A Report on Seraphis

coinstudent2048

September 11, 2021

Abstract

This document contains a concise description of Seraphis [1], 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 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 $k_a^o, k_b^o, a \in \mathbb{F}$.

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

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

This section describes a simplified instance of Seraphis.

Suppose that Alice would send $a_t \in \mathbb{F}$ amount of funds to Bob. Alice owns a set of e-notes $\{(C_i, K_i^o, m_i)\}_{i=1}^n$ with a total amount of $\left(\sum_{i=1}^n a_i\right) \geq a_t$, all *connected* to a receiver address $(K_{ali}^{DH}, K_{ali}^v, K_{ali}^s)$. This "connection" will be elaborated later on. On the other hand, Bob owns a receiver address $(K_{bob}^{DH}, K_{bob}^v, K_{bob}^s)$. For Bob to receive the funds, he will now send his receiver address to Alice. Alice will actually send funds to two addresses: to Bob's and to herself (for the "change" $a_c = \sum_{i=1}^n a_i - a_t$). 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 $R_{ali} = r_{ali}K_{ali}^{DH}$ and $R_{bob} = r_{bob}K_{bob}^{DH}$, then store R_{ali} and R_{bob} to new (empty) memos m_{ali} and m_{bob}, respectively. The name for K^{DH} should now be clear.

3. Compute the sender-receiver shared secrets $q_{ali} = \mathcal{H}_1(r_{ali}K_{ali}^v)$ and $q_{bob} = \mathcal{H}_1(r_{bob}K_{bob}^v)$.

4. Compute the one-time addresses $K_{ali}^o = \mathcal{H}_2(q_{ali})G_1 + K_{ali}^s$ and $K_{bob}^o = \mathcal{H}_2(q_{bob})G_1 + K_{bob}^s$. It is easy to see that $\mathcal{H}_2(q_{ali})$ and $\mathcal{H}_2(q_{ali})$ are uniformly randomly generated in the random oracle model.

5. Compute the amount commitments $C_{ali} = \mathcal{H}_3(q_{ali})H_0 + a_tH_1$ and $C_{bob} = \mathcal{H}_3(q_{bob})H_0 + a_tH_1$. It is easy to see that the blinding factors $\mathcal{H}_3(q_{ali})$ and $\mathcal{H}_3(q_{bob})$ are uniformly randomly generated 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 $(C_{ali}, K_{ali}^o, m_{ali})$ and $(C_{bob}, K_{bob}^o, m_{bob})$. 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 are discussed in the next subsections.

On another note, a Seraphis transaction can easily have multiple receivers aside from Bob, which means 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 trivial.

2.1 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 will be 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:

$$\{(\mathbb{S}_i; \pi \in \mathbb{N}) : 1 \leq \pi \leq s \wedge (C_\pi, K^o_\pi) = (C_i, K^o_i)\}$$

4. Store $(\mathbb{S}_i, \Pi_{\text{mem},i})$ to T.

2.2 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): $\texttt{eimg}_i = (K'^o_i, \tilde{K}_i)$.
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. Append $(\texttt{eimg}_i, \Pi_{\text{o\&u},i})$ to $(\mathbb{S}_i, \Pi_{\text{mem},i})$ in T.

2.3 Amount balance 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 masked amount commitment C'_i for (C_i, K^o_i, m_i) as per definition, *except* for $i = n$. For the case of $i = n$, [insert stuff here].
2. Insert C'_i to $\texttt{eimg}_i$ in T to complete the e-note image.

2.4 Range proofs

2.5 Receipt

3 Security Analysis

3.1 Zero-knowledge proofs

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

3.2 Anonymity

3.3 Peterson commitments

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

3.4 Symmetric encryption scheme

3.5 Seraphis security properties

3.6 Theorems

References

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