KU LEUVEN

Sharing the LUOV: post-quantum distributed signature schemes

Daniele Cozzo [1] Nigel Smart [12]

[1]KU Leuven

[2]Bristol University

August 20, 2019

Outline

Outline

The secret key $\mathfrak{sk}$ is shared among a set of parties

Any subset of t parties are able to sign a document

Any subset of less than t parties cannot do anything

Applications:

- Multiple signers are needed to produce a signature (e.g. witnesses)
- Key protection: attacker has to breack multiple devices in order to recover the key (useful e.g. in cryptocurrencies, see ByzCoin)

Secret key is shared according to some secret sharing scheme and then any operation in signing involving the secret key has to be done in MPC.

Theoretically, compiling a signature scheme to a multiparty scheme is always possible

In practice MPC is still limited in terms of number of parties and complexity of the circuit

Some operations are tricky to perfoorm in MPC

Expanding randomness and masking is usually done by computing an hash function over secret data

Hash function typically instantiated using SHAKE-256

Quite fast using state-of-the-art of MPC: 16ms to evaluate SHA-3 round function.

Problem is one has to repeat this many times.

Second round candidates

Dilithium	Lattice	A mix of linear operations (suitable for LSSS-based MPC) and non-linear operations (suitable for GC-based MPC) requires costly transferring between the two representations. We expect this to take around 12s to execute.
qTesla	Lattice	As above. We expect to take at least 16s to execute.
Falcon	Lattice	As above. We expect to take at least 6s to execute.
Picnic	MPC-in-H	Applying SHA-3 to obtain the necessary randomness in the views of the MPC parties.
SPHINCS+	Hash	Applying SHA-3 to obtain the data structures needed.
MQDSS	MQ	Applying SHA-3 to obtain the commitments.
GeMSS	MQ	Potential for threshold implementation, implementation is tricky due to need to extract polynomial roots via Berlekamp algorithm
Rainbow	MQ	Simple LSSS based MPC solution which requires 12 rounds of communication. We expect a signature can be generated in around three seconds
LUOV	MQ	Simple LSSS based MPC solution which requires 6 rounds of communication. We expect a signature can be generated in just over a second

Outline

Some steps use arithmetic over $\mathbb{F}_q$, $q = p, 2^r$.

- Use LSSS-based MPC over $\mathbb{F}_q$

Some steps use bit operations over $\mathbb{F}_q$

- Use GC-based MPC over $\mathbb{F}_2$

Means we need conversion

- daBit technique

Dilithium signature algorithm

1. $z = \perp$
2. **while** $z = \perp$ **do:**
   - Sample a short $y \in R_q^l$ with $\|y\|_\infty \leq \gamma_1$
   - $v = A \cdot y$
   - Let w be the topbits of v
   - $c = H(\mu \| w) \in R_q$
   - $z = y + c \cdot s_1$
   - Compute $v - c \cdot s_2$
   - **If** z or the lower bits of $v - c \cdot s_2$ are too big **then:**
     - $z = \perp$
3. **return** $\sigma = (z, c)$

qTesla signature algorithm

1. $z = \perp$
2. **while** $z = \perp$ **do:**
   - Sample a short $y \in R_q$ with $\|y\|_\infty \leq B$
   - $b = [a \cdot y]_M \in R_q$
   - $c = H(b \| G(\mu)) \in R_q$
   - $z = y + s \cdot c$
   - Compute $a \cdot y - e \cdot c$
   - **if** z is not short or $a \cdot y - e \cdot c$ is not well rounded **then:**
     - $z = \perp$
3. **return** $\sigma = (z, c)$

Falcon signature algorithm

1. $r \leftarrow \{0,1\}^{320}$
2. $c = H(r \| \mu)$
3. $t = (\mathsf{FFT}(c), \mathsf{FFT}(0)) \cdot \bar{B}^{-1}$
4. $z = \perp$
5. **while** $z = \perp$ **do:**
   - $z = \mathsf{ffSampling}_n(t, T)$
   - $s = (t - z) \cdot \bar{B}$
   - **if** z is not short **then:**
     - $z = \perp$
6. $(s_1, s_2) = \mathsf{FFT}^{-1}(s)$
7. $s = \mathsf{Compress}(s_2)$
8. **return** $\sigma = (r, s)$

Outline

Messages signed using many Winternitz signatures

Public keys authenticated using a dynamical structure called FORS (forest of random subsets)

Hash function of Winternitz scheme $\mathbf{F}$ and an hash function $\mathbf{PRF}$ for expanding the secret key into secret keys have to be computed on secret data.

Specification gives *exact* number of calls of these functions.

We estimated that even with the relevant simplifications, signing takes at least 85 minutes.

Outline

Prove knowledge of a preimage x of a function Φ given $y = \Phi(x)$

Prover simulates an MPC protocol to compute Φ over the shares of x.

Prover commits to status and transcripts of protocol

Prover sends commitments to the Verifier and randomly opens a non qualified subset of them (chosen by the verifier)

Verifier uses opened commitments to check that the protocol was executed correctly, accepts or rejects accordingly

Need to repeat T times to improve security

ZKPoK of a preimage of $y = f_k(x)$, where f is a OWF, x and y public and k secret to be proved.

Using FS transform, this is turned into a signature scheme with public key (x, y) and private key k.

Picnic signature algorithm

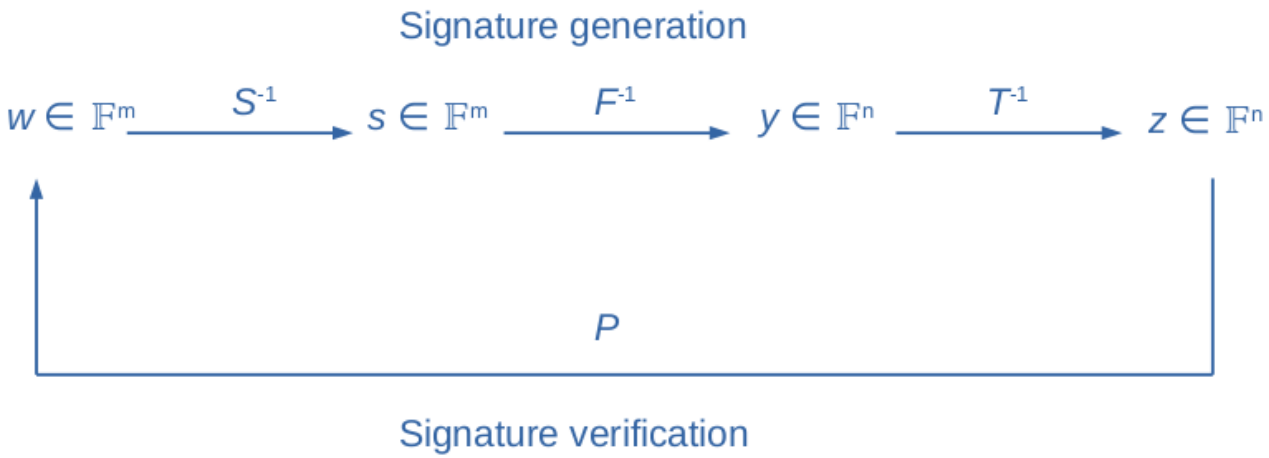
1. Generate $3 \cdot T$ secret seeds $\text{seed}_{i,j}$ for $i = 0, \ldots, T-1$ and $j = 0, 1, 2$
2. Using a KDF, exapnd $\text{seed}_{i,j}$ to a sequence of random tapes $\text{rand}_{i,j}$
3. For each round i, use three random tapes $\text{rand}_{i,j}$as the random input to a party P_j for an MPC protocol to evaluate the function $f_k(x)$
4. Commit to the resulting views, and hash them with a messageto obtain a set of challenges $e_0, \ldots, e_T \in \{0, 1, 2\}$
5. Reveal all seeds $\text{seed}_{i,j}$ except seed_{i,e_i}.

Step 2 alone takes 37 seconds

Outline

Key generation

Public key is a quadratic map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ Its factorization is secret key $\mathcal{S} : \mathbb{F}_q^m \to \mathbb{F}_q^m$, $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ and $\mathcal{T} : \mathbb{F}_q^n \to \mathbb{F}_q^n$



Partition n variables into $v = n - m$ vinegar variables $x_1, \ldots, x_v$ and m oil variables $x_{v+1}, \ldots, x_n$.

Central map is $\mathcal{F} = (f_1, \ldots, f_m)$ where

$$f_k(x_1, \ldots, x_n) = \sum_{i=1}^{v} \sum_{j=1}^{n} \alpha_{i,j}^k x_i x_j + \sum_{i=1}^{n} \beta_i^k x_i + \gamma_k$$

Use two fields. Maps $\mathcal{F}, \mathcal{T}$ and hence $\mathcal{P}$ are defined over $\mathbb{F}_2$.
The message $h(\mu)$ is defined over $\mathbb{F}_{2^r}$.

Solve the system over an extension of degree r

This maintains storage requirements at a reasonable level, but gives stronger security

The LUOV signature algorithm

1. $\mathbf{h} = H(\mu)$
2. **while** system $\mathcal{F}(\mathbf{v}||\mathbf{o}) = \mathbf{h}$ has no solution **do:**
   - $\mathbf{v} \leftarrow \mathbb{F}_{2^r}$
   - $\mathbf{A} \leftarrow$ BuildAugmentedMatrix(public_params, $\mathbf{T}$, $\mathbf{v}$, $\mathbf{h}$)
   - $\mathbf{o} \leftarrow$ GaussianElimination($\mathbf{A}$)
3. $\mathbf{s} = \begin{pmatrix} \mathbf{1}_v & -\mathbf{T} \\ \mathbf{0} & \mathbf{1}_m \end{pmatrix} \cdot \begin{pmatrix} \mathbf{v} \\ \mathbf{o} \end{pmatrix}$
4. **return** $\mathbf{s}$

The LUOV distributed signature algorithm

1. $\mathbf{h} = H(\mu)$
2. **while** system $\langle \mathcal{F} \rangle (\langle \mathbf{v} \rangle || \langle \mathbf{o} \rangle) = \mathbf{h}$ has no solution **do:**
   - $\langle \mathbf{v} \rangle \leftarrow \mathbb{F}_{2^r}$
   - $\langle \mathbf{A} \rangle \leftarrow$ BuildAugmentedMatrix(public_params, $\langle \mathbf{T} \rangle, \langle \mathbf{v} \rangle, \mathbf{h}$)
   - $\langle \mathbf{o} \rangle \leftarrow$ GaussianElimination($\langle \mathbf{A} \rangle$)
3. $\mathbf{o}$ = Open($\langle \mathbf{o} \rangle$)
4. $\langle \mathbf{s} \rangle = \begin{pmatrix} \mathbf{1}_v & -\langle \mathbf{T} \rangle \\ \mathbf{0} & \mathbf{1}_m \end{pmatrix} \cdot \begin{pmatrix} \langle \mathbf{v} \rangle \\ \mathbf{o} \end{pmatrix}$
5. $\mathbf{s} =$ Open($\langle \mathbf{s} \rangle$)
6. **return** $\mathbf{s}$

Inverting a secret matrix $\langle \mathbf{A} \rangle$

1. $\langle \mathbf{R} \rangle \leftarrow \mathbb{F}^{m \times m}$
2. $\langle \mathbf{C} \rangle = \langle \mathbf{A} \rangle \cdot \langle \mathbf{R} \rangle$
3. $\mathbf{C} = \text{Open}(\langle \mathbf{C} \rangle)$
4. $\mathbf{C}^{-1} = \text{GaussianElimination}(\mathbf{C})$
5. $\langle \mathbf{A}^{-1} \rangle = \langle \mathbf{R} \rangle \cdot \mathbf{C}^{-1}$

BuildAugmentedMatrix

1. $\mathbf{RHS} = \mathbf{h} - \mathbf{C} - \mathbf{L}\,(\mathbf{v}||\mathbf{0})^{\mathsf{T}}$
2. $\mathbf{LHS} = \mathbf{L}\begin{pmatrix} -\mathbf{T} \\ \mathbf{1}_m \end{pmatrix}$
3. **for** k from 1 to m **do:**
   - From public_params build $\mathbf{P}_{k,1}$
   - From public_params build $\mathbf{P}_{k,2}$
   - $\mathbf{RHS}[k] = \mathbf{RHS}[k] - \mathbf{v}^{\mathsf{T}} \cdot \mathbf{P}_{k,1} \cdot \mathbf{v}$
   - $\mathbf{F}_{k,2} = -\left(\mathbf{P}_{k,1} + \mathbf{P}_{k,1}^{\mathsf{T}}\right) \cdot \mathbf{T} + \mathbf{P}_{k,2}$
   - $\mathbf{LHS}[k] = \mathbf{LHS}[k] + \mathbf{v} \cdot \mathbf{F}_{k,2}$
4. **return** $\mathbf{LHS}||\mathbf{RHS}$

Distributed BuildAugmentedMatrix

1. $\langle \mathbf{RHS} \rangle = \mathbf{h} - \mathbf{C} - \mathbf{L}\left(\langle \mathbf{v} \rangle \| \mathbf{0}\right)^{\mathsf{T}}$
2. $\langle \mathbf{LHS} \rangle = \mathbf{L} \begin{pmatrix} -\langle \mathbf{T} \rangle \\ \mathbf{1}_m \end{pmatrix}$
3. **for** k from 1 to m **do:**
   - From public_params build $\mathbf{P}_{k,1}$
   - From public_params build $\mathbf{P}_{k,2}$
   - $\langle \mathbf{RHS} \rangle[k] = \langle \mathbf{RHS} \rangle[k] - \langle \mathbf{v} \rangle^{\mathsf{T}} \cdot \mathbf{P}_{k,1} \cdot \langle \mathbf{v} \rangle$
   - $\langle \mathbf{F} \rangle_{k,2} = -\left(\mathbf{P}_{k,1} + \mathbf{P}_{k,1}^{\mathsf{T}}\right) \cdot \langle \mathbf{T} \rangle + \mathbf{P}_{k,2}$
   - $\langle \mathbf{LHS} \rangle[k] = \langle \mathbf{LHS} \rangle[k] + \langle \mathbf{v} \rangle \cdot \langle \mathbf{F}_{k,2} \rangle$
4. **return** $\langle \mathbf{LHS} \rangle \| \langle \mathbf{RHS} \rangle$

Requires 6 rounds in total

Level-4 parameters $(r, m, v) = (8, 82, 323)$ require 2,756,430 secure multiplications

Arithmetic over $\mathbb{F}_{2^8}$ allows us to do 250,000 multiplications per sec, so signing takes 10 secs

Level-4 parameters $(r, m, v) = (64, 61, 302)$ give 1,372,866 secure multiplications

Arithmetic over $\mathbb{F}_{2^{64}}$ allows us to do 1,000,000 multiplications per sec

- We estimate signing takes 1.3 secs

First layer of vinegar variables $V_1 = \{1, \ldots, v_1\}$ and oil variables $O_1 = \{v_1 + 1, \ldots, v_2\}$. Second layer of vinegar variables $V_2 = \{1, \ldots, v_2\}$ and oil variables $O_2 = \{v_2 + 1, \ldots, n\}$

Central map is given by $\mathcal{F} = \underbrace{f^{v_1+1}, \ldots, f^{v_2}}_{layer1}, \underbrace{f^{v_2+1}, \ldots, f^{n}}_{layer2}$

$$f^{(k)} = \begin{cases} \sum_{i,j \in V_1, i \leq j} \alpha_{i,j}^{(k)} x_i x_j + \sum_{i \in V_1, j \in O_1} \beta_{i,j}^{(k)} x_i x_j & k = v_1 + 1, \ldots, v_2 \\ \sum_{i,j \in V_2, i \leq j} \alpha_{i,j}^{(k)} x_i x_j + \sum_{i \in V_2, j \in O_2} \beta_{i,j}^{(k)} x_i x_j & k = v_2 + 1, \ldots, n \end{cases}$$

Inversion of Rainbow central map, layer 1

1. $\hat{y}_1, \ldots, \hat{y}_{v_1} \leftarrow \mathbb{F}_q^{v_1}$
2. Substitute $\hat{y}_1, \ldots, \hat{y}_{v_1}$ into the polynomials $f^{v_1+1}, \ldots, f^{v_2}$.
3. Perform Gaussian elimination on the system

$$
\begin{aligned}
f^{(v_1+1)}\left(\hat{y}_1, \ldots, \hat{y}_{v_1}, y_{v_1+1}, \ldots, y_n\right) &= x_{v_1+1} \\
&\vdots \\
f^{(v_2)}\left(\hat{y}_1, \ldots, \hat{y}_{v_1}, y_{v_1+1}, \ldots, y_n\right) &= x_{v_2}
\end{aligned}
$$

to get the values of the second layer vinegar variables, say $\hat{y}_{v_1+1}, \ldots, \hat{y}_{v_2}$.

Inversion of Rainbow central map, layer 1

1. $\langle \hat{y} \rangle_1, \ldots, \langle \hat{y} \rangle_{v_1} \leftarrow \mathbb{F}_q^{v_1}$
2. Substitute $\langle \hat{y} \rangle_1, \ldots, \langle \hat{y} \rangle_{v_1}$ into the polynomials $\langle f \rangle^{v_1+1}, \ldots, \langle f \rangle^{v_2}$.
3. Perform Gaussian elimination on the system

$$\begin{aligned} \langle f \rangle^{(v_1+1)} \left(\langle \hat{y} \rangle_1, \ldots, \langle \hat{y} \rangle_{v_1}, y_{v_1+1}, \ldots, y_n \right) &= x_{v_1+1} \\ &\vdots \\ \langle f \rangle^{(v_2)} \left(\langle \hat{y} \rangle_1, \ldots, \langle \hat{y} \rangle_{v_1}, y_{v_1+1}, \ldots, y_n \right) &= x_{v_2} \end{aligned}$$

to get the values of the second layer vinegar variables, say $\langle \hat{y} \rangle_{v_1+1}, \ldots, \langle \hat{y} \rangle_{v_2}$.

Inversion of Rainbow central map, layer 2

1. Substitute $\hat{y}_1, \ldots, \hat{y}_{v_2}$ into the polynomials $f^{v_2+1}, \ldots, f^n$.
2. Perform Gaussian elimination on the system

$$\begin{aligned} f^{(v_2+1)}\left(\hat{y}_1, \ldots, \hat{y}_{v_2}, y_{v_2+1}, \ldots, y_n\right) &= x_{v_2+1} \\ &\vdots \\ f^{(n)}\left(\hat{y}_1, \ldots, \hat{y}_{v_2}, y_{v_2+1}, \ldots, y_n\right) &= x_n \end{aligned}$$

   to get the values of the second layer oil variables, say $\hat{y}_{v_2+1}, \ldots, \hat{y}_n$.
3. **Return** $\mathbf{y} = (\hat{y}_1, \ldots, \hat{y}_n)$.

Inversion of Rainbow central map, layer 2

1. Substitute $\langle \hat{y} \rangle_1, \ldots, \langle \hat{y} \rangle_{v_2}$ into the polynomials $\langle f \rangle^{v_2+1}, \ldots, \langle f \rangle^{n}$.
2. Perform Gaussian elimination on the system

$$\begin{aligned} \langle f \rangle^{(v_2+1)} \left(\langle \hat{y} \rangle_1, \ldots, \langle \hat{y} \rangle_{v_2}, y_{v_2+1}, \ldots, y_n \right) &= x_{v_2+1} \\ &\vdots \\ \langle f \rangle^{(n)} \left(\langle \hat{y} \rangle_1, \ldots, \langle \hat{y} \rangle_{v_2}, y_{v_2+1}, \ldots, y_n \right) &= x_n \end{aligned}$$

   to get the values of the second layer oil variables, say $\langle \hat{y} \rangle_{v_2+1}, \ldots, \langle \hat{y} \rangle_n$.
3. **Return** $\langle \mathbf{y} \rangle = (\langle \hat{y} \rangle_1, \ldots, \langle \hat{y} \rangle_n)$.

Level-3 parameters $(68, 36, 36)$ require 12 rounds of communication and 612216 secure multiplications

Arithmetic is over $\mathbb{F}_{2^8}$ thus we can perform 250,000 multiplications per sec.

- We estimate signing takes 3 secs

MQDSS is built from a 5-round ID scheme using FS

- SHAKE-256 is used a number of times to commit to secret data
- Commitment phase alone takes a lot

GeMSS is a potentially good candidate.

- Main problem is that signing involves executing several times the Berlekamp algorithm to find a root of univariate polynomials.

Conclusions

Post-quantum signatures are not easy to do in MPC

So far only the MQ-based ones seem *naturally* suitable for a threshold implementation

Rainbow better than LUOV in number of multiplications, however LUOV has half rounds and better memory storage.

Questions?