

# Multiparty Schnorr Signatures

Omer Shlomovits

July 8, 2018

## 1 Simple Schnorr Signature

In this section we describe a multiparty Schnorr signature scheme for elliptic curves based on the work of Boneh et al. [1] section 5.1. The same protocol can be found also in the MuSig paper [2]. We first start by presenting Schnorr signature algorithm:

The public parameters are  $(\mathbb{G}, g, q)$  where  $\mathbb{G}$  is a group defined by elliptic curve,  $q$  is the order of the group and  $g$  is the generator of the group. To generate a key pair Alice chooses a private signing key  $x$  from the allowed set and the corresponding public key will be  $Y = x \cdot G$ . To sign a message  $m$  Alice chooses a random number  $k$  from the allowed set  $\mathbb{Z}_q$ . Let  $R = k \cdot G$ ,  $c = H(Y || R || m)$  where  $H$  is a cryptographic hash function  $H : \{0, 1\}^* \rightarrow \mathbb{Z}_q$ . Alice calculates  $s = k + xc$  and outputs the signature  $(R, s)$ .

Validation is checked simply by:

$$s \cdot G = R + c \cdot Y \quad (1)$$

This is a key-prefixed variant of the scheme where the public key is hashed together with  $R, m$ .

## 2 Multiparty Schnorr Signature

Multiparty Schnorr signature scheme, also called multi-signature scheme is a set of protocols between  $n$  parties such that they can jointly sign a message. The specific protocol we describe uses hash functions  $H_0, H_1, H_2 : \{0, 1\}^* \rightarrow \mathbb{Z}_q$ . These hash functions can be constructed from a single one using proper domain separation. The parameters are the same as in the case of single signer signature:  $(\mathbb{G}, g, G)$ .

**Key Generation:** Each party chooses  $x$  and computes  $Y = x \cdot G$ .

**Key Aggregation:** Compute  $apk \leftarrow \prod_{i=1}^n H_1(Y_i, \{Y_1, \dots, Y_n\}) \cdot Y_i$ .

**Signing:** Signing is an interactive three round protocol :

*Round 1:* This is a commitment round. Party  $i$  chooses  $r_i$  at random and compute

$R_i = r_i \cdot G$ . Let  $t_i \leftarrow H_2(R_i)$ . Send  $t_i$  to all other signers corresponding to  $Y_1, \dots, Y_n$  and wait to receive  $t_j = H_2(R_j)$  from all other signers  $j \neq i$ .

*Round 2:* Send  $R_i$  to all other signers corresponding to  $Y_1, \dots, Y_n$  and wait to receive  $R_j$  from all other signers  $j \neq i$ . Check that  $t_j = H_2(R_j)$  for all  $j = 1, \dots, n$ .

*Round 3:* each party:

1. Compute  $apk$  - Key Aggregation with public keys  $Y_1, \dots, Y_n$ .
2. Compute  $a_i = H_1(Y_i, \{Y_1, \dots, Y_n\})$ .
3. Compute  $\hat{R} \leftarrow \prod_{j=1}^n R_j$  and  $c \leftarrow H_0(\hat{R}, apk, m)$ .
4. Compute  $s_i \leftarrow r_i + c \cdot x_i \cdot a_i \pmod{q}$ .
5. Send  $s_i$  to all other signers and wait to receive  $s_j$  from other signers  $j \neq i$ .
6. Compute  $s \leftarrow \sum_{j=1}^n s_j$  and output  $(\hat{R}, s)$  as the final signature

Validation is the same as in simple Schnorr (eq. 1).

Conventions and preferred encodings of points and scalars can be found in [3]

## References

- [1] Compact Multi-Signatures for Smaller Blockchains, <https://eprint.iacr.org/2018/483.pdf>. Visited June 6th 2018.
- [2] Simple Schnorr Multi-Signatures with Applications to Bitcoin, <https://eprint.iacr.org/2018/068.pdf>. Visited July 7th 2018.
- [3] <https://github.com/sipa/bips/blob/bip-schnorr/bip-schnorr.mediawiki>