Ring Signature

Generalized Schnorr Signature with Key Image

Kaspar Etter, November 2018

SHIFT CRYPTOSECURITY

Definition

Signing as a member of a group is known as a ring signature if no initial setup is required and the anonymity of the signer cannot be revoked (otherwise it is called a group signature).

- https://en.wikipedia.org/wiki/Ring_signature
- https://en.wikipedia.org/wiki/Group_signature

Motivation

What: Select other outputs with same amount, prove that you spend one of them without revealing which one. Of course, we need to prevent double spending (covered later).

CoinJoin requires interaction with others.

Ring signatures allow participants to mix their coins independently from each other.

Mixing improves the **anonymity** of all outputs, which is why Monero makes mixing **mandatory**.

Recap: Schnorr Signature

Signer

knows k so that $K = kG$

choose random $r < |G|$,
compute $R = rG$

$c = hash(K, R, m)$

compute $s = r - c * k$

share (c, s) or (R, s) as
signature for message m

Verifier

In case of (c, s), verify
$c = hash(K, sG + cK, m)$

In case of (R, s), verify
$R = sG + hash(K, R, m)K$

Combination of Several Equations

Choose random value r per hidden value and compute the element R for each equation.

Prove that you know k so that $K = kG$ and $I = kH$.

Compute $R_1 = rG$ and $R_2 = rH$, pass R_1 and R_2 to the verifier respectively the hash function, get c.

Verify that $R_1 = sG + cK$ and $R_2 = sH + cI$ with s being computed by the prover/signer as before.

Ring Signature (Split the Challenge)

Prove that you know a value k_1 so that $K_1 = k_1G$ or a value k_2 so that $K_2 = k_2G$ or a value k_3 so that …

If you knew all of them, you would choose a value r_i for each k_i, compute the $R_i = r_iG$ per equation, get the challenge c from the verifier or as hash of the message and all the R_is, solve $s_i = r_i - c * k_i$.

Since it is agreed that you only have to know one, you are allowed to split the challenge like $c = c_1$ xor c_2 xor c_3 xor … or $c = c_1 + c_2 + c_3 + … \% |G|$.

Ring Signature Example

- Prove $K_1 = k_1G$ or $K_2 = k_2G$. You only know k_1.
- Choose random values c_2, s_2 and r_1.
- Compute $R_1 = r_1G$, $R_2 = s_2G + c_2K_2$, $c = hash(R_1, R_2, m)$, $c_1 = c \text{ xor } c_2$ and $s_1 = r_1 - c_1 * k_1$.
- The signature then consists of c_1, c_2, s_1 and s_2.
- Verify $c_1 \text{ xor } c_2 = hash(s_1G + c_1K_1, s_2G + c_2K_2, m)$.
- Verifier does not learn whether you know k_1 or k_2.

Key Image (Avoid Double-Spending)

We need a hash function that hashes to EC points.

With $K = kG$, the key image I of k is $I = k * \mathrm{Hash}(K)$.

Prove that you know either

- k_1 so that $K_1 = k_1 G$ and $I = k_1 * \mathrm{Hash}(K_1)$
- k_2 so that $K_2 = k_2 G$ and $I = k_2 * \mathrm{Hash}(K_2)$

for the key image I provided by you.

Please note that it does not matter that there likely exists no k_i that satisfies both equations ($K_i = k_i G$ and $I = k_i * \mathrm{Hash}(K_i)$) for the case with the “fake” I.

Why not just derive I from fixed H?

The reason why the key images have to be derived from the hash of the respective public key instead of a separate but constant generator H like $I = kH$ is that Monero uses reusable addresses and whoever makes two transactions to the same recipient knows the linear correlation between the two public keys and could use this information to find the key images that are correlated.