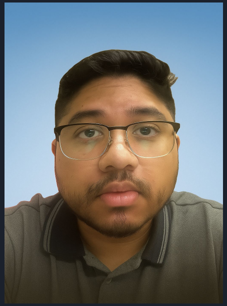
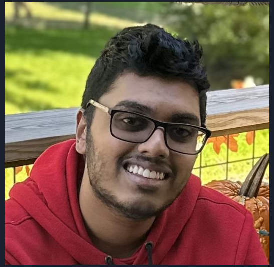
Meet the MAYO Team

Fahim Tanvir

Jude Marryshow



Mosharoof Hossain



Labib Nafi

Multivariate Cryptography & Oil–Vinegar (O&V)

Multivariate Quadratic (MQ) equations are **NP-hard**, even for quantum computers.

MQ cryptography shapes the basis for many post-quantum signature schemes.

The **Oil–Vinegar (O&V)** design divides variables into *oil* and *vinegar* groups.

Combining these variables creates a **trapdoor** that makes signing easy but inversion difficult.

Provides **fast signing and verification** without depending on factoring.

Limitations of Previous MQ Schemes

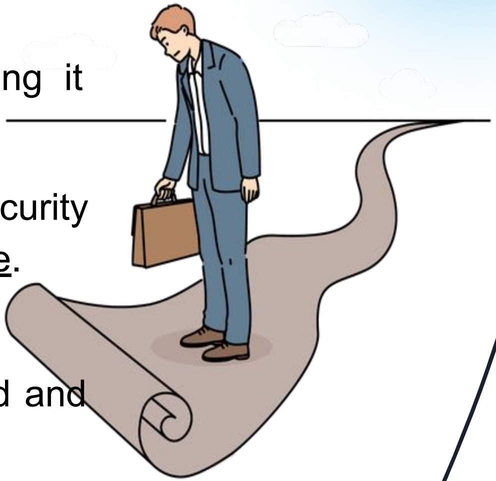
Prior multivariate schemes like Rainbow and UOV (Unbalanced Oil and Vinegar) faced several problems.

1. ***Large Key Sizes***

- The public key could be *several hundred* kilobytes long, making it unsuitable for devices with limited memory.
- Examples include smartphones, gaming consoles, and home security systems, which often run on low-power chips with restricted storage.

2. ***Structural Weaknesses***

- Researchers discovered patterns in how these systems generated and used their equations.
- Once analyzed, these patterns exposed parts of the private key, weakening the system's overall security.



Limitations of Previous MQ Schemes

Example - The Rainbow Attack

- Rainbow was broken in 2022 when researchers found math dependencies in its design.
- These dependencies helped them to reconstruct the private key using only the public key, showing that the scheme's inner structure wasn't concealed well enough.

Impact on Research

- These weaknesses showed that better multivariate schemes were needed. Schemes that could:
    a. Conceal structural patterns better
    b. Decrease key size while preserving security, and
    c. Remain efficient for present-day systems.

The Birth of MAYO

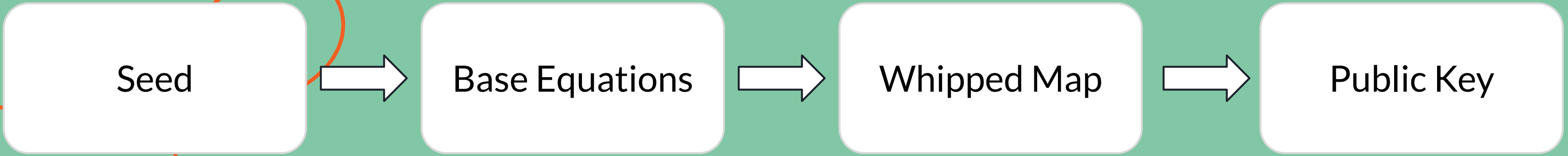
- Launches the “whipping” technique to compress keys.
- Uses a smaller oil space → less equations, smaller parameters.
- Keeps strong trapdoor structure for secure signatures.
- Balances compactness, effectiveness and stability.

MAYO = Applicable, lightweight post-quantum digital signature scheme.

Scheme Overview

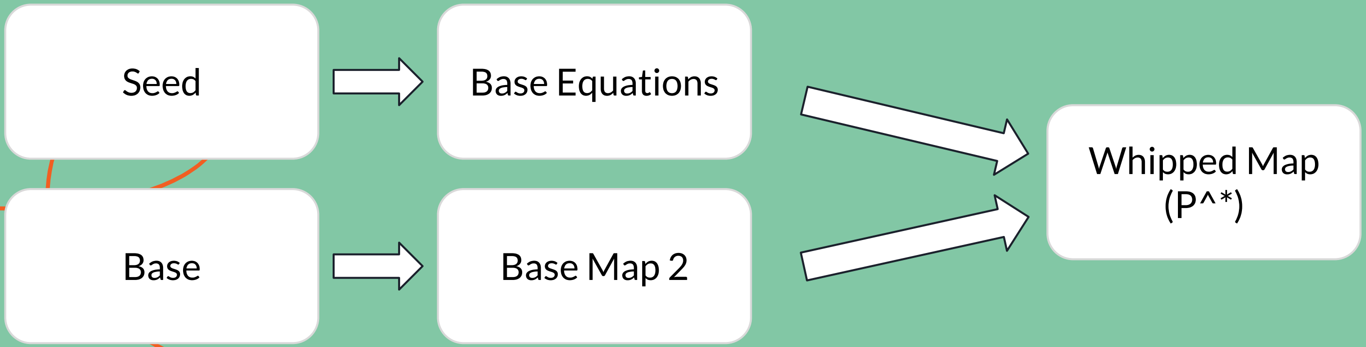
MAYO at a Glance

- MAYO is based on UOV (Unbalanced Oil and Vinegar), a well-known multivariate signature scheme.
- The main problem with UOV: large key sizes (especially the public key).
- MAYO solves this by introducing whipped maps, a structural trick that lets us generate the key deterministically from a small seed.
- This means we can rebuild the full key whenever needed, instead of storing the entire matrix.
- Result: drastically smaller key sizes without hurting performance or security.



The Whipped Map Construction

- Whipping = combining multiple base maps generated from the same seed.
- Each base map corresponds to an instance of the UOV structure.
- By using a *whipping parameter* **k**, we combine ***k*** instances of the map creating a composite public map P*.
- The result expands the **oil space dimension** (from *o* to *k times o*) and strengthens security.
- This composite mapping gives MAYO its **compactness** and **strong resistance** to algebraic attacks.



System Parameters and Components

- n = total number of variables (vinegar + oil)
- m = number of equations
- o = oil dimension (affects difficulty of solving system)
- k = whipping parameter (number of maps combined)
- MAYO’s compactness comes from generating everything from a short random seed, not storing large matrices.
- This structure ensures deterministic key generation — both sides can recreate the same key from the same seed.

How the Scheme Operates

KeyGen:

- Start from a random seed.
- Generate private affine transformations (S, T) and the public map P*.
- S: linear transformation applied before the central map.
- T: linear transformation applied after the central map.
- Output: Private key = seed + transformations, Public key = P*

Sign:

- Use private key to invert the central map efficiently In other words, they find a preimage under P* that corresponds to the given message hash.
- Produce a valid signature (vector) corresponding to the message hash.
- Unlike traditional schemes that rely on modular arithmetic of elliptic curves, this uses multivariate polynomial inversion as its core operation

How the Scheme Operates

Verify:

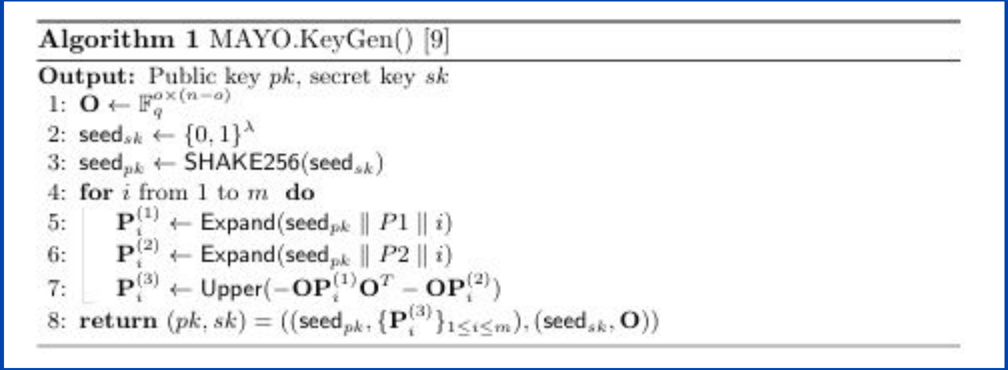
- Anyone can use the public map P* to check if the signature satisfies the polynomial equations
- The verifier checks whether the given signature actually satisfies the public equations: P*(x)=hash(message)
- Only the public information is used no access to the seed or transformations.

What’s public vs. private:

- Public: P*, message hash, signature
- Private: seed, affine transformations

Algorithms

Key Generation



```
Algorithm 1 MAYO.KeyGen() [9]
Output: Public key $pk$, secret key $sk$
1: $\mathbf{O} \leftarrow \mathbb{F}_q^{o \times (n-o)}$
2: $\mathsf{seed}_{sk} \leftarrow \{0,1\}^{\lambda}$
3: $\mathsf{seed}_{pk} \leftarrow \mathsf{SHAKE256}(\mathsf{seed}_{sk})$
4: for $i$ from 1 to $m$ do
5:     $\mathbf{P}_i^{(1)} \leftarrow \mathsf{Expand}(\mathsf{seed}_{pk} \parallel P1 \parallel i)$
6:     $\mathbf{P}_i^{(2)} \leftarrow \mathsf{Expand}(\mathsf{seed}_{pk} \parallel P2 \parallel i)$
7:     $\mathbf{P}_i^{(3)} \leftarrow \mathsf{Upper}(-\mathbf{O}\mathbf{P}_i^{(1)}\mathbf{O}^T - \mathbf{O}\mathbf{P}_i^{(2)})$
8: return $(pk, sk) = ((\mathsf{seed}_{pk}, \{\mathbf{P}_i^{(3)}\}_{1 \le i \le m}), (\mathsf{seed}_{sk}, \mathbf{O}))$
```

1. (Choose) Secret
Sample a random matrix ***O***. This is the core secret structure, used to simplify solving the system later.

2. (Generate) Seeds
Create *seedsk* (Secret) and derive *seedpk* (Public). Deterministically links the secret and public parts for consistency.

3. (Build Public Key)
Use *seedpk* to create matrices ***P_i(1)***, ***P_i(2)***. These are public constants defining the quadratic equation structure.

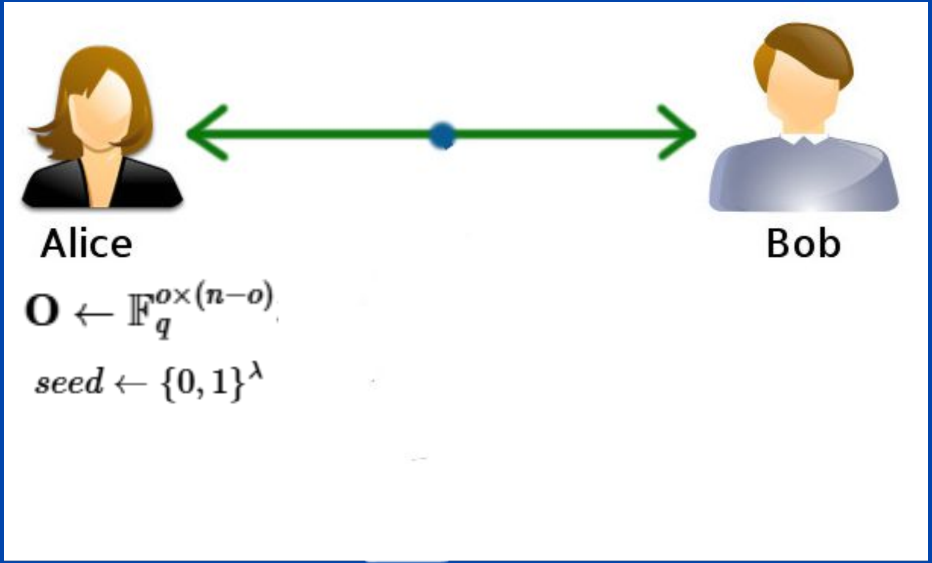
4. (Combine)
Compute the final public matrices ***P_i(3)*** using ***O***, ***P_i(1)***, ***P_i(2)***. ***P_i(3)*** represents the complex public quadratic system.

Outputs:
pk = (*seedpk*, ***{P_i(3)}***)
sk = (*seedsk*, ***O***)

Key Generation

1. Create the Trapdoor

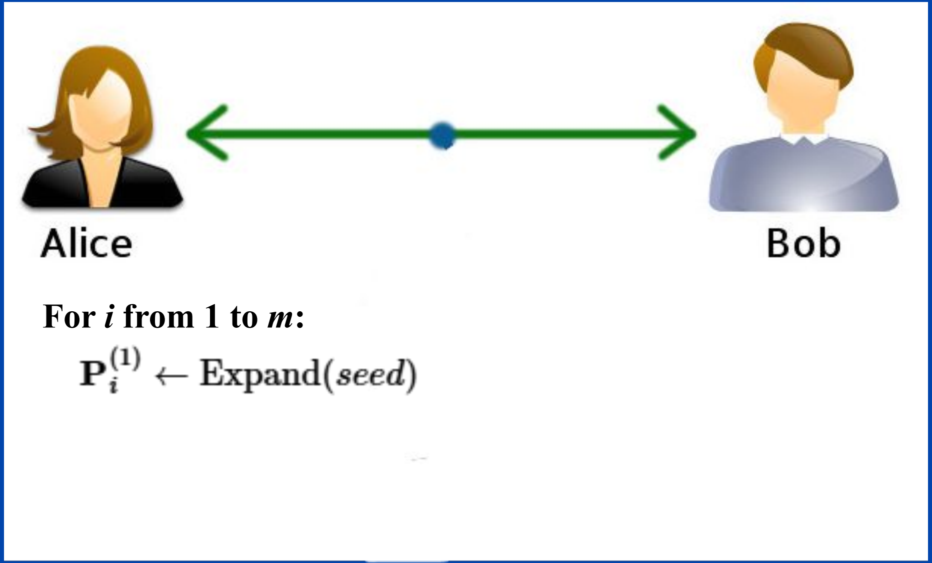
Alice creates a special system of quadratic equations (P) that is structured according to the original Oil and Vinegar design. ***O*** is her starting matrix where P will equal 0.


Alice
Bob
$\mathbf{O} \leftarrow \mathbb{F}_q^{o\times(n-o)}$
$seed \leftarrow \{0,1\}^\lambda$

Key Generation

2. Create the Shufflers

Alice chooses two random, invertible linear mixing functions, *S* and *T*. These are her random shuffling cards. She'll use them to completely hide the simple structure of the trapdoor P.

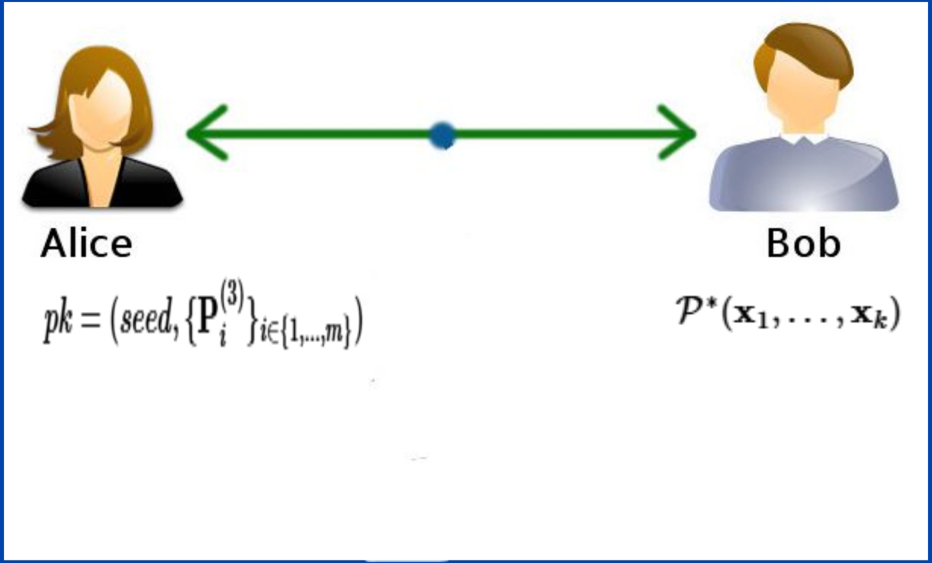


```
For $i$ from 1 to $m$:
    $\mathbf{P}_i^{(1)} \leftarrow \mathrm{Expand}(seed)$
```

Key Generation

3. Build the Public Lock

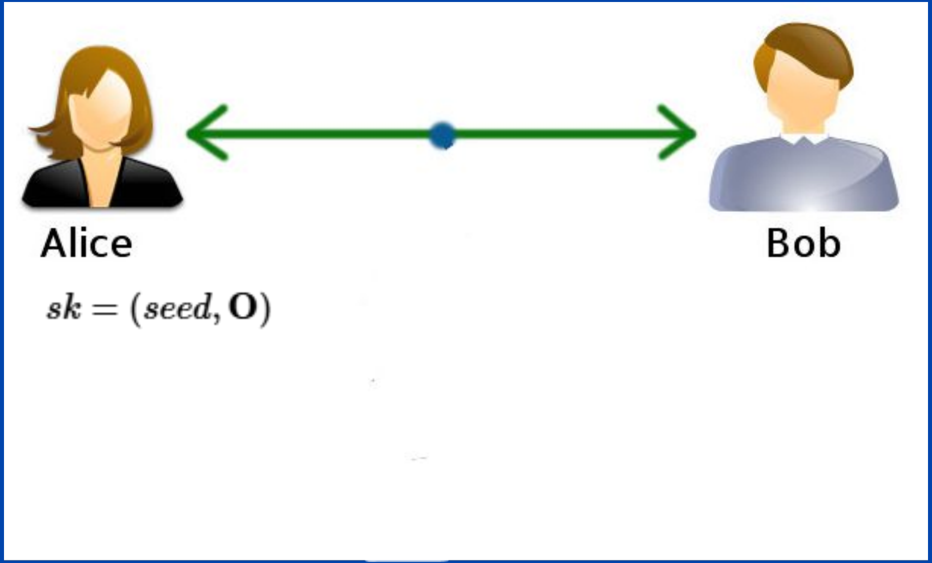
Alice composes the three functions:
$P* = S \circ P \circ T$. She then publishes the coefficients of this massive, complex $P*$ as her Public Key. This is the final, scrambled equation. It's so tangled that no one can easily see the structure inside.


Alice
Bob
$pk = \left(seed, \{\mathbf{P}_i^{(3)}\}_{i\in\{1,\dots,m\}}\right)$
$\mathcal{P}^*(\mathbf{x}_1,\dots,\mathbf{x}_k)$

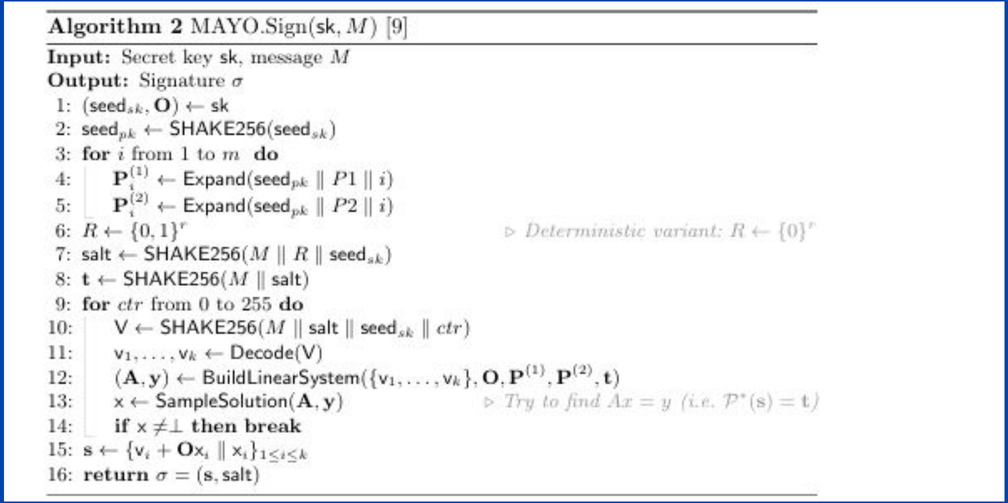
Key Generation

4. Keep the Secret Key

Alice keeps the original trapdoor P and the two shufflers (*S* and *T*) as her Secret Key. This is the key to creating a valid signature: the map P tells her how the equations are structured, and *S* and *T* tell her how to un-scramble and descramble everything.



Sign



```
Algorithm 2 MAYO.Sign(sk, $M$) [9]
Input: Secret key sk, message $M$
Output: Signature $\sigma$
(seed$_{sk}$, $\mathbf{O}$) $\leftarrow$ sk
seed$_{pk}$ $\leftarrow$ SHAKE256(seed$_{sk}$)
for $i$ from 1 to $m$ do
    $\mathbf{P}_i^{(1)} \leftarrow$ Expand(seed$_{pk}$ $\|$ $P1$ $\|$ $i$)
    $\mathbf{P}_i^{(2)} \leftarrow$ Expand(seed$_{pk}$ $\|$ $P2$ $\|$ $i$)
$R \leftarrow \{0,1\}^r$                                   ▷ Deterministic variant: $R \leftarrow \{0\}^r$
salt $\leftarrow$ SHAKE256($M$ $\|$ $R$ $\|$ seed$_{sk}$)
$\mathbf{t} \leftarrow$ SHAKE256($M$ $\|$ salt)
for $ctr$ from 0 to 255 do
    V $\leftarrow$ SHAKE256($M$ $\|$ salt $\|$ seed$_{sk}$ $\|$ $ctr$)
    $\mathsf{v}_1, \ldots, \mathsf{v}_k \leftarrow$ Decode(V)
    $(\mathbf{A}, \mathbf{y}) \leftarrow$ BuildLinearSystem($\{\mathsf{v}_1, \ldots, \mathsf{v}_k\}, \mathbf{O}, \mathbf{P}^{(1)}, \mathbf{P}^{(2)}, \mathbf{t}$)
    $\mathsf{x} \leftarrow$ SampleSolution($\mathbf{A}, \mathbf{y}$)          ▷ Try to find $Ax = y$ (i.e. $\mathcal{P}^*(\mathbf{s}) = \mathbf{t}$)
    if $\mathsf{x} \neq \perp$ then break
$\mathbf{s} \leftarrow \{\mathsf{v}_i + \mathbf{O}\mathsf{x}_i \| \mathsf{x}_i\}_{1 \le i \le k}$
return $\sigma = (\mathbf{s}, \text{salt})$
```

1. (Initialization)
Re-derive public constants $P_i(1)$, $P_i(2)$ from *seedsk*. Ensures the signer works with the correct public system.

2. (Target Hash)
Compute salt and the target vector t = Hash(M // salt). t is the specific output the quadratic system must produce.

3. (Build Linear System)
Construct a solvable linear system $Ax = y$ using the ***O*** matrix and t. This transforms the hard quadratic problem into an easy linear problem using the secret key ***O***.

4. (Solve)
Find the solution vector x for the linear system. The necessary component for the signature is found.

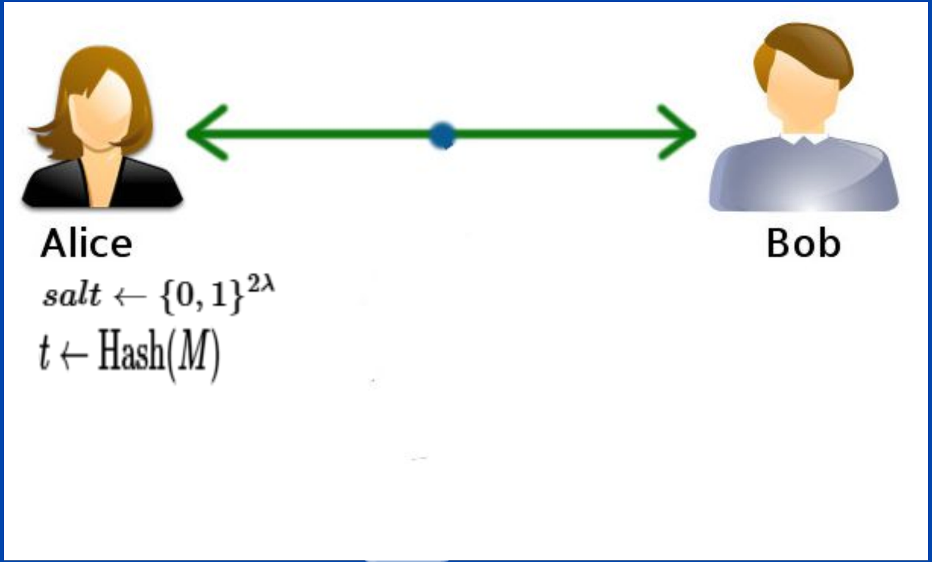
5. (Finalize)
Compute the final signature vector s from x and other components. s is the input vector that satisfies the public quadratic system.

Output Signature σ = (s, salt)

Sign

1. Define the Target

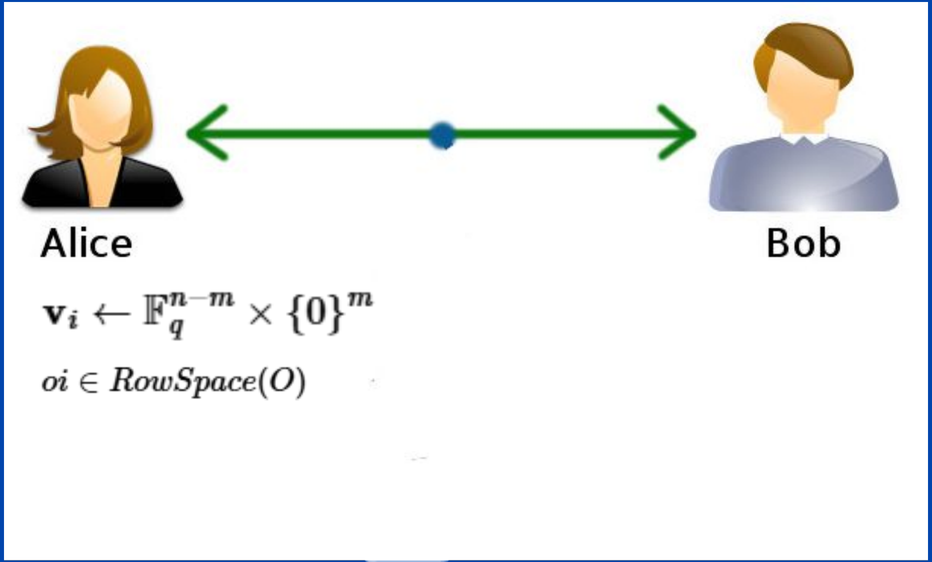
Alice first calculates the hash of the message: H(μ). This hash is her target output. This is the goal number she needs her final equation to equal.


Alice
Bob
$salt \leftarrow \{0,1\}^{2\lambda}$
$t \leftarrow \mathrm{Hash}(M)$

Sign

2. Un-shuffle the Target

Using her secret shuffler S, she calculates the inverse target *b* = $S^{-1}(H(\mu))$. She figures out what the target number must have been before she applied her output shuffler S. The goal is now to solve P(…) = *b*.

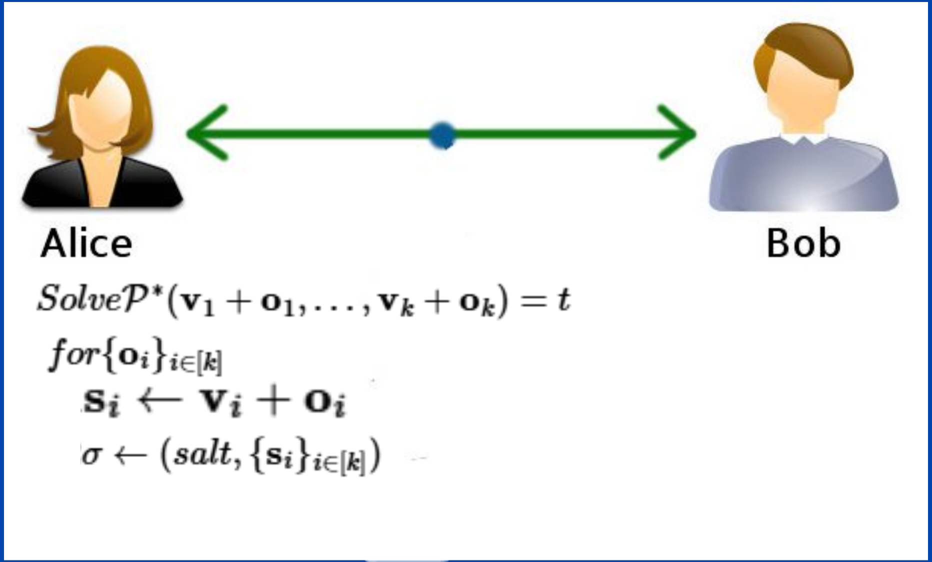


$$\mathbf{v}_i \leftarrow \mathbb{F}_q^{n-m} \times \{0\}^m$$

$$oi \in RowSpace(O)$$

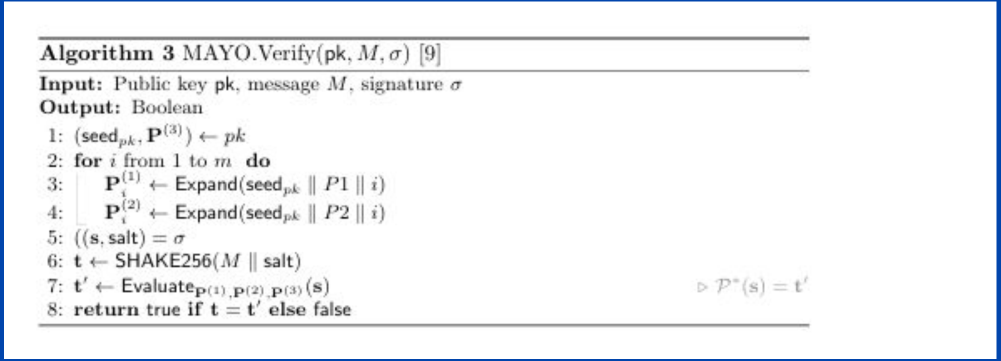
Sign

3. Solve the trapdoor



```
$Solve\mathcal{P}^*(\mathbf{v}_1 + \mathbf{o}_1, \ldots, \mathbf{v}_k + \mathbf{o}_k) = t$
 $for\{\mathbf{o}_i\}_{i\in[k]}$
    $\mathbf{s}_i \leftarrow \mathbf{v}_i + \mathbf{o}_i$
    $\sigma \leftarrow (salt, \{\mathbf{s}_i\}_{i\in[k]})$
```

Verification



```
Algorithm 3 MAYO.Verify(pk, M, σ) [9]
Input: Public key pk, message M, signature σ
Output: Boolean
1: $(\mathsf{seed}_{pk}, \mathbf{P}^{(3)}) \leftarrow pk$
2: for $i$ from 1 to $m$ do
3:     $\mathbf{P}_i^{(1)} \leftarrow \mathsf{Expand}(\mathsf{seed}_{pk} \,\|\, P1 \,\|\, i)$
4:     $\mathbf{P}_i^{(2)} \leftarrow \mathsf{Expand}(\mathsf{seed}_{pk} \,\|\, P2 \,\|\, i)$
5: $((\mathbf{s}, \mathsf{salt}) = \sigma$
6: $\mathbf{t} \leftarrow \mathsf{SHAKE256}(M \,\|\, \mathsf{salt})$
7: $\mathbf{t}' \leftarrow \mathsf{Evaluate}_{\mathbf{P}^{(1)},\mathbf{P}^{(2)},\mathbf{P}^{(3)}}(\mathbf{s})$        ▷ $\mathcal{P}^*(\mathbf{s}) = \mathbf{t}'$
8: return true if $\mathbf{t} = \mathbf{t}'$ else false
```

1. (Setup)
Extract (s, salt) from *σ*. Re-derive $P_i(1)$, $P_i(2)$ from *seedpk*. Prepare the public components needed for the check.

2. (Recalculate Target)
Compute the expected target t′ = Hash(M // salt). Determines what the public system should output for this message/salt pair.

3. (Evaluate)
Plug the signature vector s into the public quadratic system defined by $P_i(1)$, $P_i(2)$, $P_i(3)$.This is the core check: Calculate the actual output t of the public system when using s.

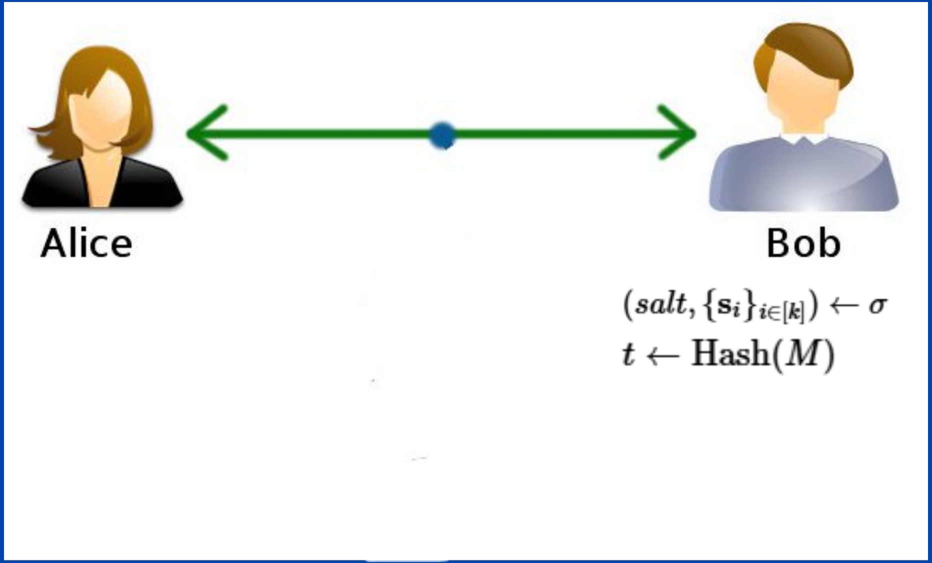
4. (Compare)
Check if t = t′. If they match, the signature is VALID. The input s truly leads to the expected hash t′.

Output: TRUE (Valid) or **FALSE** (Invalid)

Verification

1. Get the Target

Bob calculates the hash of the message: H(*μ*). He determines the target number the equation should equal.

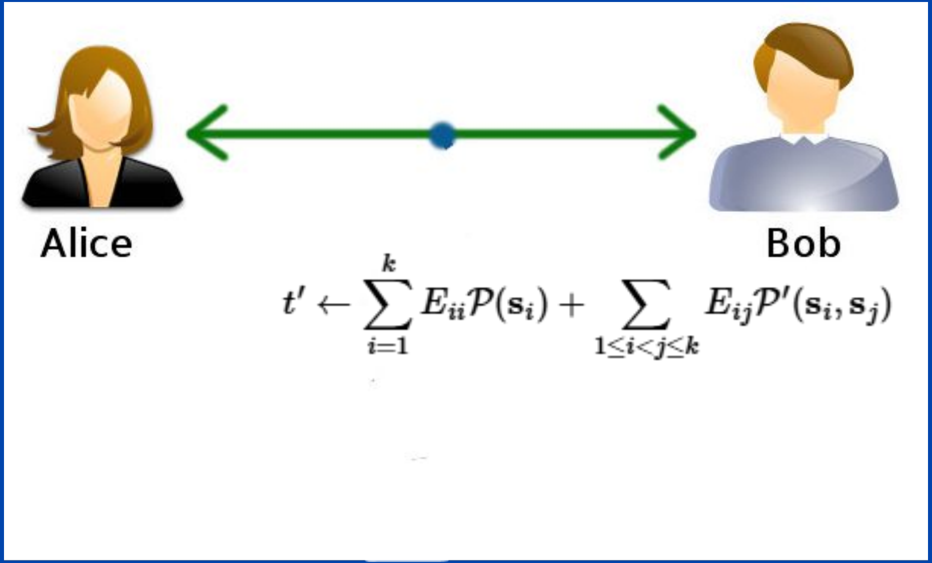


$$(salt, \{\mathbf{s}_i\}_{i\in[k]}) \leftarrow \sigma$$
$$t \leftarrow \mathrm{Hash}(M)$$

Verification

2. Test the Lock

Bob takes the signature *s* and plugs it directly into Alice's massive, complex Public Key equation P. He calculates the output *y* = *P∗(s)*. He plugs the key (*s*) into the lock (P∗). Since this is a massive quadratic system, it takes him a bit of time to compute, but it's straightforward (no need to solve equations, just evaluate them).

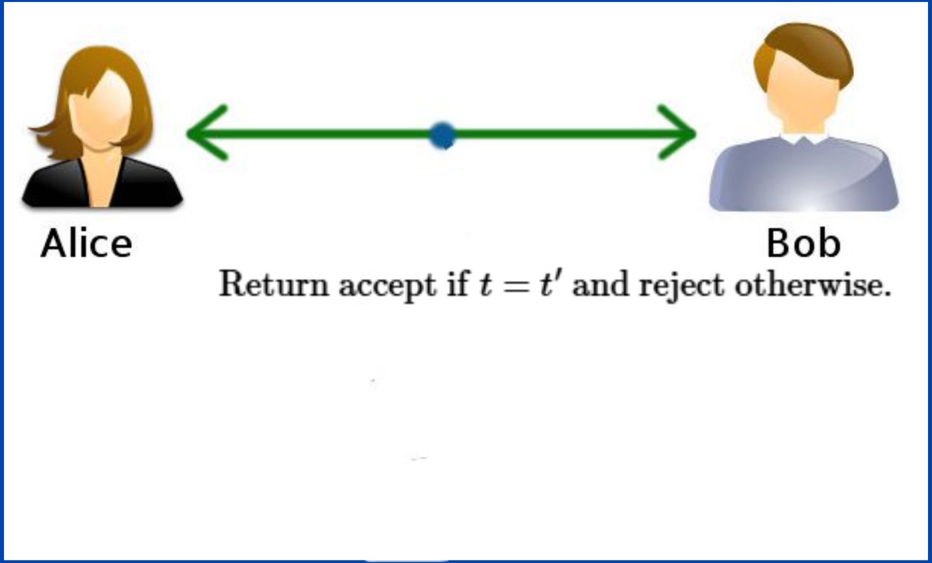


$$t' \leftarrow \sum_{i=1}^{k} E_{ii}\mathcal{P}(\mathbf{s}_i) + \sum_{1\le i<j\le k} E_{ij}\mathcal{P}'(\mathbf{s}_i, \mathbf{s}_j)$$

Verification

3. Compare

Bob checks: Is the output y equal to the target hash H(*μ*)? If the signature came from Alice (i.e., she used the secret trapdoor correctly), the output must match the hash.
Result: If *y* = *H(μ)*, the signature is VALID. Otherwise, it's invalid.



Return accept if $t = t'$ and reject otherwise.

Security Analysis

Security Analysis

- One of the many benefits of MAYO is its lightened computational load.
- This makes it quicker and easier to follow than something popular such as RSA, which falls victim due to its factorization techniques within its computations.
- However, to analyze its applications in security, we have to take a look at its older brother: the Oil and Vinegar scheme.

Oil and Vinegar (OV) Problem

- The Oil and Vinegar signature scheme asks an adversary to distinguish a random multivariate quadratic map P from one that vanishes on the row space of a specific O.
- This problem is considered relatively well-understood. However it does have a caveat of rather large public keys. This is good for MAYO as it is just a variant of the OV problem with a reduced oil space.
- This means that MAYO is something that is computationally readily available and the hardness of distinguishing structured maps from random ones is preserved under this restriction.
- MAYO inherits the foundational security assumptions of OV while offering a more compact and implementation-friendly design.
- Moreover, the reduced oil space simplifies the algebraic structure of the public key, which can lead to performance gains in both signature generation and verification.
- This makes MAYO a promising candidate for constrained environments, such as embedded systems or post-quantum secure messaging protocols.

MAYO’s Own Features

- The Multi-Target Whipped MQ Problem extends the OV scheme by introducing structured bilinear combinations across multiple inputs and targets, making it harder to forge signatures on average.
- Its older counterpart, OV focuses on distinguishing structured quadratic maps from random ones using a single trapdoor and target, which is effective but MQ is more secure.
- What sets Mayo apart is its emphasis on efficiency and adaptability. Although still in the testing phase, the scheme is being continuously refined to minimize memory consumption without compromising computational performance.
- This optimization is crucial for deployment in environments ranging from lightweight embedded systems to high-throughput quantum-resistant infrastructure.
- Mayo is engineered to maintain consistent performance across both conventional and quantum computing platforms.
- The signing process for it is designed to run in constant time to prevent information leaks from timing attacks, improves security.

Limitations and Caveats of MAYO

- MAYO, while efficient and definitely faster & more optimized than something like RSA, still requires heavy computation and can occasionally fail to sign if its linear system lacks full rank.
- This is a rare but important design consideration, especially since it's still being tested.
- Also since the seed, which is used for the trapdoor function and public key generation is deterministic, it could general predictable keys.
- There is a possibility that it is vulnerable to fault injection attacks through both instances. As such, this problem needs to be addressed under newer development runs of this scheme.
- Its security relies on ad-hoc assumptions, included with its added “Multi-Target Whipped MQ Problem, which lacks broad validation.
- Although it resists key leakage under well-chosen parameters, its theoretical guarantees remain uncertain for now.

Big round of applause for the Team behind our scheme!

Because *they* are the real heroes!

- **Ward Beullens**: (Cryptography Researcher at IBM Research)
- **Fabio Campos**: (Professor at Darmstadt University of Applied Sciences, Germany)
- **Sofía Celiasil Hess**: (Honorary Industrial Fellow in Cryptography for Privacy at the University of Bristol)
- **Matthias J. Kannwischer**: (Research Director at Chelpis Quantum Tech and Adjunct Assistant Professor at National Taiwan University)

References

- The Oil and Vinegar Method
- MAYO: Overview + Updates
- MAYO: Round 2 Version
- MAYO | Open Quantum Safe
- Whipping the Multivariate-based MAYO Signature Scheme using Hardware Platforms
- MAYO: Practical Post-Quantum Signatures from Oil-and-Vinegar Maps

Thank you!

We will talk about it more, next time

*And no, it's **not** a instrument!*