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: sum(owned) - sum(owned + 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/RingCT Recap

Cryptonote/RingCT SubAddresses

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

CN/RingCT Enotes (To Subaddress)

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

CN/RingCT View Scanning (1)

- Scan Info: {{Ko, 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' != view_tag then ABORT
  - Nominal Sender-Receiver Secret: q' = Hn(K_d' || t)
  - Nominal Spend Key: K^i_s' = Ko - Hn(q') G
    - If K^i_s' not in {..., K^i_s, …} then ABORT
  - Payment ID: PID = PID_enc ^ Hn(K_d' || 0x8d) (ignore if 0x00s)

CN/RingCT View Scanning (2)

- Scan Info: {{Ko, C, a_enc, view_tag, PID_enc}, R_t, t} + {q', i, PID}
  - Amount Recovery
    - Nominal Amount: a' = a_enc ^ H8(".." || q')
      - If C != 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

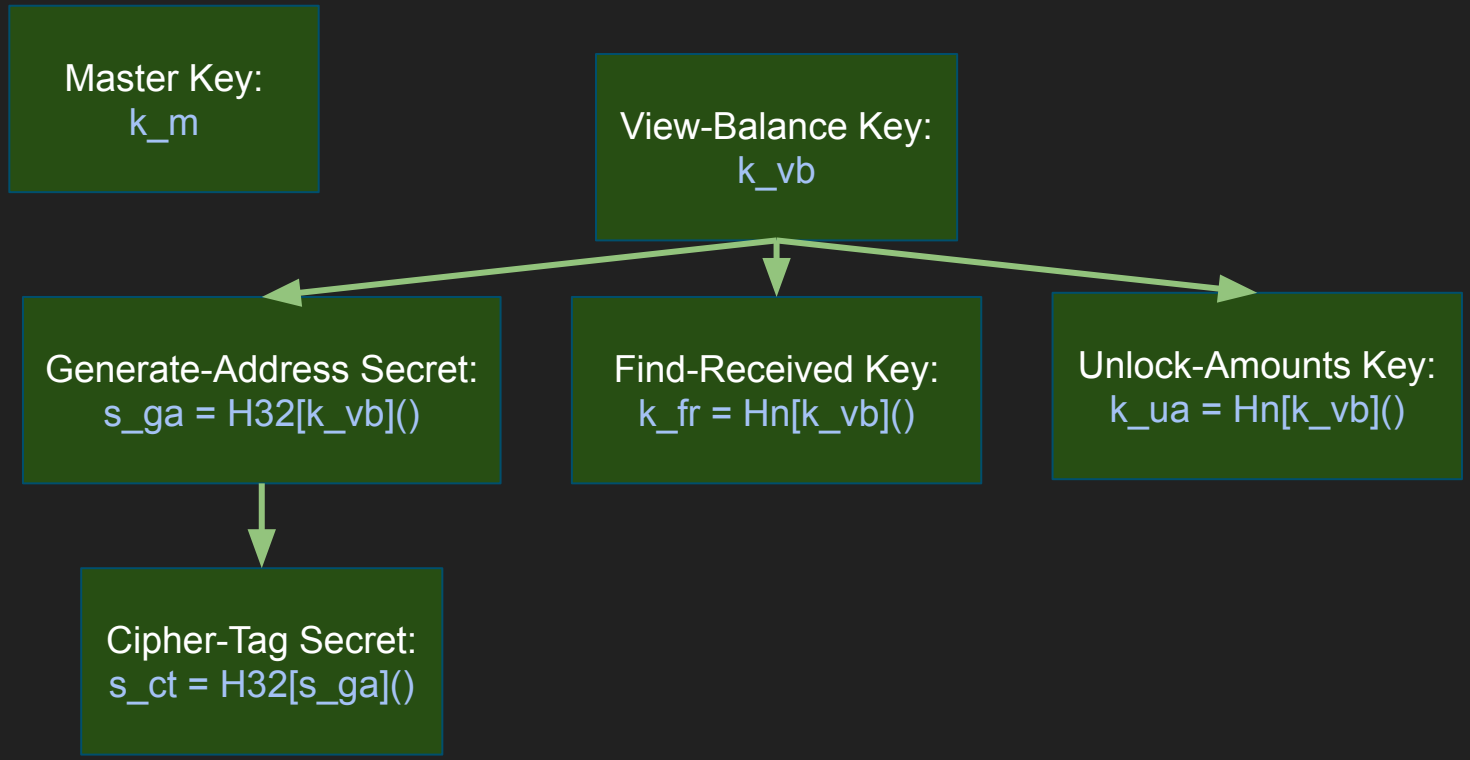
- Enote: {owner, amount} pair (for a tx output)
  - One-Time Address: Ko = k_a X + k_b U
  - Amount Commitment (amount 'a'): C = x G + a H
  - Encoded Amount: a_enc = enc(a)
- Enote Image: representation of an enote (for a tx input)
  - Masked Address: K" = t_k G + Ko
  - Masked Amount Commitment: C"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

Balance Recovery (outline)

- Recover Owned: examine an enote {Ko, C, a_enc}
  - Try Reproduce Address: Ko' ?= Ko
  - Decode Amount: a = dec(a_enc)
  - Try Reproduce Amount Commitment: C' ?= C
  - Store Key Image: KI = (k_b / k_a) U
- Identify Spent: examine an enote image {K", C", KI}
  - Compare Key Images: enote_store.has(KI)

Part 4: Jamtis Balance Recovery

Jamtis Keys


Master Key:
k_m
View-Balance Key:
k_vb
Generate-Address Secret:
s_ga = H32[k_vb]()
Find-Received Key:
k_fr = Hn[k_vb]()
Unlock-Amounts Key:
k_ua = Hn[k_vb]()
Cipher-Tag Secret:
s_ct = H32[s_ga]()

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 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, addr_tag}, a, input_context}
  - Enote Ephemeral Pubkey (r = 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 || input_context)
  - Amount Commitment: C = Hn(q || 8 r G) G + a H
  - Encoded Amount: a_enc = a ^ H8(q || 8 r G)
  - One-Time Address: Ko = Hn(q || C) X + K_1
  - Encrypted Address Tag: addr_tag_enc = addr_tag ^ Hx(q || Ko)
  - View Tag: view_tag = H1(K_d || Ko)
- Enote: {{Ko, C, a_enc, addr_tag_enc, view_tag}, K_e}

Jamtis SelfSend Enotes

- Enote Construction: {{K_1, K_2, K_3, addr_tag}, a, input_context, k_vb}
  - Enote Ephemeral Pubkey (r = gen_scalar()): K_e = r K_3
  - Sender-Receiver Secret*: for desired [selfsend type]
    - q = H32[k_vb]("..[selfsend type].." || K_e || input_context)
  - Amount Commitment*: C = Hn(q) G + a H
  - Encoded Amount*: a_enc = a ^ H8(q)
  - One-Time Address: Ko = Hn(q || C) X + K_1
  - Encrypted Address Tag*: addr_tag_enc = {j || MAC} ^ Hx(q || Ko)
  - View Tag (K_d = 8 r K_2): view_tag = H1(K_d || Ko)
- Enote: {{Ko, C, a_enc, addr_tag_enc, 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 = Hn(H32[k_vb]("..[self type].." || K_e || in_ctx)) G + a H
    - Tags: depend on Ko (no duplicate Ko in tx allowed)

Rules: Additional Output

- 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
  - 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: {{Ko, C, a_enc, view_tag, addr_tag_enc}, K_e, input_context}
- Find-Received Scan: given {k_fr}
  - Nominal DH Derivation: K_d' = 8 k_fr K_e (reuse if K_e duplicated)
  - Nominal View Tag: view_tag' = H1(K_d' || Ko)
    - If view_tag' != view_tag then ABORT
  - Nominal Sender-Receiver Secret (normal):
    - q' = H32(K_d' || K_e || input_context)
  - Nominal Address Tag: addr_tag' = addr_tag_enc ^ Hx(q' || Ko)
  - 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: {{Ko, C, a_enc, view_tag, addr_tag_enc}, K_e, input_context, addr_tag'}
- View-Balance Scan: given {k_vb, k_m U} (derive k_ua, k_fr, s_ga, s_ct)
  - Nominal Address Index: j' = 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 Ko != (Hn(q' || C) + k^j_x + k_vb) X + k_m U then ABORT

Find-Received + Generate-Address: {s_ga, k_fr, k_ua X, k_vb X + k_m U}

View-Balance Scanning (Normal 2)

- Find-Received Scan Info: {{Ko, 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 ^ H8(q' || baked_key)
      - If C != Hn(q' || baked_key) G + a' H then ABORT (Janus)

Payment Validator: {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: {{Ko, 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 ^ Hx(q' || Ko) -> {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 Ko != (Hn(q' || C) + k^j_x + k_vb) X + k_m U then ABORT (or ..)

View-Balance Scanning (SelfSend 2)

- Find-Received Scan Info: {{Ko, 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 ^ H8(q')
      - If C != Hn(q') G + a' H then ABORT (Janus)
  - 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, [selfsend type]}

Chunk Processing

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

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