MAYO,

PRACTICAL POST-QUANTUM SIGNATURES AND A BIT MORE

Sofía Celi

Multivariate Quadratic (MQ) cryptography is based on the assumed hardness of finding a solution to a **system of multivariate quadratic equations** (over a **finite field)**. This problem is called the *MQ problem*.

Known to be *NP-complete*: seems to be hard on average for an extensive range of parameters:

The current record *mod 31* is solving a system of 22 equations in 22 variables.

$$\begin{aligned} x + 5x^2 + 3xy &= 4 \quad mod\ 7 \\ x^2 + 5xy + 5y^2 &= 1 \quad mod\ 7 \end{aligned}$$

General notation

- $n \in N$ be the number of variables
- $m \in N$ the number of equations
- For a prime power q, we denote by F_q the finite field of order q

Multivariate Quadratic Polynomials

A system of *m* (or *k*) equations (f_k) in *n* variables in $\mathbf{F}_q$

$$f_k(x_1, x_2, \ldots, x_n) = \sum_{1 \le i \le j \le n} a_{i,j}^{(k)} x_i x_j + \sum_{1 \le i \le n} b_i^{(k)} x_i + c^{(k)}$$

where:

- First term consists of the quadratic terms with $a_{i,j}$ as the coefficient
- Second term consists of the linear terms, with b_i as the coefficient
- *c* is the constant term
- All the coefficients are in $\mathbf{F}_q$

$$\begin{cases} y_1 &= p_1(x_1, \ldots, x_n) \\ y_2 &= p_2(x_1, \ldots, x_n) \\ &\vdots \\ y_m &= p_m(x_1, \ldots, x_n) \end{cases}$$

Multivariate quadratic (MQ) problem

$$\mathcal{P} : F_n \to F_m : x \to \mathcal{P}(x) = (p_1(x), \cdots, p_m(x))$$

The MQ problem asks:

given a multivariate quadratic map $P : F_q^{\,n} \to F_q^{\,m}$ over a finite field F_q and a target $\boldsymbol{t} \in \boldsymbol{F_q^{\,m}}$ *to find a solution* $\boldsymbol{s}$ *such that* $\boldsymbol{P(s) = t}$.

or

given $(y_1, \ldots, y_m)$ in $F_q^{\,m}$, find $(x_1, \ldots, x_n)$ in $F_q^{\,n}$ with $f_k(x_1, \ldots, x_n) = y_k$ for $1 \le k \le m$

if they exist.

Multivariate quadratic (MQ) assumption

- Can be a decisional problem: does the preimage (the input) exist?
- We are mostly interested in the **computational version**

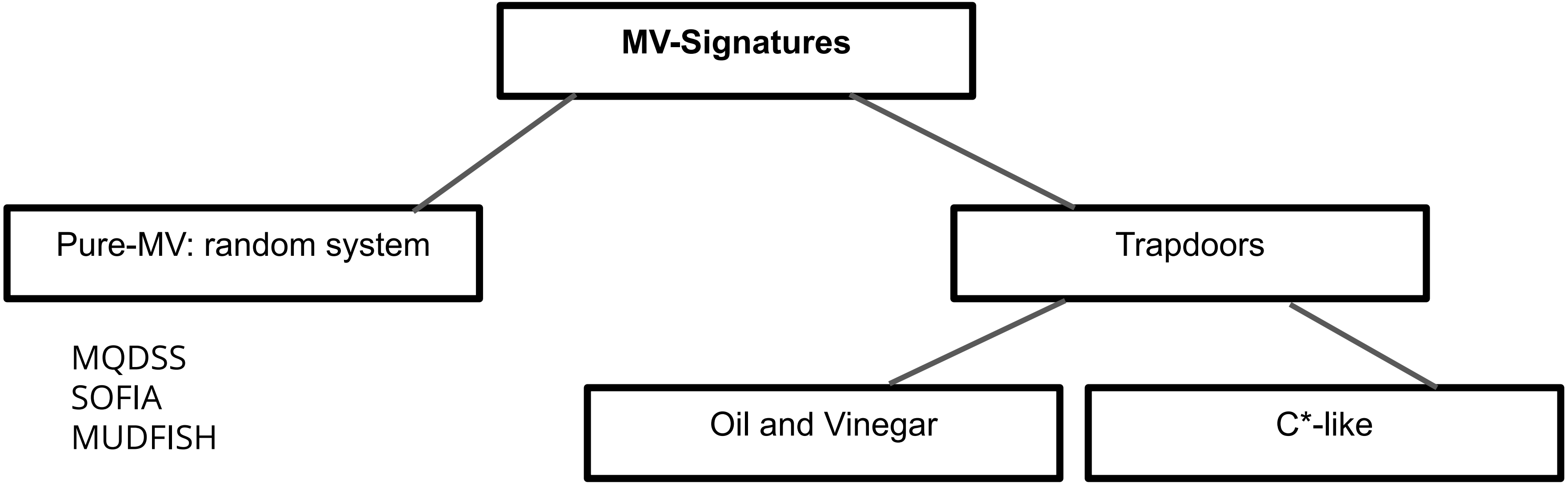
- It is known to be NP-hard [1] in its decisional form over any finite field F_q
- If $n > m(m + 1)$ (underdetermined) or $m > n(n - 1)/2$ (overdetermined) the problem can be solved in polynomial time [2], but when $n \sim m$ the problem is believed to be exponentially hard on average (even for quantum algorithms)
- Best algorithms to solve: *F4/F5* or XL that use a Gröbner-basis-like approach

---

[1] Garey, M. R. *Computers and intractability: A guide to the theory of np-completeness.* Revista Da Escola De Enfermagem Da USP 44, 2 (1979), 340.

[2] Thomae, E., and Wolf, C. Solving underdetermined systems of multivariate quadratic equations revisited. In PKC 2012 (May 2012), M. Fischlin, J. Buchmann, and M. Manulis, Eds., vol. 7293 of LNCS, Springer, Heidelberg, pp. 156–171.

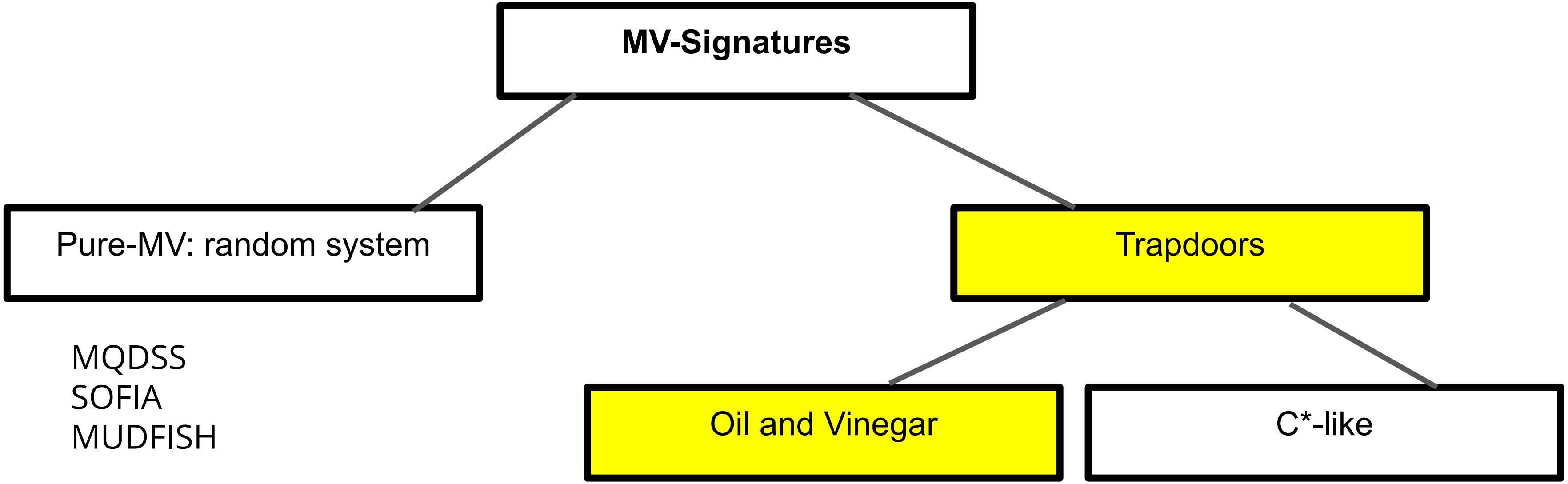
Signatures?



MQDSS
SOFIA
MUDFISH

Pure-MV: random system: slower, large signature sizes and smaller keys

Signatures?



Pure-MV: random system: slower, large signature sizes and smaller keys

Trapdoor-based

- Trapdoor-based:
  - Maps look random but have a “hidden structure” that allows one (a signer) to compute the pre-images
  - They are based in the *Hash-and-Sign with Retries* approach

PK: *P*

SK: trapdoor information

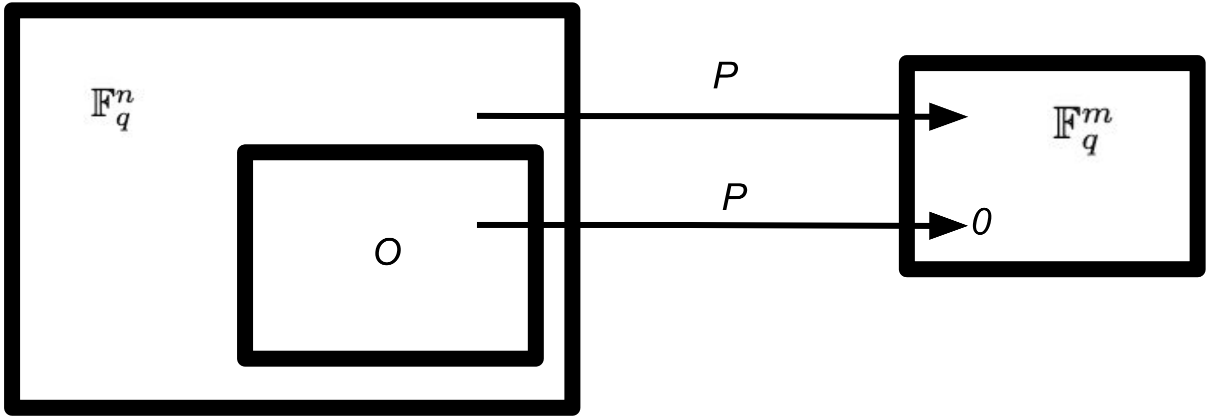
Signature: input *s* from *P(s) = H(m)*

```
proc sign(sk : skey, m : msg) : signature = {
  var i1, i2, z, salt, sol, s;

  (i1, i2) <- sk;
  z <$ i1;
  (* repeat until *)
  salt <$ dsalt;
  sol <@ H.get(m, salt);
  s <- i2 z sol;
  while (s = None) {
    salt <$ dsalt;
    sol <@ H.get(m, salt);
    s <- i2 z sol;
  }
  return (salt, oget s);
}
```

Trapdoor-based: *Unbalanced Oil and Vinegar*

Trapdoor: linear subspace $O \subset \mathbb{F}_q^n$ (oil space) of dimension m on which P vanishes (for every vector: $P(o) = 0\ \forall\ o \in O$)



***Unbalanced Oil and Vinegar*: yet another look**

Given this description of the trapdoor, we can simplify the description

We define the differential as:

$$\Delta(x) : \mathbb{F}_q^n \to \mathbb{F}_q^m : y \to \mathcal{P}(x+y) - \mathcal{P}(x) - \mathcal{P}(y) + \mathcal{P}(0)$$

which is symmetric and bilinear in both *x* and *y* (we eliminate the linear and constant terms). It shows the purely quadratic interactions.

Each component can be written as a bilinear form: $xGy^\top$, with a symmetric matrix G

"Selecting and Reducing Key Sizes for Multivariate Cryptography" by Albrecht Petzoldt

Unbalanced Oil and Vinegar: yet another look

Key generation:

$P^{(1)}$ → square upper triangular matrices, random
$P^{(2)}$ → rectangular matrices, random
Solve for $P^{(3)}$:

$$\mathbf{P}_i^{(3)} := \mathsf{Upper}\left(-\mathbf{O}^\intercal \mathbf{P}_i^{(1)} \mathbf{O} - \mathbf{O}^\intercal \mathbf{P}_i^{(2)}\right), \quad \forall i \in [m].$$

$$\mathbf{P}_i = \begin{bmatrix} \mathbf{P}_i^{(1)} & \mathbf{P}_i^{(2)} \\ \mathbf{0} & \mathbf{P}_i^{(3)} \end{bmatrix} \longrightarrow \begin{bmatrix} \mathbf{O}^\intercal & \mathbf{I}_m \end{bmatrix} \mathbf{P}_i \begin{bmatrix} \mathbf{O} \\ \mathbf{I}_m \end{bmatrix} = \mathbf{O}^\intercal \mathbf{P}_i^{(1)} \mathbf{O} + \mathbf{O}^\intercal \mathbf{P}_i^{(2)} + \mathbf{P}_i^{(3)} \in \mathbb{F}_q^{n \times n}$$

is skew symmetric

Unbalanced Oil and Vinegar: yet another look

Problem: $P(s) = t$

- Pick random $v \in F_q^{\,n}$ (the vinegar)
- Solve for $o \in O$ (the oil), such that $P(v, o) = t$:

$$\mathcal{P}(o+v) = \mathcal{P}(v) + \mathcal{P}(o) + \Delta_v(o) = t$$

$$\mathcal{P}(o+v) = \underbrace{\cancel{\mathcal{P}(v)} + \cancel{\mathcal{P}(o)}}_{a\ constant} + \Delta_v(o) = t$$

linear system of m equations in m variables.

If no solution, **re-try** for another v.

- Signature: $s = o + v$

Unbalanced Oil and Vinegar: yet another look

Signature:

The message is the solution to the system of equations (*t*)

In detail, how *P(x) = t (t = Hash(m||salt))*?

$$p'(\mathbf{x},\mathbf{y}) := p(\mathbf{x}+\mathbf{y}) - p(\mathbf{x}) - p(\mathbf{y})$$

$$\mathcal{P}(\mathbf{v}+\mathbf{o}) = \underbrace{\mathcal{P}(\mathbf{v})}_{\text{fixed by choice of } \mathbf{v}} + \underbrace{\mathcal{P}(\mathbf{o})}_{=0} + \underbrace{\mathcal{P}'(\mathbf{v},\mathbf{o})}_{\text{linear function of } \mathbf{o}} = \mathbf{t}\,.$$

Unbalanced Oil and Vinegar

- Unbalanced: $n > m$
- Good performance
- Good signature sizes
- Large public keys

Let's make it practical! MAYO

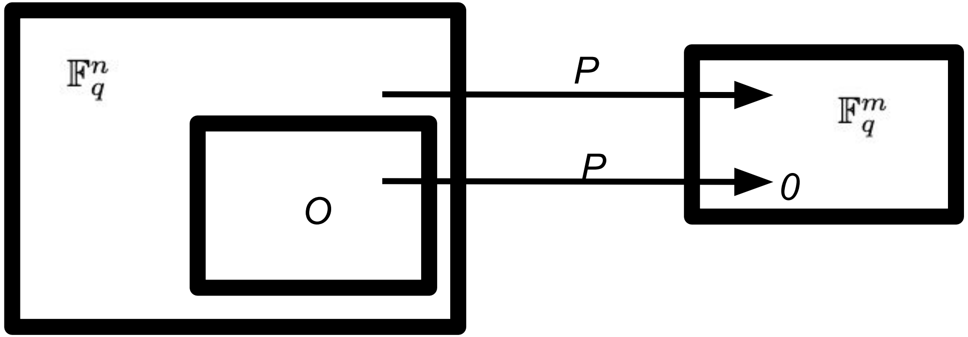
- Make the oil space smaller: *dim(O) < m*
- Vanishes on space dimension *o*, and it is "whipped-up"^{tm} to vanish on *ko*

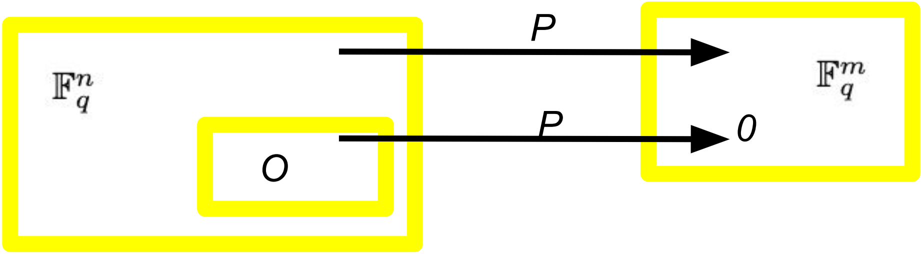
non-polar polar polar and non-polar

Mix the oils and vinegars to get something^{TM} add emulsifiers and you get MAYO^{TM} (kinda of)

polar and non-polar

MAYO





MAYO

- Make the oil space smaller: *dim(O)* < *m*
- We can use a smaller *n* → smaller parameters and faster computation

Why can't this be used as is?

Remember: $$\mathcal{P}(\mathbf{v}+\mathbf{o}) = \underbrace{\mathcal{P}(\mathbf{v})}_{\text{fixed by choice of } \mathbf{v}} + \underbrace{\mathcal{P}(\mathbf{o})}_{=0} + \underbrace{\mathcal{P}'(\mathbf{v},\mathbf{o})}_{\text{linear function of } \mathbf{o}} = \mathbf{t}\,.$$

This is a system of linear equations: *m* equations, *o* < *m* degrees of freedom → with high probability the system does not have solutions

Whipping up MAYO

Make the oil space smaller… and now whip it up (with a param *k*)!

$$\mathcal{P}(x) : \mathbb{F}_q^n \to \mathbb{F}_q^m \longrightarrow \mathcal{P}^*(x) : \mathbb{F}_q^{kn} \to \mathbb{F}_q^m$$

- Parameter *k* → the “scaler” (*ko* ≥ *m*):
  - *k = 1* and *o* = *m* → UOV signature
  - Larger *k*, reduces *o* to *ceil(m/k)*
- A public operation so that *P** vanishes on O^k
- How can we whip it up?

Whipping up MAYO

Easy way that "works":

$$\boxed{\mathcal{P}^*(x_1, \cdots, x_k) = \mathcal{P}(x_1) + \cdots + P(x_k)}$$

Dimension $o < m \rightarrow$ whipping up gives us $ko \geq m$

Not preimage resistant: for $k \geq 2$ and α in F_q, which is $\alpha^2 = -1$, and δ is at random:

$$\begin{aligned}\mathcal{P}^\star(\mathbf{x}_1, \ldots, \mathbf{x}_k) &= \mathcal{P}(\mathbf{x}_1) + \mathcal{P}(\alpha\mathbf{x}_1 + \delta) \\ &= \mathcal{P}(\mathbf{x}_1) + \mathcal{P}(\alpha\mathbf{x}_1) + \mathcal{P}(\delta) + \mathcal{P}'(\alpha\mathbf{x}_1, \delta) \\ &= \mathcal{P}(\delta) + \mathcal{P}'(\alpha\mathbf{x}_1, \delta),\end{aligned}$$

which is linear in x (there is an "additional oil space") ⇒ collapses into a linear map

Whipping up MAYO

$$\begin{aligned}\mathcal{P}^\star(\mathbf{x}_1,\ldots,\mathbf{x}_k) &= \mathcal{P}(\mathbf{x}_1)+\mathcal{P}(\alpha\mathbf{x}_1+\delta)\\ &= \boxed{\mathcal{P}(\mathbf{x}_1)+\mathcal{P}(\alpha\mathbf{x}_1)}+\mathcal{P}(\delta)+\mathcal{P}'(\alpha\mathbf{x}_1,\delta)\\ &= \mathcal{P}(\delta)+\mathcal{P}'(\alpha\mathbf{x}_1,\delta)\,,\end{aligned}$$

P is homogenous, $P(\alpha x_1) = -P(x_1)$

What remains is linear in *x* (there is an "additional oil space") ⇒ collapses into a linear map

---

You can use this attack to attack SNOVA: I*mproved Cryptanalysis of SNOVA* by Ward Beullens

Whipping up MAYO

Add “emulsifiers”:

$$\boxed{\mathcal{P}^\star(\mathbf{x}_1, \ldots, \mathbf{x}_k) = \mathbf{E}_1\mathcal{P}(\mathbf{x}_1) + \cdots + \mathbf{E}_k\mathcal{P}(\mathbf{x}_k)\,.}$$

Note: Avoids the previous “collapsing” attack:

$$\boxed{P(\alpha_i x_1) = \alpha_i^2 P(x_1)}$$

$$P(\alpha_i x_1 + \delta_i) = P(\alpha_i x_1) + P(\delta_i) + P'(\alpha_i x_1, \delta_i)$$

$$E_i P(x_i) = E_i P(\alpha_i x_1 + \delta_i) = E_i(\alpha_i^2 P(x_1) + \text{other terms})$$

$$P^*(x_1, \ldots, x_k) \approx E_1 P(x_1) + \alpha_2^2 E_2 P(x_1) + \cdots + \alpha_k^2 E_k P(x_1)$$

$$P^*(x_1, \ldots, x_k) \approx \left(E_1 + \sum_{i=2}^{k} \alpha_i^2 E_i\right) P(x_1)$$

Whipping up MAYO

Add “emulsifiers”:

$$\boxed{\mathcal{P}^\star(\mathbf{x}_1,\ldots,\mathbf{x}_k) = \mathbf{E}_1\mathcal{P}(\mathbf{x}_1) + \cdots + \mathbf{E}_k\mathcal{P}(\mathbf{x}_k)\,.}$$

But how to choose *Es*?

- Avoid the probability of any linear combination with rank lower than *n*
- Choose the *Es* from a set of q^m matrices such that any non-zero linear combination of these matrices has full rank → no extra “oil” spaces
- Make them public and part of the parameters

Whipping up MAYO

Add “emulsifiers”:

$$\boxed{\mathcal{P}^\star(\mathbf{x}_1, \ldots, \mathbf{x}_k) = \mathbf{E}_1\mathcal{P}(\mathbf{x}_1) + \cdots + \mathbf{E}_k\mathcal{P}(\mathbf{x}_k)\,.}$$

m-by-m invertible linear matrices E_i

- Small chance of extra oil spaces, but this is still **not preimage resistant**:
- P^* is now the sum of k functions with independent inputs, which reduces to the k-SUM problem:
  - The attacker constructs k lists of evaluations of $E_i(P(x))$ respectively, and searches for one value in each list such that their sum is t ⇒ Wagner’s k-tree algorithm ($q^{m/[\log k]}$) [6].

Expanding this attack to SNOVA

[8] David Wagner. A generalized birthday problem. In Moti Yung, editor, CRYPTO 2002, volume 2442 of LNCS, pages 288–303, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany.

Whipping up MAYO

Add invertible linear “emulsifiers”:

m-by-m matrices $E_{i,j}$

$$\boxed{\mathcal{P}^*(\mathbf{x}_1,\ldots,\mathbf{x}_k) := \sum_{i=1}^{k} \mathbf{E}_{ii}\mathcal{P}(\mathbf{x}_i) + \sum_{i=1}^{k}\sum_{j=i+1}^{k} \mathbf{E}_{ij}\mathcal{P}'(\mathbf{x}_i,\mathbf{x}_j)}$$

- No K-Tree attacks apply → presence of cross terms
- But are there attacks?
- We call this the *Multi-Target Whipped MQ problem*

$$\mathcal{P}^\star(\mathbf{v}+\mathbf{o}) = \sum_{i=1}^{k} \mathbf{E}_{ii}\mathcal{P}\left(\mathbf{v}_i+\mathbf{o}_i\right) + \sum_{1\le i<j\le k} \mathbf{E}_{ij}\mathcal{P}'\left(\mathbf{v}_i+\mathbf{o}_i, \mathbf{v}_j+\mathbf{o}_k\right)$$

Whipping up MAYO

We call this the *Multi-Target Whipped MQ problem*

$$\mathrm{Adv}^{\mathrm{MTWMQ}}_{\{\mathbf{E}_{ij}\},n,m,k,q}(\mathcal{A}) = \Pr\left[\sum_{i=1}^{k} \mathbf{E}_{ii}\mathcal{P}(\mathbf{s}_i) + \sum_{i=1}^{k}\sum_{j=i+1}^{k} \mathbf{E}_{ij}\mathcal{P}'(\mathbf{s}_i,\mathbf{s}_j) = \mathbf{t}_I \;\middle|\; \begin{array}{c} \mathcal{P} \xleftarrow{\$} \mathrm{MQ}_{n,m,q} \\ \{\mathbf{t}_i\} \xleftarrow{\$} \mathbb{F}_q^{m\times\mathbb{N}} \\ (I,\mathbf{s}_1,\ldots,\mathbf{s}_k) \leftarrow \mathcal{A}^{t_i}(\mathcal{P}) \end{array}\right]$$

Assumed hard. How hard, though?

- Oil-and-Vinegar maps P are indistinguishable from random MQ maps.
- Whipping up a random map P, results in a (multi-target) preimage resistant MQ map P^*
- My current interest

MAYO WITH NEW BITS

PQMAYO.COM

New representation of public key

- We use now a “nibble-sliced” representation
- Good for AVX2 shuffle-based arithmetic on “big” CPUs and table-lookup-based multiplication on embedded platforms
- Upcoming work in Jan. for Arm/NEON

Nibbling MAYO: Optimized Implementations for AVX2 and Cortex-M4 by Ward Beullens, Fabio Campos, Sofia Celi, Basil Hess and Matthias J.Kannwischer.

Whipping the Multivariate-based MAYO Signature Scheme using Hardware Platforms by Florian Hirner, Michael Streibl, Florian Krieger, Ahmet Can Mert, Sujoy Sinha Roy

New parameters

Improved method for solving underdetermined systems by Hashimoto (2023) reduced security margin by between 14 bits (MAYO1) and 2 bits (MAYO2)

	MAYO1	MAYO2	MAYO3	MAYO5
Security Level	1	1	3	5
(n,m,o,k)	(86,78,8,10)	(81,64,17,4)	(117, 107, 10, 11)	(153,141,12,12)
Signature size	454 B	186 B	676 B	958 B
Pk size	1420 B	4912 B	2959 B	5515 B
Restart Prob	2^{-12}	2^{-20}	2^{-16}	2^{-16}
Forgery Attacks	156	155*	222	295
Key Recovery	197	167	260	332

Security proof

with *Matthew Swann* and *Fraçoise Dupressoir: EUF-CMA Security of MAYO in the Random Oracle Model*

- Prove that the randomization *R* is optional → provided by the adversary
- Tightened bound on *EUF-CMA*

$$\mathsf{SHAKE256}(\mathsf{seed_{sk}}) \sim \mathcal{K}(\mathsf{seed})$$
$$\mathsf{SHAKE256}(\mathsf{M}) \sim \mathcal{G}(\mathsf{M})$$
$$\mathsf{SHAKE256}(\mathsf{M\_digest} \parallel \mathsf{R} \parallel \mathsf{seed_{sk}}) \sim \mathcal{H}(\mathsf{M\_digest}, \mathsf{R}, \mathsf{seed})$$
$$\mathsf{SHAKE256}(\mathsf{M\_digest} \parallel \mathsf{salt}) \sim \mathcal{I}(\mathsf{M\_digest}, \mathsf{salt})$$
$$\mathsf{SHAKE256}(\mathsf{M\_digest} \parallel \mathsf{salt} \parallel \mathsf{seed_{sk}} \parallel \mathsf{ctr}) \sim \mathcal{J}(\mathsf{M\_digest}, \mathsf{salt}, \mathsf{seed}, \mathsf{ctr})$$

Threshold MAYO

Thresholdizing UOV-based algorithms is *not that hard*:

- The crux is in the restart probability of the algorithm to solve systems of equations

Protocol 1: Π_{Solve}

The protocol is set in the $\mathcal{F}_{\mathsf{ABB}}$-hybrid. All the commands except for (solve) are forwarded directly to $\mathcal{F}_{\mathsf{ABB}}$.

On input $(\mathsf{solve}, [\![\mathbf{A}]\!], [\![\mathbf{b}]\!])$, where $\mathbf{A}$ has dimensions $s \times t$ and $[\![\mathbf{b}]\!]$ has dimension s, the parties proceed as follows:

1. Parties call $[\![\mathbf{R}]\!] \leftarrow \mathsf{rand}(\mathbb{F}_q^{s\times s})$ and $[\![\mathbf{S}]\!] \leftarrow \mathsf{rand}(\mathbb{F}_q^{t\times t})$.
2. Parties call $[\![\mathbf{A}\cdot\mathbf{S}]\!] \leftarrow [\![\mathbf{A}]\!]\cdot[\![\mathbf{S}]\!]$.
3. Parties call $[\![\mathbf{T}]\!] \leftarrow [\![\mathbf{R}]\!]\cdot[\![\mathbf{A}\cdot\mathbf{S}]\!]$.
4. Parties open $\mathbf{T} \leftarrow [\![\mathbf{T}]\!]$. If $r = \mathsf{rank}(\mathbf{T}) < s$ then the parties output $(\mathsf{rank\text{-}defect}, r)$.
5. Otherwise, let $\mathbf{T}^{-1} \in \mathbb{F}_q^{t\times s}$ be a right inverse of $\mathbf{T}$, that is, $\mathbf{T}\mathbf{T}^{-1} = \mathbf{I}_{s\times s}$. The parties call $[\![\mathbf{A}^{-1}]\!] \leftarrow [\![\mathbf{S}]\!]\cdot\mathbf{T}^{-1}\cdot[\![\mathbf{R}]\!]$. It can be checked that $\mathbf{A}^{-1} \in \mathbb{F}_q^{t\times s}$ satisfies $\mathbf{A}\cdot\mathbf{A}^{-1} = \mathbf{I}_{s\times s}$.
6. Let $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{t-s} \in \mathbb{F}_q^{t-s}$ be a basis for $\ker(\mathbf{T})$. The parties call $([\![z_1]\!], \ldots, [\![z_{t-s}]\!]) \leftarrow \mathsf{rand}(\mathbb{F}_q^{t-s})$.
7. Parties compute locally $[\![\mathbf{z}]\!] \leftarrow \sum_{i=1}^{t-s} [\![z_i]\!]\cdot\boldsymbol{\beta}_i$.
8. Parties call $[\![\mathbf{x}]\!] \leftarrow [\![\mathbf{A}^{-1}]\!]\cdot[\![\mathbf{b}]\!] + [\![\mathbf{S}]\!]\cdot[\![\mathbf{z}]\!]$
9. Output $[\![\mathbf{x}]\!]$.

Share the MAYO: thresholdizing MAYO by Sofia Celi, Daniel Escudero, and Guilhem Niot

Threshold MAYO

Thresholdizing UOV-based algorithms is *not that hard*:

- We extend for other threshold functionality as:
  - Distributed Key Generation (DKG)
  - Threshold verification
- We provide a “complete” matrix representation of UOV-based systems
- We have not define the security model or implementation
- Already making it more optimal with Giacomo Borin

Share the MAYO: thresholdizing MAYO by Sofia Celi, Daniel Escudero, and Guilhem Niot

Jasmin and Easycrypt for MAYO

With Fraçoise Dupressoir, Manuel Barbosa and Pierre-Yves Strub

```
proc sign(sk : skey, m : msg) : signature = {
  var i1, i2, z, salt, sol, s;

  (i1, i2) <- sk;
  z <$ i1;
  (* repeat until *)
  salt <$ dsalt;
  sol <@ H.get(m, salt);
  s <- i2 z sol;
  while (s = None) {
    salt <$ dsalt;
    sol <@ H.get(m, salt);
    s <- i2 z sol;
  }
  return (salt, oget s);
}
```

```
export
fn m_vec_mul_add_x(reg u64[m_legs] in acc) {
  inline int i;
  stack u64[m_legs * 2] mul_res;
  reg u64 tmp, t;
  reg u8 a;

  // Precompute multiplication results
  for i = 0 to m_legs * 2 {
    t = in[i];
    a = 0x2;
    tmp = gf16v_mul_u64(t, a);
    mul_res[i] = tmp;
  }

  // Now apply XOR using the precomputed results
  for i = 0 to m_legs * 2 {
    acc[i] ^= mul_res[i];
  }
}
```

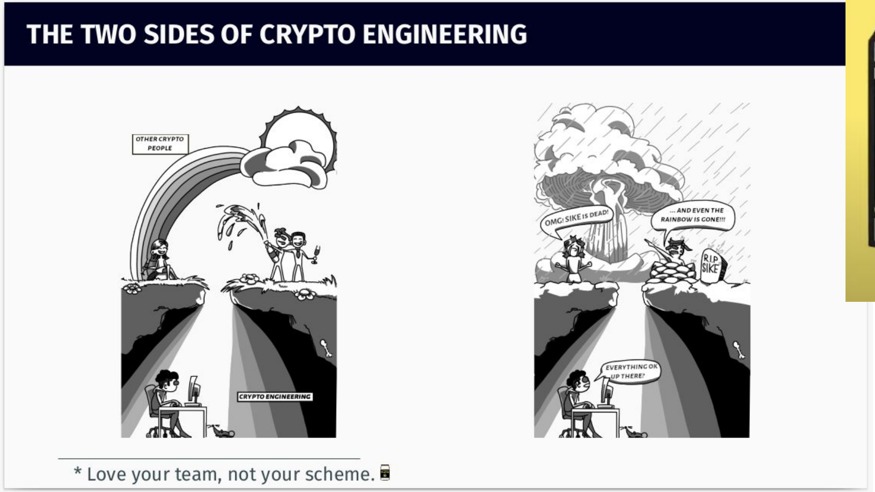
The amazing team

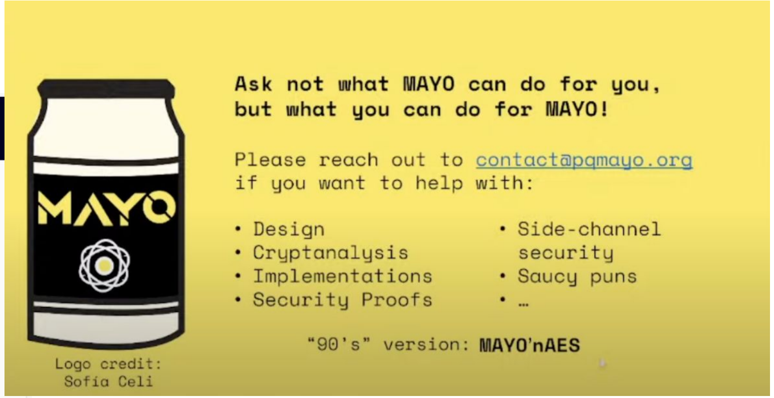
My fav papers/thesis

- “Improved cryptanalysis of UOV and Rainbow” by Ward Beullens
- “MAYO: Practical Post-Quantum Signatures from Oil-and-Vinegar Maps” by Ward Beullens
- “An Estimator for the Hardness of the MQ Problem” by Emanuele Bellini, Rusydi H. Makarim, Carlo Sanna, and Javier Verbel
- “Cryptanalysis of the Oil and Vinegar Signature Scheme” by Aviad Kipnis, Adi Shamir
- “A Study of the Security of Unbalanced Oil and Vinegar Signature Schemes“ by An Braeken, Christopher Wolf, and Bart Preneel
- “Selecting and Reducing Key Sizes for Multivariate Cryptography” by Albrecht Petzoldt

References

- For quick learning: https://github.com/PQCMayo/MAYO-sage
- Our C implementation: https://github.com/PQCMayo/MAYO-C
- Our website: https://pqmayo.org/





THANK YOU!

@claucece