A Report on Seraphis

coinstudent2048

September 12, 2021

Abstract

This document contains a concise description of Seraphis [2], a novel privacy-preserving transaction protocol abstraction, and a security analysis for it.

1 Preliminaries

1.1 Public parameters and notations

Let $\mathbb{G}$ be a prime order group where the Discrete Logarithm (DL) problem is hard and the Decisional Diffie-Hellman assumption (DDH) holds, and let $\mathbb{F}$ be its scalar field. Let G_0, G_1, H_0, H_1 be generators of $\mathbb{G}$ with unknown DL relationship to each other. Note that these generators may be produced using public randomness. Let $\mathcal{H} : \{0,1\}^* \to \mathbb{F}$ be a cryptographic hash function. We add a subscript to $\mathcal{H}$, such as $\mathcal{H}_1$, in lieu of domain-separating the hash function explicitly; any domain-separation method may be used in practice.

The notation $\leftarrow_R$ 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 $C = xH_0 + aH_1$ for $x \leftarrow_R \mathbb{F}$, $K^o = k_b^o G_0 + k_a^o G_1$, and m is an arbitrary data.*

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

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

$$\begin{aligned} C' &= t_c H_o + C \\ &= (t_c + x)H_0 + aH_1 \\ &= v_c H_o + aH_1 \ , \\ K'^o &= t_k G_0 + K^o \\ &= (t_k + k_b^o)G_0 + k_a^o G_1 \\ &= v_k G_0 + k_a^o G_1 \ , \ \textit{and} \\ \tilde{K} &= (1/k_a^o)G_0 \end{aligned}$$

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

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

Definition 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 $K^s = k_b^s G_0 + k_a^s G_1$.*

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

1.3 Symmetric encryption scheme

We require the use of a symmetric encryption scheme. The Diffie-Hellman base public key enables shared secrets between the sender and the receiver. We denote the encryption and decryption of data x with key k as $\texttt{enc}[k](x)$ and $\texttt{dec}[k](x)$, respectively. We put overlines (e.g. $\overline{x}$) to indicate encrypted data.

2 A Seraphis transaction

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

1. Generate $r_{ali}, r_{bob} \leftarrow_R \mathbb{F}$ and independent to each other.
2. Compute $R_{ali} = r_{ali} K^{dh}_{ali}$ and $R_{bob} = r_{bob} K^{dh}_{bob}$, then store R_{ali} and R_{bob} to new (empty) memos m_{ali} and m_{bob}, respectively. The name for K^{dh} should now be clear.
3. Compute the sender-receiver shared secrets $q_{ali} = \mathcal{H}_1(r_{ali} K^v_{ali})$ and $q_{bob} = \mathcal{H}_1(r_{bob} K^v_{bob})$.
4. Compute the one-time addresses $K^o_{ali} = \mathcal{H}_2(q_{ali}) G_1 + K^s_{ali}$ and $K^o_{bob} = \mathcal{H}_2(q_{bob}) G_1 + K^s_{bob}$. It is easy to see that $\mathcal{H}_2(q_{ali})$ and $\mathcal{H}_2(q_{ali})$ are uniformly random in the random oracle model.
5. Compute the amount commitments $C_{ali} = \mathcal{H}_3(q_{ali}) H_0 + a_c H_1$ and $C_{bob} = \mathcal{H}_3(q_{bob}) H_0 + a_t H_1$. It is easy to see that the blinding factors $\mathcal{H}_3(q_{ali})$ and $\mathcal{H}_3(q_{bob})$ are uniformly random in the random oracle model.
6. Encrypt the amounts: $\overline{a_c} = \texttt{enc}[q_{ali}](a_c)$ and $\overline{a_t} = \texttt{enc}[q_{ali}](a_t)$, and store $\overline{a_c}$ and $\overline{a_t}$ to memos m_{ali} and m_{bob}, respectively.

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

On another note, a Seraphis transaction can easily have multiple receivers aside from Bob, which implies that Alice will create more than 2 new e-notes. We did not present this more general instance of Seraphis for the sake of simpler security analysis; extending such analysis to that case must be easy to carry out.

2.1 Ownership and unspentness proofs

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

1. Generate a *partial* e-note image for (C_i, K^o_i, m_i): $\texttt{enimg}_i = (K'^o_i, \tilde{K}_i)$.
2. Prepare the proof transcripts $\Pi_{\text{o\&u},i}$ for a non-interactive proving system for the following relation:
$$\{(G_0, G_1, K'^o_i, \tilde{K}_i \in \mathbb{G}; v_k, k^o_a \in \mathbb{F}) : k^o_a \neq 0 \wedge K'^o_i = v_k G_0 + k^o_a G_1 \wedge \tilde{K}_i = (1/k^o_a) G_0\}$$
3. Store $(\texttt{enimg}_i, \Pi_{\text{o\&u},i})$ to T.

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

2.2 Amount balance

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

1. Generate the masked amount commitment C_i' for (C_i, K_i^o, m_i) as per definition, *except* for $i = n$. For the case of $i = n$, set
$$v_{c,n} = \mathcal{H}_3(q_{ali}) + \mathcal{H}_3(q_{bob}) - \sum_{i=1}^{n-1} v_{c,i}.$$
Note that the value of $v_{c,n}$ is still uniformly random because the values of $t_{c,i}$ for $i \in \{1, \dots, n-1\}$ are uniformly random.

2. Insert C_i' to $\texttt{enimg}_i$ in T to complete the e-note image.

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

2.3 Membership proofs

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

1. 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**.

2. 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: $\mathbb{S}_i = \{(C_j, K_j^o)\}_{j=1}^s$.

3. Prepare the proof transcripts $\Pi_{\text{mem},i}$ for a non-interactive proving system for the following relation:
$$\{(G_0, H_0, C_i', K_i'^o \in \mathbb{G}, \mathbb{S}_i \subset \mathbb{G}^2; \pi \in \mathbb{N}, t_c, t_k \in \mathbb{F}) : 1 \leq \pi \leq s \wedge C_i' - C_\pi = t_c H_0 \wedge K_i'^o - K_\pi^o = t_k G_0\}$$

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

2.4 Range proofs

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

1. Prepare the respective proof transcripts $\Pi_{\text{ran},ali}$ and $\Pi_{\text{ran},bob}$ for a non-interactive proving system for the following relation:
$$\{(H_0, H_1, C \in \mathbb{G}; x, a \in \mathbb{F}) : C = xH_0 + aH_1 \wedge 0 \leq a \leq a_{max}\}$$

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

2.5 Receipt

Once the construction of T is completed, Alice sends it to the network. Its contents must now be
$$T = (\texttt{enote}_{ali}, \texttt{enote}_{bob}, \Pi_{\text{ran},ali}, \Pi_{\text{ran},bob}, \{(\texttt{enimg}_i, \Pi_{\text{o\&u},i}, \mathbb{S}_i, \Pi_{\text{mem},i})\}_{i=1}^n).$$

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

1. Get a new e-note (C, K^o, m) in T. Note that m contains $(R, \overline{a})$ (see the beginning of Section 2).

2. Compute the nominal sender-receiver shared secret: $q_{nom} = \mathcal{H}_1(k_{bob}^v R)$.

3. Compute the nominal spend public key: $K_{nom}^s = K^o - \mathcal{H}_2(q_{nom})G_1$. 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 Section 2). Otherwise (if not equal), the e-note is not connected, and hence go to Step 1.

4. Decrypt the amount: $a = \texttt{dec}[q_{nom}](\overline{a})$.

5. Compute the nominal amount commitment: $C_{nom} = \mathcal{H}_3(q_{nom})H_0 + aH_1$. If $C_{nom} \neq C$, then the e-note is malformed and cannot be spent.

6. Compute the nominal linking tag: $\tilde{K}_{nom} = (1/(k^s_{a,bob} + \mathcal{H}_2(q_{nom})))G_0$. If he finds a copy of $\tilde{K}_{nom}$ in the ledger, then the e-note has already been spent.

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

3.1 Zero-knowledge arguments of knowledge

- *Perfectly Complete*:
- *Special Sound*:
- *Special Honest Verifier Zero-Knowledge*:

Fiat-Shamir heuristic [1] transforms sigma protocols with the above properties into non-interactive zero-knowledge arguments of knowledge (NIZKAoKs) in the random oracle model. We now assume that the proving systems for Ownership and Unspentness proofs, and Range proofs are NIZKAoKs.

3.2 Membership proof security properties

- *Correctness*:
- *Unforgeability*:
- *Anonymity*:

3.3 Peterson commitments

- *Perfectly Hiding*:
- *Computationally Binding*:

3.4 Symmetric encryption scheme

3.5 Seraphis security properties

3.6 Theorems

References

[1] 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.

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