# The Schnorr Signature

Corso di dottorato di Blockchain e criptoeconomia
Andrea Gangemi, 18 June 2020

## Introduction

- The *Schnorr signature* is a digital signature algorithm which was described by Claus Schnorr in 1989.
- This signature scheme was patented until February 2008.
- The scheme is known for its simplicity, and its security is based on the supposed intractability of the Discrete Logarithm problem.

The classic scheme is based on the so-called *Schnorr groups*.

## Elliptic curve variant

The Schnorr's signature can also be computed on an elliptic curve.

- *KEY GENERATION*
  - Fix an elliptic curve $E$ on a finite field $\mathbb{F}_q$. Let $N$ be the order of the curve.
  - Fix a generator $G$ and a Hash Function $h$.
  - Every user chooses his *secret key* $d$, $0 < d < N$, and computes his *public key* $P = dG = (P_x, P_y)$.
- *SIGNING*
  - Let $M$ be the message.
  - The signer chooses an integer $k$, $1 < k < N$ and computes $R = kG = (R_x, R_y)$.
  - The signer computes $e = h(R_x || M)$.
  - The signer computes $s = k - de \mod N$.
  - The signature is the couple $(R_x, s)$.

## Elliptic curve variant

- *VERIFICATION*
  - The recipient computes $R_v = sG$.
  - The recipient computes $e = h(R_x||M)$.
  - If $R - R_v = eP$, the signature is valid.

This works because

$$R_v = sG = (k - de)G = kG - e(dG) = R - eP.$$

To recover the $y$ coordinate, we choose the one which is a quadratic residue (this works only if $q \equiv 3 \mod 4$).

## Related Key Attack

This scheme is vulnerable to the *Related Key Attack*.

- Starting from a valid signature $(R_x, s)$ for the public key $P$, an attacker can obtain a valid signature $(R_x, s + ae)$ for the public key $P - aG$.

In fact, observe that

$$R_v = (s + ae)G = (k - de + ae)G = kG - e(dG) + e(aG) = R - e(P - aG).$$

- This would render signatures insecure when keys are generated using additive tweaks.
- We can use *key-prefixed* Schnorr signatures to protect against this attack, i.e. computing $e = h(R_x||P_x||M)$.

## Schnorr advantages

- ECDSA is fast and secure, however it *does not allow a multisignature scheme*.
- ECDSA requires the *computation of an inverse and two multiplications*, which are not computationally inefficient but they are also not the best option.
- Schnorr signatures allow a multisig scheme and a batch verification feature in a natural and efficient way, thanks to its *linearity*.

## Batch Verification

We can check the correctness of every transaction in a block with a *single verification step*.

- Let $u$ be the number of transactions in a block.
- Let $P_1, \ldots, P_u$ be the public keys, $M_1, \ldots, M_u$ be the messages and $(R_{x_1}, s_1), \ldots, (R_{x_u}, s_u)$ be the signatures.
- Generate $u - 1$ random integers $a_2, \ldots, a_u \in [1, N - 1]$.
- Verify that

$$(s_1 + \sum_{i=2}^{u} a_i s_i)G = R_1 + \sum_{i=2}^{u} a_i R_i - (e_1 P_1 + \sum_{i=2}^{u} a_i e_i P_i, )$$

where $e_i = h(R_{x_i} || P_{x_i} || M_i)$.

## Multisignature scheme

- Schnorr's signatures allow for easy *n-of-n multisignatures schemes*.
- They can be useful in a lot of situations, for example if we possess a shared account where every user is equally important.
- *SIGNING*
  - Let's call $P_{sum} = P_1 + \cdots + P_n$ the sum of every public key involved $(d_1, \ldots, d_n$ are the associated private keys).
  - Every user chooses $k_i$ and computes $R_i = k_i G$. then they sum the points: $R_{sum} = R_1 + \cdots + R_n$.
  - Every user computes $s_i = k_i - d_i e$, where $e = h(R_{sum}||P_{sum}||M)$, then they sum: $s_{sum} = s_1 + \cdots + s_n$.
  - The signature is the couple $(R_{sum}, s_{sum})$.

## The Rogue Attack

- *VERIFICATION*
  - The signature is valid if $s_{sum}G = R_{sum} - eP_{sum}$.
- This scheme is vulnerable to the *Rogue Attack*.
- Suppose $n = 2$: Alice has the couple $(d_A, P_A)$, while Bob has $(d_B, P_B)$.
- Bob could lie to Alice, saying its public key is $P'_B = P_B - P_A$: then, $P_{sum} = P_A + P'_B = P_B$, so Bob can sign the message without the Alice's public key.

We need a method to utilize every single user public key, so that the rogue attack becomes impossible to realize.

# MuSig

The *MuSig scheme* solves the Rogue Attack. Let's see how it works (for $n$-of-$n$ schemes):

- *SIGNING*
  - Let $L = h(P_1||\dots||P_n)$. Every user computes the quantity $b_i = h(L||P_i)$.
  - Let $X = \sum_{i=1}^{n} b_i P_i$.
  - Every user chooses $k_i$ and computes $R_i = k_i G$. then they sum the points: $R_{sum} = R_1 + \dots + R_n$.
  - Every user computes $e_i = h(R_{sum}||X||M)b_i$.
  - Every user computes $s_i = k_i - d_i e_i$, then the aggregate is $s_{sum} = s_1 + \dots + s_n$.
  - The signature is the couple $(R_{sum}, s_{sum})$.

# MuSig

- *VERIFICATION*
  - Check that $s_{sum}G = R_{sum} - e'X$, where $e' = h(R_{sum}||X||M)$.

Let's see why this works (for $n = 2$):

$$s_{sum}G = (s_1 + s_2)G = (k_1 - d_1e_1 + k_2 - d_2e_2)G =$$

$$(k_1 + k_2)G - (d_1e_1 + d_2e_2)G = R_1 + R_2 - (e_1P_1 + e_2P_2) =$$

$$R_{sum} - (h(R_{sum}||X||M)b_1P_1 + h(R_{sum}||X||M)b_2P_2) =$$

$$R_{sum} - e'(b_1P_1 + b_2P_2) = R_{sum} - e'X.$$

The verification step can be performed without knowing every single public key: we just need the aggregate $X$.

# MuSig with a threshold

What if we want to make a *m-of-n multisignature scheme*? In some situations, we want to be able to sign a message without the presence of all $n$ private keys.

- This scheme is possible, but it is not efficient: we need to construct a Merkle tree of aggregated public keys for all combinations we can use.
- However, this number, which is trivially equal to $\binom{n}{m}$, grows exponentially in $n$.
- For this reason, a possible $m$-of-$n$ scheme is built on top of another scheme, known as *Pedersen Verifiable Secret Sharing Scheme* (VSS scheme).

## Schnorr: pro and cons

- Schnorr signature allows the creation of MultiSig schemes without increasing the computational complexity.
- In every aspect, it is more efficient than ECDSA.
- The signature is shorter than the ECDSA one: for this reason, more transactions can be inserted into a block, and this could reduce the transaction fees.
- The MuSig scheme needs 3 different rounds, and every round could potentially be attacked.
- $m$-of-$n$ schemes are not trivial: BLS signatures allow for more natural threshold schemes.