding property of $\mathsf{PedersenC}$.

This completes the proof. □

Proof of Theorem 3.4. We prove by hybrid arguments [4]. □

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