A Report on Seraphis

coinstudent2048

September 19, 2021

Abstract

This document contains a concise description of Seraphis [4], a novel privacy-preserving transaction protocol abstraction, and its security model. This document will also serve as a suggestion to the organization of the contents of the final Seraphis paper.

1 Introduction

In a p2p (peer-to-peer) electronic cash system, the entire supply of currency exists as a digital record that can be stored by any person, and transactions (attempts to transfer money to new owners) are mediated by a network of *peers* (usually called *nodes*)... ...An unfortunate consequence of cryptocurrencies being decentralized is that the ledger is *public*, implying that all e-notes and transaction events are public knowledge. If amounts are in cleartext, addresses are trivially traceable, and e-notes to be spent are referenced directly, then observers can discern many details about users' finances.

Hence, several confidential transaction protocols have been proposed to ensure financial privacy and currency fungibility. [List privacy technologies and limitations]

We introduce Seraphis, privacy-preserving transaction protocol abstraction, which means the unlike previous...

1.1 Concurrent Works

Lelantus Spark

2 Preliminaries

2.1 Public parameters

Let λ be the security parameter. Let $\mathbb{G}$ be a prime order group based on λ where the Discrete Logarithm (DL), Computational Diffie-Hellman (CDH), and Decisional Diffie-Hellman (DDH) problems are hard, and let $\mathbb{F}$ be its scalar field. Let G, H, U 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 $\overset{\$}{\leftarrow}$ 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.

2.2 E-notes and e-note images

Definition 2.1. *An* **e-note** *for scalars $k_a^o, k_b^o, a \in \mathbb{F}$ is a tuple (C, K^o, m) such that $C = xG + aH$ for $x \overset{\$}{\leftarrow} \mathbb{F}$, $K^o = k_b^o G + k_a^o U$, 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 2.2. *An* **e-note image** *for an e-note* (C, K^o, m) *is a tuple* $(C', K'^o, \tilde{K})$ *such that*

$$
\begin{aligned}
C' &= t_c G + C \\
&= (t_c + x)G + aH \\
&= v_c G + aH \ , \\
K'^o &= t_k G + K^o \\
&= (t_k + k_b^o)G + k_a^o U \\
&= v_k G + k_a^o U \ , \ and \\
\tilde{K} &= (1/k_a^o)G
\end{aligned}
$$

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**.

Definition 2.3. *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* $K^s = k_b^s G + k_a^s U$.

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 subsection, while the reason for the names of superscripts will be discussed in the addressing schemes (Subsection 4.1). We say that someone *owns* a receiver address if they know the corresponding scalars $k^v, k_a^s, k_b^s \in \mathbb{F}$.

2.3 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.

3 A Seraphis transaction

We now describe a simple Seraphis transaction. This will be used as the basis for further instantiations and modifications (Section 4) and for the security model (Section 5).

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" $a_c = \sum_{i=1}^n a_i - a_t$ *even if* $a_c = 0$). Hence, Alice must create 2 new e-notes. She starts the transaction by doing the following:

1. Generate $r_{ali}, r_{bob} \xleftarrow{\$} \mathbb{F}$ and independent to each other.

2. Compute $R_{ali} = r_{ali} K_{ali}^{dh}$ and $R_{bob} = r_{bob} K_{bob}^{dh}$, then store R_{ali} and R_{bob} to new (empty) memos m_{ali} and m_{bob}, respectively. The name for K^{dh} should now be clear.

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

4. Compute the one-time addresses $K_{ali}^o = \mathcal{H}_2(q_{ali})U + K_{ali}^s$ and $K_{bob}^o = \mathcal{H}_2(q_{bob})U + K_{bob}^s$. 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 $C_{ali} = \mathcal{H}_3(q_{ali})G + a_c H$ and $C_{bob} = \mathcal{H}_3(q_{bob})G + a_t H$. 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: $\mathtt{enote}_{ali} = (C_{ali}, K_{ali}^o, m_{ali})$ and $\mathtt{enote}_{bob} = (C_{bob}, K_{bob}^o, m_{bob})$. These will then be stored to a new (empty) *whole transaction* T. Other objects that will be stored to the whole transaction are from proving systems, which can be executed in *any* order. Proving systems are discussed in the next subsections.

For specific instances of Seraphis, there might be changes in some parts of the above steps, and by reflection, in some parts of the Receipt. Here are some notable changes:

- For some addressing schemes, the input to $\mathcal{H}_2$, the input to $\mathcal{H}_3$, and the key for both $\mathtt{enc}$ and $\mathtt{dec}$ may be constructed differently and different to each other. Nevertheless, these inputs and key must still be sender-receiver shared secrets.

- A Seraphis transaction can easily have multiple receivers aside from Bob, which implies that Alice will create more than 2 new e-notes. See Subsection 5.6 for a discussion of how this affects the security model.

- A Seraphis transaction can be collaboratively constructed by multiple players. This is the subject of the so-called "proof dependency" (Subsection 4.3). Also see Subsection 5.6 for a discussion of how this affects the security model.

3.1 Ownership and unspentness 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. If the masked address $K_i'^o$ is already in the e-note image $\mathtt{enimg}_i$ 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.

2. If the linking tag $\tilde{K}_i$ is already in $\mathtt{enimg}_i$ 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 $\mathtt{enimg}_i$ in T.

3. Prepare the proof transcripts $\Pi_{\mathrm{o\&u},i}$ for a non-interactive proving system for the following relation:

$$\{(G, U, K_i'^o, \tilde{K}_i \in \mathbb{G}; v_k, k_a^o \in \mathbb{F}) : k_a^o \neq 0 \wedge K_i'^o = v_k G + k_a^o U \wedge \tilde{K}_i = (1/k_a^o)G\}$$

4. Append $\Pi_{\mathrm{o\&u},i}$ to $(\mathtt{enimg}_i, \ldots)$ in T.

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

3.2 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. If the masked amount commitment C_i' is already in $\mathtt{enimg}_i$ in T, then exit this subsection. Else generate C_i' from (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 $i \in \{1, \ldots, n-1\}$ are uniformly random.

2. Insert C_i' to $\mathtt{enimg}_i$ in T.

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

3.3 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. 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 3.2, 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 transcripts $\Pi_{\mathrm{mem},i}$ for a non-interactive proving system for the following relation:

$$\{(G, C_i', K_i'^o \in \mathbb{G}, \mathbb{S}_i \subset \mathbb{G}^2; \pi \in \mathbb{N}, t_c, t_k \in \mathbb{F}) : 1 \leq \pi \leq s \wedge C_i' - C_\pi = t_c G \wedge K_i'^o - K_\pi^o = t_k G\}$$

6. Append $(\mathbb{S}_i, \Pi_{\mathrm{mem},i})$ to $(\texttt{enimg}_i, \ldots)$ in T.

3.4 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 $\Pi_{\mathrm{ran},ali}$ and $\Pi_{\mathrm{ran},bob}$ for a non-interactive proving system for the following relation:

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

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

3.5 Receipt

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

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

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 $(R, \bar{a})$ (see the beginning of this whole section).

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

3. Compute the nominal spend public key: $K_{nom}^s = K^o - \mathcal{H}_2(q_{nom})U$. 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: $a = \texttt{dec}[q_{nom}](\bar{a})$.

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

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

4

If an e-note (C, K^o, m) is connected to Bob's receiver address, then he knows the corresponding scalars of that e-note: $(k_a^o, k_b^o, a, x) = (k_{a,bob}^s + \mathcal{H}_2(q_{nom}), k_{b,bob}^s, a, \mathcal{H}_3(q_{nom}))$. Hence, "connection" implies e-note ownership. The transaction is complete for Bob.

For Alice to receive the "change" e-note, she must do the same above steps. After that, the transcation is complete for Alice. This finishes a Seraphis trancation.

4   Instantiations and Modifications

There are a number of details to consider when implementing Seraphis in a real cryptocurrency. Aside from instances of proving systems mentioned already in the previous section, this section is comprised of 'recommendations' for instantiations and modifications of other parts of Seraphis, which are inspired by historical privacy-focused cryptocurrency implementations.

4.1   Addressing schemes

4.2   Multisignature operations

4.3   Proof dependency

Transaction Chaining

4.4   Transaction fees

4.5   Coinbase transactions

4.6   Squashed e-note model

4.7   ?????

5   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 [3].

Subsections 5.1 to 5.4 outline the required security properties of the cryptographic components for Seraphis, then Subsection 5.5 is the main security analysis of Seraphis, and Subsection 5.6 discusses how some instantiations and modifications described in Section 4 affect the security analysis.

5.1   Zero-knowledge proofs

- *Completeness*:

- *Special Soundness*:

- *Special Honest Verifier Zero-Knowledge*:

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

5.2   Membership proof security properties

- *Completeness*:

- *Unforgeability*:

- *Anonymity*:

5.3 Peterson commitments

- *Perfectly Hiding*:

- *Computationally Binding*:

5.4 Symmetric encryption scheme

IND-CCA2

5.5 Seraphis security properties

From Omniring [2]

Completeness: This property immediately follows from the completeness properties of the cryptographic components and by inspection of the protocol description.

- *Balance*:

- *Privacy*:

- *Non-slanderability*:

Insert theorems here...

5.6 Discussions

6 Efficiency

Acknowledgements

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] 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`.

[3] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

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

A Composition proving system

B Security proofs for Seraphis

C ?????