

Seraphis Balance Recovery

Seraphis: <https://github.com/UkoeHB/Seraphis>

Jamtis: <https://gist.github.com/tevador/50160d160d24cfc6c52ae02eb3d17024>

Part 1: Warm-Up

Tx Protocol Structure (**'Enote'** Paradigm)

- **Enote**: an 'amount' that is 'owned' (aka output/Txo)
- **Transaction Events**: spend owned enotes, create new enotes
 - **Inputs**: spend old enotes
 - **Outputs**: new enotes with new owners
- **Blockchain**: series of blocks of transactions
 - **Tx in Block**: spends enotes from txs in prior blocks

Balance Recovery

- **New Funds**: identify owned enotes in transactions
- **Spent Funds**: locate where/if owned enotes are spent
- **User Balance**: $\text{sum}(\text{owned}) - \text{sum}(\text{owned} + \text{spent})$

Design Goals: Enote Ownership

- Privacy
- Tiered Information Access
- Principle of Least Astonishment

Notation

- **EC Points**: capital letters (K or G)
- **EC Scalars (and bytestrings)**: lower-case letters (x)
- **EC Operations**: additive notation ($K + G$ or $x G + a H$)
- **Generators**: G, H, X, U
- **Hash Functions**: always domain-separated in Seraphis (not Cryptonote)
 - $Hx()$: hash to x bytes
 - $Hn()$: hash to scalar
 - $Hp()$: hash to point
 - $H..[k]()$: keyed hash

Part 2: Cryptonote Recap

Cryptonote SubAddresses

- Cryptonote Keys
 - View Key: k_v
 - Spend Key: k_s
- Address Index Modifier: for subaddress index i
 - Spend Key Extension: $H_n(k_v || j)$
- Address: for subaddress index i
 - Spend Key (DH Base Pubkey): $K^i_s = H_n(k_v || i) G + k_s G$
 - View Key (DH Pubkey): $K^i_v = k_v K^i_s$

Cryptonote Enotes (To Subaddress)

- **Enote Construction:** given $\{K^i_v, K^i_s\}$, a , PID (optional), t (out index)
 - **Enote Ephemeral Pubkey** ($r_t = \text{gen_scalar}()$): $R_t = r_t K^i_s$
 - **Diffie-Hellman Derivation:** $K_d = 8 r_t K^i_v$
 - **Sender-Receiver Secret:** $q = \text{Hn}(K_d \parallel t)$
 - **Amount Commitment:** $C = \text{Hn}(\text{"commitment_mask"} \parallel q) G + a H$
 - **Encoded Amount:** $a_{\text{enc}} = a \wedge \text{H8}(\text{"amount"} \parallel q)$
 - **One-Time Address:** $K_o = \text{Hn}(q) G + K^i_s$
 - **View Tag:** $\text{view_tag} = \text{H1}(\text{"view_tag"} \parallel K_d \parallel t)$
 - **Encrypted Payment ID (optional):** $\text{PID_enc} = \text{PID} \wedge \text{Hn}(K_d \parallel 0x8d)$
- **Enote:** $\{K_o, C, a_{\text{enc}}, \text{view_tag}, \text{PID_enc}\}, R_t$

Cryptonote Key Images

- Key Image: $KI = (Hn(8 k_v R_t || t) + Hn(k_v || i) + k_s) Hp(Ko)$
- View/Sign Key Images: $\{k_v, k_s\}$ and $\{R_t, t, i\}$

View Scanning (1)

- Scan Info: $\{\{K_o, C, a_{enc}, view_tag, PID_{enc}\}, R_t, t\}$
- View-Key Scan: given $\{k_v, k_s G\}$
 - Pregenerate Spend Keys: $\{..., K^i_s, ...\} = Hn(k_v || i) G + k_s G$
 - Nominal DH Derivation: $K_d' = 8 k_v R_t$
 - Nominal View Tag: $view_tag' = H1("view_tag" || K_d' || t)$
 - If $view_tag' \neq view_tag$ then ABORT
 - Nominal Sender-Receiver Secret: $q' = Hn(K_d' || t)$
 - Nominal Spend Key: $K^i_s' = K_o - Hn(q') G$
 - If K^i_s' not in $\{..., K^i_s, ...\}$ then ABORT
 - Payment ID: $PID = PID_{enc} \wedge Hn(K_d' || 0x8d)$ (ignore if $0x00s$)

View Scanning (2)

- Scan Info: $\{\{K_o, C, a_enc, view_tag, PID_enc\}, R_t, t\} + \{q', i, PID\}$
 - Amount Recovery
 - Nominal Amount: $a' = a_enc \wedge H8(".. " || q')$
 - If $C \neq Hn(".. " || q')$ $G + a' H$ then ABORT (malformed)
 - Result: $\{\{..enote..\}, R_t, t, a, i, PID\}$ (no KI)

Part 3: Seraphis Balance Recovery

Seraphis Enotes

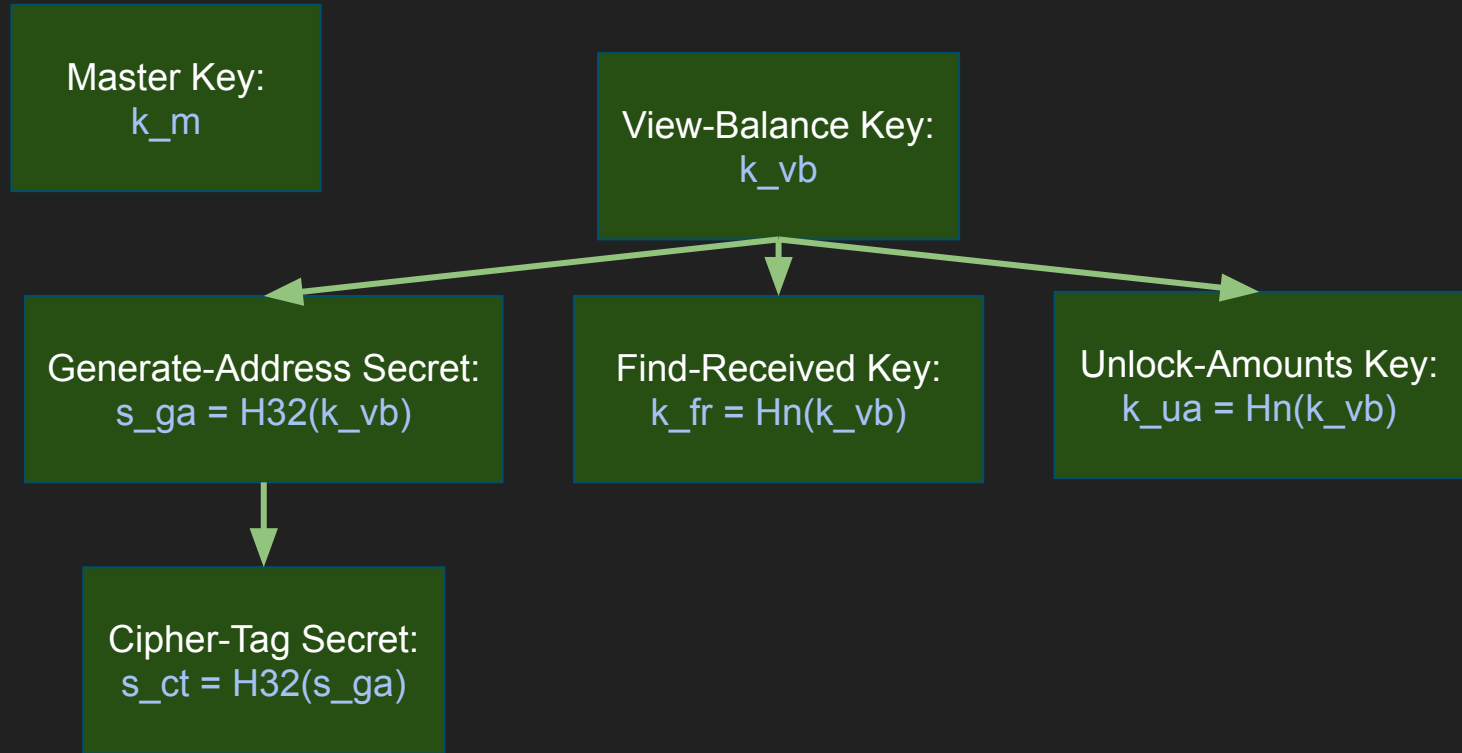
- **Enote**: {owner, amount} pair (for a tx output)
 - **One-Time Address**: $K_o = k_a X + k_b U$
 - **Amount Commitment (amount 'a')**: $C = x G + a H$
 - **Encoded Amount**: $a_{enc} = enc(a)$
- **Enote Image**: representation of an enote (for a tx input)
 - **Masked Address**: $K'' = t_k G + K_o$
 - **Masked Amount Commitment**: $C'' = t_c G + C$
 - **Key Image**: $KI = (k_b / k_a) U$

Balance Recovery (outline)

- **Recover Owned**: examine an enote $\{K_o, C, a_enc\}$
 - **Try Reproduce Address**: $K_o' \neq K_o$
 - **Decode Amount**: $a = dec(a_enc)$
 - **Try Reproduce Amount Commitment**: $C' \neq C$
 - **Store Key Image**: $KI = (k_b / k_a) \cup$
- **Identify Spent**: examine an enote image $\{K'', C'', KI\}$
 - **Compare Key Images**: `enote_store.has(KI)`

Part 4: Jamtis Balance Recovery

Jamtis Keys



Jamtis Wallet Tiers

- **Master**: view full balance, spend $\{k_{vb}, k_m\}$
 - normal wallet, hw wallet
- **View All**: view full balance $\{k_{vb}, k_m U\}$
 - view-only wallet, hw/cold wallet hot interface
- **View Received**: view non-change enotes (amount, address index) + make addresses $\{s_{ga}, k_{fr}, k_{ua}, k_{vb} X + k_m U\}$
 - payment validation
- **Find Received**: pre-process enotes to speed up **View All** scanning $\{k_{fr}\}$
 - third-party scanning service, private scanning server
- **Generate Address**: make any random address $\{s_{ga}, k_{ua} G, k_{fr} k_{ua} G, k_{vb} X + k_m U\}$

Jamtis Addresses

- **Address Index Modifiers:** for address index j
 - Spend Key Extension: $k^j_x = Hn[s_{ga}]("..extension.." || j)$
 - Address Privkey: $k^j_a = Hn[s_{ga}]("..address.." || j)$
- **Address Contents:** for address index j
 - Spend Key: $K_1 = k^j_x X + k_{vb} X + k_m U$
 - DH Pubkey: $K_2 = k^j_a k_{fr} k_{ua} G$
 - DH Base Pubkey: $K_3 = k^j_a k_{ua} G$
 - Address Index Tag: $addr_tag = cipher[s_{ct}](j || MAC)$

Jamtis Normal Enotes

- **Enote Construction:** given $\{K_1, K_2, K_3, \text{addr_tag}\}, a, \text{input_context}\}$
 - **Enote Ephemeral Pubkey** ($r = \text{gen_scalar}()$): $K_e = r K_3$
 - **Diffie-Hellman Derivation:** $K_d = 8 r K_2$
 - **Sender-Receiver Secret:** $q = H32(K_d || K_e || \text{input_context})$
 - **Amount Commitment:** $C = Hn(q || 8 r G) G + a H$
 - **Encoded Amount:** $a_{\text{enc}} = a ^ H8(q || 8 r G)$
 - **One-Time Address:** $K_o = Hn(q || C) X + K_1$
 - **Encrypted Address Tag:** $\text{addr_tag_enc} = \text{addr_tag} ^ Hx(q || K_o)$
 - **View Tag:** $\text{view_tag} = H1(K_d || K_o)$
- **Enote:** $\{K_o, C, a_{\text{enc}}, \text{addr_tag_enc}, \text{view_tag}\}, K_e\}$

Jamtis SelfSend Enotes

- **Enote Construction:** $\{\{K_1, K_2, K_3, \text{addr_tag}\}, a, \text{input_context}, k_{\text{vb}}\}$
 - **Enote Ephemeral Pubkey** ($r = \text{gen_scalar}()$): $K_e = r K_3$
 - **Sender-Receiver Secret***: for desired [selfsend type]
 - $q = \text{H32}[k_{\text{vb}}]("\text{..}[\text{selfsend type}]\text{..}" \parallel K_e \parallel \text{input_context})$
 - **Amount Commitment***: $C = \text{Hn}(q) G + a H$
 - **Encoded Amount***: $a_{\text{enc}} = a \wedge \text{H8}(q)$
 - **One-Time Address**: $K_o = \text{Hn}(q \parallel C) X + K_1$
 - **Encrypted Address Tag***: $\text{addr_tag_enc} = \{j \parallel \text{MAC}\} \wedge \text{Hx}(q \parallel K_o)$
 - **View Tag** ($K_d = 8 r K_2$): $\text{view_tag} = \text{H1}(K_d \parallel K_o)$
- **Enote:** $\{\{K_o, C, a_{\text{enc}}, \text{addr_tag_enc}, \text{view_tag}\}, K_e\}$

Jamtis Key Images

- Key Image: $KI = (1 / (Hn(q || C) + k^j_x + k_{vb})) * k_m U$
- View Key Image: $\{k_{vb}, k_m U\}$ and $\{C, K_e, j, input_context\}$
- Sign Key Image: $\{k_{vb}, k_m\}$ and $\{C, K_e, j, input_context\}$

Two-Output Optimization

- **Typical Transaction (>90%)**: destination + change/dummy
- **Optimization**: 2-output txs have only 1 enote ephemeral pubkey K_e
 - **Scanning**: compute $8 k_{fr} K_e$ once for both enotes in 2-out txs
- **How**: 2-out tx must have 1 self-send/dummy (dummy can be self-send)
 - **Self-Send Key Derivation**: k_{fr} is known, so reuse K_{e_other}
 - $K_d = 8 k_{fr} [K_{e_other}] = 8 k_{fr} [r K_{3_other}]$
 - **Duplication**
 - **Amounts**: if two same-type self-send outputs (note: amount DL)
 - $C = H_n(H_{32}[k_{vb}]("..[self type].." || K_e || in_ctx)) G + a H$
 - **Tags**: depend on K_o (no duplicate K_o in tx allowed)

- **Requirement:** all txs have ≥ 2 outputs and ≥ 1 self-send output
 - Locate key images: only look in txs with 1+ view tag matches

- **Rules:** ignoring error cases


Rules: Additional

Output

- 1 self-send, 0 change: + dummy (no K_e)
- 1 normal, 0 change: + self-send dummy (reuse K_e)
- 1 non-change self-send/normal, >0 change: + change (reuse K_e)
- ≥ 2 normals, 0 change: + self-send dummy
- ≥ 2 normals, >0 change: + change
- 2 outputs, 0 change: + dummy (if no shared K_e)
- ≥ 2 normals and self-sends, >0 change: + change
- >2 normals and self-sends, 0 change: + nothing

Part 5: Enote Scanning Workflow

Find-Received Scanning

- Scan Info: $\{\{K_o, C, a_enc, view_tag, addr_tag_enc\}, K_e, input_context\}$
 - Find-Received Scan: given $\{k_fr\}$
 - Nominal DH Derivation: $K_d' = g^{k_fr} K_e$ (reuse if K_e duplicated)
 - Nominal View Tag: $view_tag' = H_1(K_d' || K_o)$
 - If $view_tag' \neq view_tag$ then ABORT
 - Nominal Sender-Receiver Secret (normal):
 - $q' = H_{32}(K_d' || K_e || input_context)$
 - Nominal Address Tag: $addr_tag' = addr_tag_enc \oplus H_x(q' || K_o)$
 - Result: $\{\{..enote..\}, K_e, input_context, addr_tag'\}$ 
- Non-local scan: match saved tags & notify user.

View-Balance Scanning (Normal 1)

- Find-Received Scan Info: $\{\{K_o, C, a_enc, view_tag, addr_tag_enc\}, K_e, input_context, addr_tag'\}$
- View-Balance Scan: given $\{k_vb, k_m \cup\}$ (derive k_ua, k_fr, s_ga, s_ct)
 - Nominal Address Index: $j' = \text{decipher}[s_ct](addr_tag')$
 - If decipher fails (invalid MAC) then ABORT
 - Nominal Sender-Receiver Secret (normal): with $K_d' = 8 k_fr K_e$
 - $q' = H32(K_d' || K_e || input_context)$
 - Spend Key Extension: $k^j_x = Hn[s_ga]("..extension.." || j')$
 - If $K_o \neq (Hn(q' || C) + k^j_x + k_vb) \times k_m \cup$ then ABORT

View-Balance Scanning (Normal 2)

- Find-Received Scan Info: $\{\{K_o, C, a_{enc}, view_tag, addr_tag_enc\}, K_e, input_context, addr_tag'\} + \{j', q', k^j_x\}$

- Address Privkey: $k^j_a = Hn[s_ga]("../address.." || j')$
- Amount Recovery: $baked_key = 8 (1/(k^j_a k_{ua})) K_e = 8 r G$
 - Nominal Amount: $a' = a_{enc} \wedge H8(q' || baked_key)$

Payment
Validator:

- If $C \neq Hn(q' || baked_key) G + a' H$ then ABORT (Janus)

$\{s_ga, k_{fr}, k_{ua}, k_{vb}, X + k_m U\}$

Key Image: $KI = (1 / (Hn(q' || C) + k^j_x + k_{vb})) * k_m U$

- If KI appears on-chain (or in an off-chain context), then the enote is spent.

- Result: $\{\{..enote..\}, K_e, input_context, a, j, KI, [normal\ type]\}$

View-Balance Scanning (SelfSend 1)

- Find-Received Scan Info: $\{\{K_o, C, a_enc, view_tag, addr_tag_enc\}, K_e, input_context, addr_tag'\}$ (not used)}
- View-Balance Scan: given $\{k_vb, k_m U\}$ (derive k_fr, s_ga)
 - Nominal Sender-Receiver Secret (selfsend): test a [selfsend type]
 - $q' = H32[k_vb]("..[selfsend type].." || K_e || input_context)$
 - Nominal Address Index: $addr_tag_enc \wedge Hx(q' || K_o) \rightarrow \{j', MAC\}$
 - If raw decrypt fails (invalid MAC) then ABORT (or try new type)
 - Spend Key Extension: $k^{j_x} = Hn[s_ga]("..extension.." || j')$
 - If $K_o \neq (Hn(q' || C) + k^{j_x} + k_vb) X + k_m U$ then ABORT (or ..)

View-Balance Scanning (SelfSend 2)

- Find-Received Scan Info: $\{\{K_o, C, a_enc, view_tag, addr_tag_enc\}, K_e, input_context, addr_tag'\}$ (not used) + $\{j', q', k^j_x\}$
 - Address Privkey: $k^j_a = Hn[s_ga]("../address.." || j')$
 - Amount Recovery: no baked key (q' is function of k_vb)
 - Nominal Amount: $a' = a_enc \wedge H8(q')$
 - If $C \neq Hn(q') \wedge G + a' \wedge H$ then ABORT (Janus)
 - Key Image: $KI = (1 / (Hn(q' || C) + k^j_x + k_vb)) * k_m \wedge U$
 - If KI appears on-chain (or in an off-chain context), then the enote is spent.
 - Result: $\{\{..enote..\}, K_e, input_context, a, j, KI, [selfsend\ type]\}$

- **Enote Chunk**: from find-received scanning a set of txs (slow)
 - **Found Enotes**: $\text{vec}\langle\{\text{basic enote record, origin context}\}\rangle$
 - **Key Images**: from txs with found enotes $\text{vec}\langle\{\text{KI}, \text{spent context}\}\rangle$

- **Process the Chunk**

- Identify old enotes spent in chunk
 - Store tx id where spent
- Normal-scan the found enotes (fast)
 - Store tx id if spent in this chunk
- Loop until stored tx ids empty
 - Self-send-scan found enotes from stored tx ids (fast)
 - Store tx id if self-send enote is spent in this chunk

Chunk Processing

Ledger Scanning (reorg-safe)

- **Ledger Chunk**: enote chunk from block range **[a, b]**
 - **Prefix Block**: from block before range (for chunk contiguity)
 - **Block IDs**: **[prefix height, b]**
- **Scan**
 - **On-chain Loop**: process ledger chunk until no more chunks
 - Check if chunk is contiguous (starting from enote store)
 - If not, restart scanning (note: full-rescan vs partial-rescan)
 - **Unconfirmed**: process enote chunk for txs in tx pool
 - **On-chain Follow-up Loop**: in case unconfirmed chunk is stale
 - **Enote Store Update**: replace unconfirmed and records > alignment

End