Borromean Ring Signatures

Sammy, Hao Xu

2017-09-21

1 Concrete Algorithm

All the works listed in this section is from [MP16].

1.1 Signing

Suppose a signer has a collection of verification keys $P_{i,j}$ for $0 \le i \le n-1$, $0 \le j \le m_i - 1$, and wants to create a signature of knowledge of the n keys $\{P_{i,j_i^*}\}_{i=0}^{n-1}$ where the j_i^*'s are some fixed and unknown (to a verifier) indices. Denote the secret key to P_{i,j_i^*} by x_i. He acts as follows:

1. Compute M as the hash of the message to be signed and the set of verification keys.
2. For each $0 \le i \le n-1$:
   (a) Choose a scalar k_i uniformly at random.
   (b) Set $L_{i,j^*} = k_i G$.
   (c) For j such that $j_i^* + 1 \le j \le m_i - 1$ choose $s_{i,j}$ at random and compute

$$e_{i,j} = H(M \| L_{i,j-1} \| i \| j-1) \tag{1}$$
$$L_{i,j} = s_{i,j}G - e_{i,j}P_{i,j} \tag{2}$$

3. Finally, set
$$e_{i,0} = H(L_{0,m_0-1} \| \cdots \| L_{n-1,m_{n-1}-1})$$
   That is, $e_{i,0}$ commits to several (s, P) pairs, one from each ring.
4. For each $0 \le i \le n-1$:
   (a) Let $L_{i,-1} = L_{i,m_i-1}$
   (b) For j such that $0 \le j \le j_i^* - 1$ choose $s_{i,j}$ at random and compute

$$L_{i,j} = s_{i,j}G - e_{i,j}P_{i,j} \tag{3}$$
$$e_{i,j+1} = H(M \| L_{i,j} \| i \| j) \tag{4}$$

   Note that this calculation is identical to the one in Step 2c.
   (c) To wrap around making $L_{i,j_i^*} = s_{i,j}G - e_{i,j_i^*}P_{i,j_i^*}$, we should set

$$s_{i,j_i^*} = k_i + x_i e_{i,j_i^*} \tag{5}$$

The resulting signature on m consists of

$$\sigma = \{e_0, s_{i,j} \mid 0 \le i \le n-1, 0 \le j \le m_i - 1\} \tag{6}$$

where e_0 means any of $e_{i,0}$. We should publish

$$\{M, \{P_{i,j}\}, \sigma \mid 0 \le i \le n-1, 0 \le j \le m_i - 1\} \tag{7}$$

1.2 Verification

Since verification does not depend on which specific keys are known, it avoids the "two-phase" structure of signing and is therefore much simpler.

We assume we have a message m, a collection $\{P_{i,j}\}$ of verification keys whose indices range as before, and a signature σ whose notation is the same as before. The verifier acts as follows:

1. Compute M as the hash of the message to be signed and the set of verification keys.
2. For each $0 \le i \le n-1$,
   (a) For each $0 \le j \le m_i - 1$, compute
   $$L_{i,j} = s_{i,j}G - e_{i,j}P_{i,j} \tag{8}$$
   $$e_{i,j+1} = H(M\|L_{i,j}\|i\|j) \tag{9}$$
   (As before, we always take $e_{i,0}$ to be e_0.)
3. Compute
   $$e'_0 = H(M\|L_{0,m_0-1}\|\cdots\|L_{n-1,m_{n-1}-1}) \tag{10}$$
   and return 1 iff $e'_0 \stackrel{?}{=} e_0$.

A visualization of the whole scheme is as Figure 1

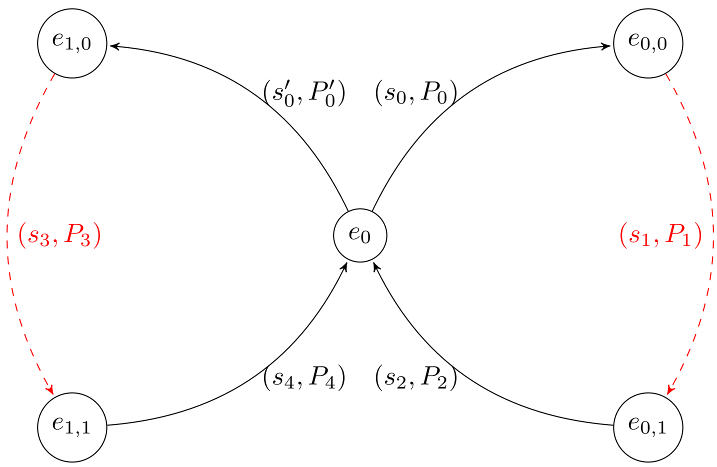

Figure 1: A Borromean ring signature for $(P_0|P_1|P_2)\&(P'_0|P_3|P_4)$

2 Implementation in Monero

Summarized from the official codebase [SML] as Algorithm 1 and 2, where it commits on empty hash digest M and include no indices (i, j) in calculating $e_{i,j}$.

References

[MP16] Gregory Maxwell and Andrew Poelstra. *Borromean Ring Signatures*. 2016. URL: https://github.com/ElementsProject/borromean-signatures-writeup.

[SML] Noether Shen, Adam Mackenzie, and The Monero Lab. *monero*. URL: https://github.com/monero-project/monero.

Algorithm 1: The Implementation of Signing for Borromean Ring Signatures in Monero

input : $\mathbf{x}$: set of secret keys $\{x_i\}_{i=1}^{63}$
input : $\{\mathbf{P}_i\}$: set of pubkey rings, where $\{\mathbf{P}_i = (P_{i,0}, P_{i,1})\}$
input : $\{j_i^*\}$: indices set, s.t., for each pair $(x_i, \mathbf{P}_i)$, $P_{i,j_i^*} = x_i \cdot G$ for $i = 0, 1, \ldots, 63$
output: $\sigma = (e_0, \{(s_{i,0}, s_{i,1})\})$: e_0 is a derived scalar, $s_{i,0}, s_{i,1} \in_u R$

for $i \leftarrow 0$ **to** 63 **do**
  $k_i \in_u R$
  $L_{i,j_i^*} \leftarrow k_i \cdot G$
  if $j_i^* = 0$ **then**
    $s_{i,1} \in_u R$
    $L_{i,1} \leftarrow s_{i,1} \cdot G + s_{i,1} \cdot P_{i,1}$
$e_{i,0} \leftarrow H_s(L_{0,1} \| L_{1,1} \| \cdots \| L_{63,1})$
for $i \leftarrow 0$ **to** 63 **do**
  if $j_i^* = 0$ **then**
    $s_{i,0} = k_i - x_i \cdot e_i$
  else
    $s_{i,0} \in_u R$
    $L_{i,0} \leftarrow H_s(s_{i,0} \cdot G + e_{i,0} \cdot P_{i,0})$
    $s_{i,1} \leftarrow k_i - x_j \cdot L_{i,0}$
$e_0 \leftarrow e_{i,0}$

Algorithm 2: The Implementation of Verification for Borromean Ring Signatures in Monero

input : $\{\mathbf{P}_i\}$, set of pubkey rings, where $\{\mathbf{P}_i = (P_{i,0}, P_{i,1})\}_{i=1}^{63}$
input : $\sigma = (e_0, \{(s_{i,0}, s_{i,1})\})$, a signature generated by Algorithm 1
output: true in case the verification is passed, and false otherwise

for $i \leftarrow 0$ **to** 63 **do**
  $L_{i,0} \leftarrow s_{i,0} \cdot G + e_0 \cdot P_{i,0}$
  $L_{i,1} \leftarrow s_{i,1} \cdot G + H_s(L_{i,0}) \cdot P_{i,1}$
$\hat{e_0} \leftarrow H_s(L_{0,1} \| L_{1,1} \| \cdots \| L_{63,1})$
return $\hat{e_0} = e_0$