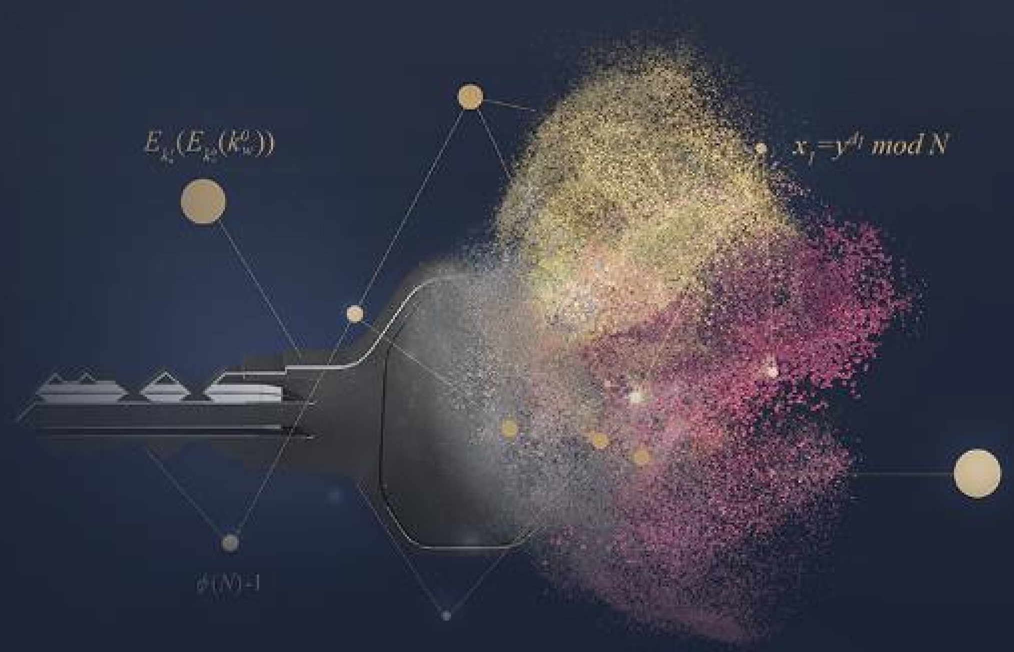
UNBOUND

Fast Secure Multiparty ECDSA

Samuel Ranellucci,

Cryptographer at Unbound Tech

11/03/2019



Agenda

1. Introduction
2. Building blocks
3. Fmult
4. Signing
5. Issues
6. Comparison

What kind of devices

Server

What kind of devices

Laptop

What kind of devices

Cell phone

Building block

1. Additively-homomorphic “encryption” scheme
   1. “Decryption” produces a commitment to a value

2. Multiplication protocol

3. Assumptions
   1. DDH
   2. Fmult
      1. Paillier or OT

ElGamal in the exponent

- Keygen
  - $k \in_R \mathbb{G}$
  - $P \leftarrow k \cdot G$

- Encrypt m
  - $r \in_R G$
  - Output $[\, r \cdot G,\ r \cdot P + m \cdot G\,]$

- “Decrypt” $[r \cdot G,\ r \cdot P + m \cdot G]$
  - $d \leftarrow (r \cdot P + m \cdot G) - (r \cdot G) \cdot k$

  Decrypts to $m \cdot G$

Fmult

Functionality that allows the following operations

Initialize

1. Stores input $(\mathbb{G}, G, q)$

Input

1. On input (input, sid)
2. Send random a_i to party P_i
3. $\text{secret}[\text{sid}] \leftarrow \sum a_i$

Mult

1. On input (mult, sid1, sid2)
2. $\text{c} \leftarrow \text{secret}[\text{sid1}] \cdot \text{secret}[\text{sid2}] \; mod \; q$
3. Send c to all parties

- Affine (sid1, sid2, x , y)
  - $\text{secret}\,[sid2] \leftarrow \text{secret}[sid1] \cdot x + y \;\; mod \;\; q$
- Element-out (sid)
  - $A \leftarrow \text{secret}[sid] \cdot G$

Fmult

1. Init (create ElGamal key)
    1. Private share $\Rightarrow d_i$
    2. Private key $\Rightarrow d \leftarrow \sum d_i$
    3. Public-key $\Rightarrow\ P \leftarrow d \cdot G$

1. Input (Private share a_i)
    1. Create ElGamal encryption of shares

Fmult

3. Affine (sid1, sid2, x, y)
    1. Linear combination of elements

3. Element-out (sid2)
    1. $A_i \leftarrow a_i \cdot G$
    2. $A \leftarrow \sum A_i$

Fmult

1. Private mult $(c \leftarrow a \cdot b)$
   1. $c \leftarrow a \cdot b$
   2. $c_1, \ldots, c_n \leftarrow$ Share(c)
   3. Private
   4. No correctness guarantee.
2. Verify correctness (see next slide)

Fmult

1. Verify correctness
   1. Construct encryption of $c = a \cdot b$
   2. Construct encryption of $c = a \cdot b$ via c_i
   3. Prove that the difference between these encryptions is zero.
   4. Prove that each share of the second encryption is consistent with c_i

Signing

- Fact: $k^{-1} \cdot (H(m) + r \cdot x) = k^{-1} \cdot \rho^{-1} \cdot \rho \cdot (H(m) + r \cdot x)$

- $\text{sign}_x(m; k)$
  - Fmult.Input $\Rightarrow$ random ρ, k
  - Fmult.output $\Rightarrow$ $R \leftarrow k \cdot G$
  - $(r, y) \leftarrow R$
  - Fmult.affine $\Rightarrow H(m) + r \cdot x$
  - Fmult.mult $\Rightarrow$ reveals $\rho \cdot k$
  - Compute $k^{-1} \cdot \rho^{-1}$
  - Fmult.mult $\Rightarrow \rho \cdot (H(m) + r \cdot x)$

Multiplication instantiation

1. OT-based solution
    1. Computational overhead : low
    2. Communication overhead : high

2. Paillier-based
    1. Computational overhead : high
    2. Communication overhead : low
    3. Suitable for mobiles.
    4. Expensive “Range” proof

Issues

1. Support BIP derivation

2. Proactive security
    1. Periodic refresh of shares
    2. We provide security as long as the adversary does not control a threshold of parties at any given time.

Issues

1. Failures do not require replacing keys

2. Arbitrary thresholds

Issues

Our protocol needs to work with smart phones

1. Multiplication protocol uses Paillier encryption to reduce communication

2. OT-based protocols too expensive

3. Low-round complexity

Security of our protocol

Our protocol is secure with simulation-based security under DDH.

Fast Multiparty Threshold ECDSA with Fast Trustless Setup

1. *Uses RSA to create multiplicative shares*
2. *Uses a conversion from multiplicative to additive sharing*
3. *Uses mult functionality*
4. *Base protocol requires expensive Range-proof (just like us)*
5. *Protocol improvement requires*
   1. *Game-based definition*
   2. *Strong RSA assumption*
   3. *Allows some leakage*

Secure Two-party Threshold ECDSA from ECDSA Assumptions

1. 2-out-of-n
2. OT-based instantiation
3. Convert additive to multiplicative shares
4. Uses a mult functionality
5. An improved multiplication functionality
6. Log(n) round complexity

Open question

1. One-sided OT extension
    1. OT extension where only one party is required to create large communication.

2. Better Range proof
    1. Lower computational, communication complexity.

UNBOUND

(MATH OVER MATTER)

Thank You