

Breaking the One-Way Property of a SHA-3 Implementation via Fault Injection

Key Recovery Attacks on Post-Quantum Digital Signatures

FDTC - October 11, 2026

Mona Sobhani, Sönke Jendral, Mats Näslund, Elena Dubrova
KTH Royal Institute of Technology, Sweden

Motivation

1. Quantum computers are threat to current public-key cryptography
 - Post-Quantum Cryptography (PQC) algorithms suggested
 - Digital signatures and key encapsulation schemes
 - Digital signature candidates:
 - NIST competition round 2 (March 2025)
 - NIST competition round 3 (May 2026)
2. New PQC implementations need to be evaluated against physical attacks (side-channel attacks & fault injection attacks)

Motivation

1. Quantum computers are threat to current public-key cryptography
 - Post-Quantum Cryptography (PQC) algorithms suggested
 - Digital signatures and key encapsulation schemes
 - Digital signature candidates:
 - **NIST competition round 2 (March 2025)**
 - NIST competition round 3 (May 2026)
2. New PQC implementations need to be evaluated against physical attacks (side-channel attacks & **fault injection attacks**)

Digital Signature & Key Recovery Idea

3 Phases:

1. Key Generation
2. Signature Generation
3. Verification

Goal:

Signer proves knowledge of secret,
without revealing it

- Zero-Knowledge Proof
- Non-Interactive (Fiat-Shamir transform)

Digital Signature & Key Recovery Idea

3 Phases:

1. Key Generation
2. Signature Generation
3. Verification

$$\text{Secret Key} = A + B$$

For i rounds:

reveal(A), hide(B)

OR

hide(A), reveal(B)

For i rounds:

Validate using A

OR

Validate using B

Digital Signature & Key Recovery Idea

3 Phases:

1. Key Generation
2. Signature Generation
3. Verification

Secret Key = $A + B$

For i rounds:

reveal(A), hide(B)

OR

hide(A), reveal(B)

For i rounds:

Validate using A

OR

Validate using B

Digital Signature & Key Recovery Idea

3 Phases:

1. Key Generation
2. Signature Generation
3. Verification

$$\text{Secret Key} = A + B$$

For i rounds:

reveal(A), hide(B)

OR AND

hide(A), reveal(B)

For i rounds:

Validate using A

OR

Validate using B

Digital Signature & Key Recovery Idea

3 Phases:

1. Key Generation
2. Signature Generation
3. Verification

$$\text{Secret Key} = A + B$$

For i rounds:

reveal(A), hide(B)

OR AND

hide(A), reveal(B)

For i rounds:

Validate using A

OR

Validate using B

Digital Signature & Key Recovery Idea

CROSS

```

5: for  $i$  from 1 to  $t$  do
6:    $\bar{e}'[i], u'[i] \leftarrow \text{CSPRNG}(\text{Seed}[i] \parallel \text{Salt} \parallel i + 2t - 1)$ 
7:    $\bar{v}[i] \leftarrow \bar{e} - \bar{e}'[i]$ 
8:    $v[i] \leftarrow g^{\bar{v}[i]}$ 
9:    $u[i] \leftarrow v[i] * u'[i]$ 
10:   $s'[i] \leftarrow u[i] H^T$ 
11:   $\text{cmt}_0[i] \leftarrow \text{Hash}(s'[i] \parallel \bar{v}[i] \parallel \text{Salt} \parallel i + 2t - 1)$ 
12:   $\text{cmt}_1[i] \leftarrow \text{Hash}(\text{Seed}[i] \parallel \text{Salt} \parallel i + 2t - 1)$ 
13:  $\text{digest}_{\text{cmt}_0} \leftarrow \text{TreeRoot}(\text{cmt}_0)$ 
14:  $\text{digest}_{\text{cmt}_1} \leftarrow \text{Hash}(\text{cmt}_1)$ 
15:  $\text{digest}_{\text{cmt}} \leftarrow \text{Hash}(\text{digest}_{\text{cmt}_0} \parallel \text{digest}_{\text{cmt}_1})$ 
16: // Challenge generation

```

$$\text{Secret Key} = A + B$$

For i rounds:

reveal(A), hide(B)

OR AND

hide(A), reveal(B)

For i rounds:

Validate using A

OR

Validate using B

Digital Signature & Key Recovery Idea

CROSS

```

5: for  $i$  from 1 to  $t$  do
6:    $\bar{e}'[i], u'[i] \leftarrow \text{CSPRNG}(\text{Seed}[i] \parallel \text{Salt} \parallel i + 2t - 1)$ 
7:    $\bar{v}[i] \leftarrow \bar{e} - \bar{e}'[i]$ 
8:    $v[i] \leftarrow g^{\bar{v}[i]}$ 
9:    $u[i] \leftarrow v[i] * u'[i]$ 
10:   $s'[i] \leftarrow u[i] H^T$ 
11:   $\text{cmt}_0[i] \leftarrow \text{Hash}(s'[i] \parallel \bar{v}[i] \parallel \text{Salt} \parallel i + 2t - 1)$ 
12:   $\text{cmt}_1[i] \leftarrow \text{Hash}(\text{Seed}[i] \parallel \text{Salt} \parallel i + 2t - 1)$ 
13:  $\text{digest}_{\text{cmt}_0} \leftarrow \text{TreeRoot}(\text{cmt}_0)$ 
14:  $\text{digest}_{\text{cmt}_1} \leftarrow \text{Hash}(\text{cmt}_1)$ 
15:  $\text{digest}_{\text{cmt}} \leftarrow \text{Hash}(\text{digest}_{\text{cmt}_0} \parallel \text{digest}_{\text{cmt}_1})$ 
16: // Challenge generation

```

$$\text{Secret Key} = A + B$$

For i rounds:

reveal(A), hide(B)

OR AND

hide(A), reveal(B)

For i rounds:

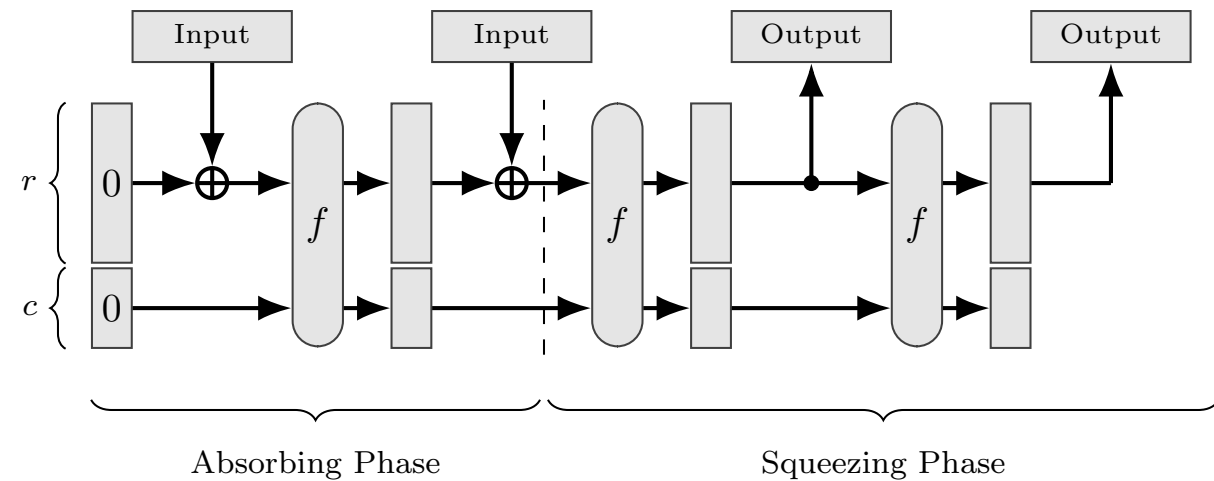
Validate using A

OR

Validate using B

SHA-3 hash function

- SHAKE from FIPS-202
 - **Secure Hash Algorithm Keccak**
Extendable Output Function
 - One-way property
- Sponge construction
 - Absorbing phase
 - Squeezing phase



CROSS

```
5: for  $i$  from 1 to  $n$  do
6:    $\bar{\mathbf{e}}'[i], \mathbf{u}'[i] \leftarrow \text{CSPRNG}(\text{Seed}[i] \mid \text{Salt} \mid i + 2t - 1)$ 
7:    $\bar{\mathbf{v}}[i] \leftarrow \bar{\mathbf{e}} - \bar{\mathbf{e}}'[i]$ 
8:    $\mathbf{v}[i] \leftarrow g^{\bar{\mathbf{v}}[i]}$ 
9:    $\mathbf{u}[i] \leftarrow \mathbf{v}[i] * \mathbf{u}'[i]$ 
10:   $\mathbf{s}'[i] \leftarrow \mathbf{u}[i] \mathbf{H}^T$ 
11:   $\text{cmt}_0[i] \leftarrow \text{Hash}(\mathbf{s}'[i] \mid \bar{\mathbf{v}}[i] \mid \text{Salt} \mid i + 2t - 1)$ 
12:   $\text{cmt}_1[i] \leftarrow \text{Hash}(\text{Seed}[i] \mid \text{Salt} \mid i + 2t - 1)$ 
13:  $\text{digest}_{\text{cmt}_0} \leftarrow \text{TreeRoot}(\text{cmt}_0)$ 
14:  $\text{digest}_{\text{cmt}_1} \leftarrow \text{Hash}(\text{cmt}_1)$ 
15:  $\text{digest}_{\text{cmt}} \leftarrow \text{Hash}(\text{digest}_{\text{cmt}_0} \mid \text{digest}_{\text{cmt}_1})$ 
16: // Challenge generation
```

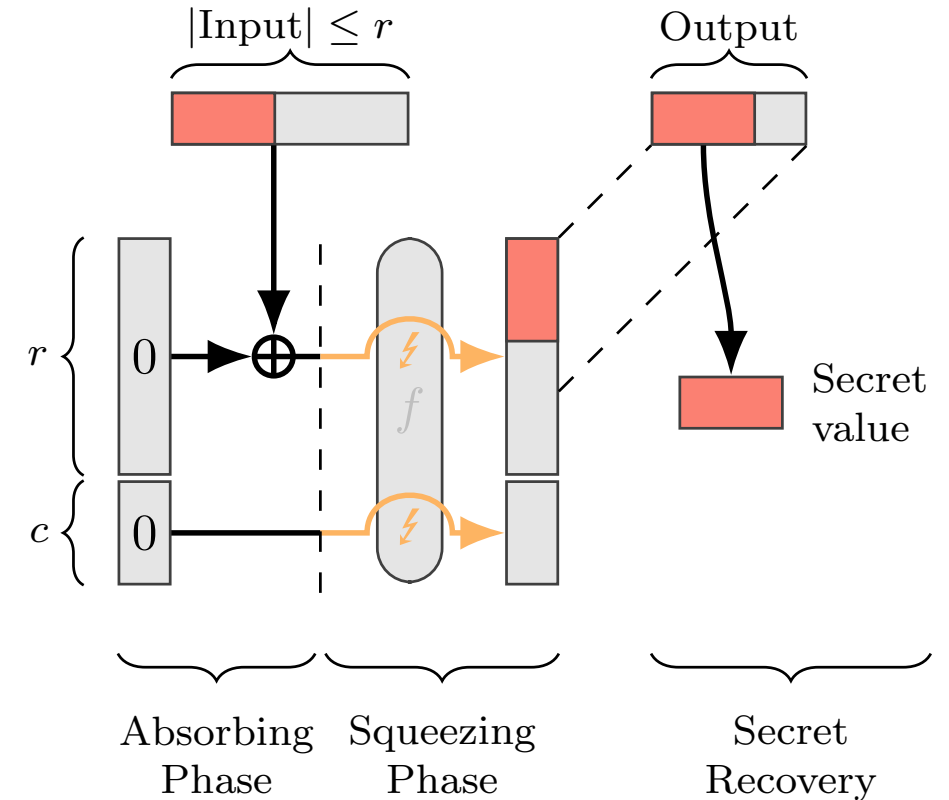
Direct-Read Attack

ATTACK 1 OF 3

- Example CROSS:

$$cmt_1[i] = \text{Hash}(\text{Seed}[i] \parallel \text{Salt} \parallel i + 2t - 1)$$

- Input $\leq$ rate size
 - No permutation in absorb phase
- Skip permutation in squeeze phase



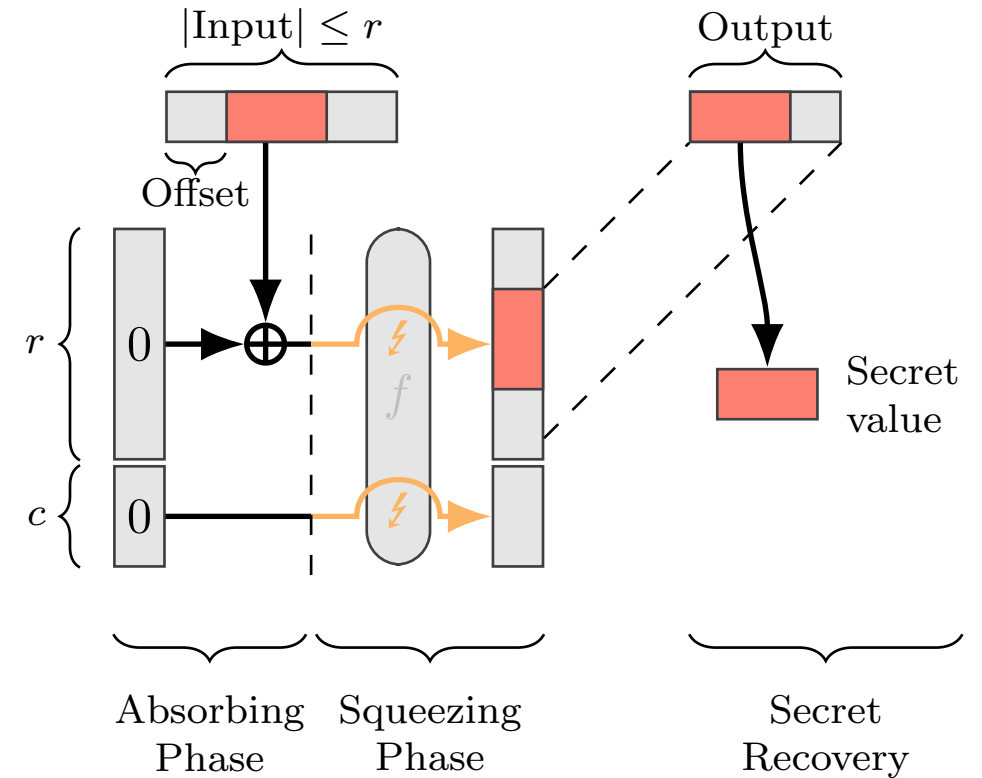
Offset-Read Attack

ATTACK 2 OF 3

- Example CROSS:

$$cmt_1[i] = \text{Hash}(s'[i] \parallel \bar{v}[i] \parallel \text{Salt} \parallel i + 2t - 1)$$

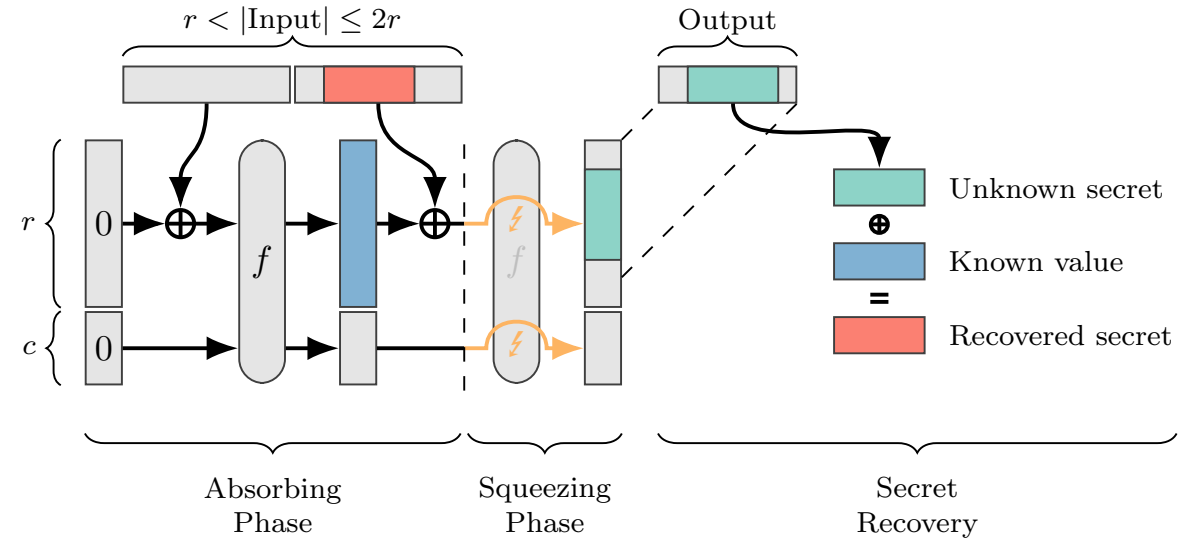
- Input $\leq$ rate size
 - No permutation in absorb phase
- Skip permutation in squeeze phase
- Output read from offset-position



Known-Prefix Attack

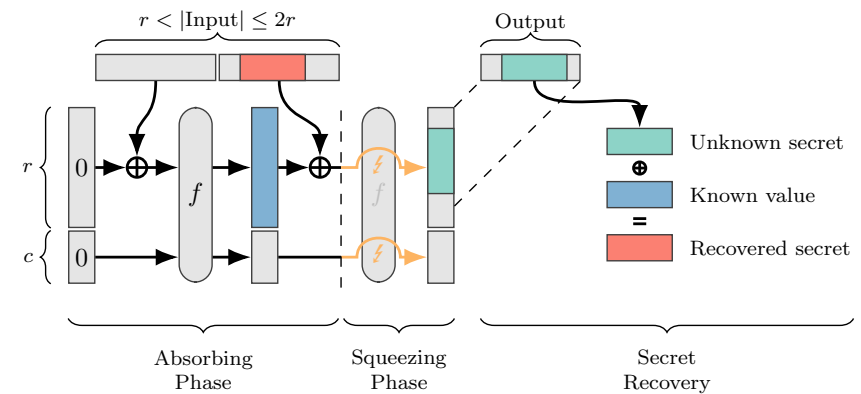
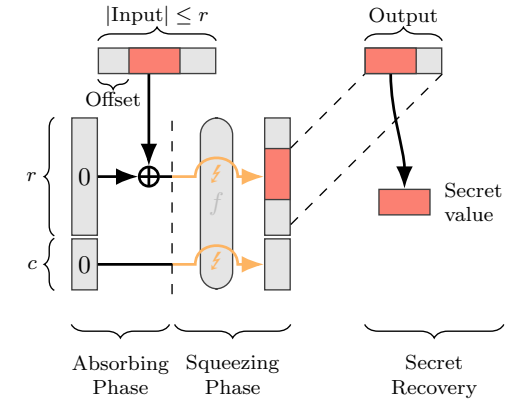
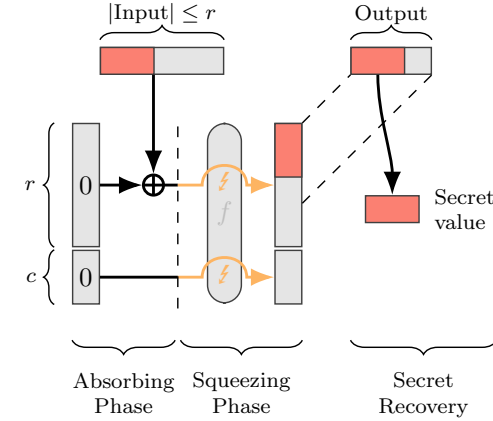
ATTACK 3 OF 3

- Secret within second input block
- Absorb contains permutation f
- First input block must be known
- XOR output with known state



SHA-3 implementation

- PQM4 implementation





SHA-3 implementation

- PQM4 implementation
- Experiments on CROSS

CROSS

Code based signature

$\text{cmt}_1[i] = \text{Hash}(\text{Seed}[i] \mid \text{Salt} \mid i + 2t - 1)$

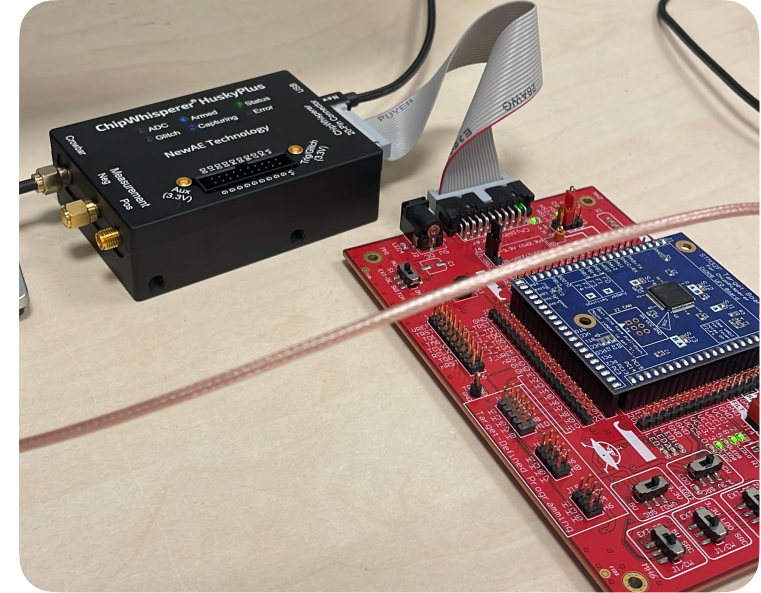
SHA-3 implementation

- PQM4 implementation
- Experiments on CROSS
- Applicable to multiple digital signature candidates

CROSS	Code based signature	$\text{cmt}_1[i] = \text{Hash}(\text{Seed}[i] \mid \text{Salt} \mid i + 2t - 1)$
LESS	Code based signature	$\text{state} = \text{XOF.INIT}(\mathcal{T}[\text{parent}] \mid \text{salt} \mid \text{parent})$
PERK	MPC-in-the-Head	$\text{com}_{e,i} = \text{SHA3-}\lambda(\text{salt} \mid \text{index} \mid \text{seed}_{e,i} \mid \text{dom})$
RYDE	MPC-in-the-Head	$\text{com}_{e,i} = \text{Hash3}(\text{salt} \mid \text{bin32}(\text{id}_x) \mid \text{seed}_{e,i})$
MIRATH	MPC-in-the-Head	$\text{salt} = \text{SHAKE256}(M_{\text{digest}} \mid R \mid \text{seed}_{\text{sk}})$
MAYO	Multivariate	$\text{com}[e][i] = \text{SHA3-}\lambda(0x03 \mid \text{salt} \mid i \mid \text{seed}_{e,i})$

Experimental Setup & Results

- Voltage glitch with ChipWhisperer HuskyPlus
- Optimized PQM4 implementation
- ARM Cortex-M4
- 1000 signature runs for one attack



Experimental Setup & Results

- Direct Read Attack
 - 100% success rate
- Offset-Read Attack
 - 63,9% success rate
- Known-Prefix Attack
 - Not applicable to CROSS

Parameter Set	Direct-Read	Offset-Read	Known-Prefix
CROSS-RSDP-1	✓	✓(2^6)	—
CROSS-RSDP-3	✓	—	—
CROSS-RSDP-5	✓	—	—
CROSS-RSDP(G)-1	✓	—	—
CROSS-RSDP(G)-3	✓	✓(2^{48})	—
CROSS-RSDP(G)-5	✓	—	—

Parameter Set	Direct-Read	Offset-Read	Known-Prefix
LESS-I	✓	—	—
LESS-III	✓	—	—
LESS-V	✓	—	—
PERK-3	—	✓	—
PERK-5	—	—	✓ [*] (2^{40})
RYDE-3	—	✓ [*] (2^{24})	—
RYDE-5	—	—	✓
Mirath-3	—	✓ [*] (2^{16})	—
Mirath-5	—	—	✓ [*] (2^{24})
MAYO-1	—	✓	—
MAYO-2	—	✓	—

CROSS

Code based signature

$$\text{cmt}_1[i] = \text{Hash}(\text{Seed}[i] \mid \text{Salt} \mid i + 2t - 1)$$

LESS

Code based signature

$$\text{state} = \text{XOF.INIT}(\mathcal{T}[\text{parent}] \mid \text{salt} \mid \text{parent})$$

PERK

MPC-in-the-Head

$$\text{com}_{e,i} = \text{SHA3-}\lambda(\text{salt} \mid \text{index} \mid \text{seed}_{e,i} \mid \text{dom})$$

RYDE

MPC-in-the-Head

$$\text{com}_{e,i} = \text{Hash3}(\text{salt} \mid \text{bin32}(\text{id}_x) \mid \text{seed}_{e,i})$$

MIRATH

MPC-in-the-Head

$$\text{salt} = \text{SHAKE256}(\text{M}_{\text{digest}} \mid R \mid \text{seed}_{\text{sk}})$$

MAYO

Multivariate

$$\text{com}[e][i] = \text{SHA3-}\lambda(0x03 \mid \text{salt} \mid i \mid \text{seed}_{e,i})$$



Search NIST



Menu

PUBLICATIONS

Status Report on the Second Round of the Additional Digital Signature Schemes for the NIST Post-Quantum Cryptography Standardization Process

Published: May 14, 2026

CROSS	Code based signature	$\text{cmt}_1[i] = \text{Hash}(\text{Seed}[i] \mid \text{Salt} \mid i + 2t - 1)$
LESS	Code based signature	$\text{state} = \text{XOF.INIT}(\mathcal{T}[\text{parent}] \mid \text{salt} \mid \text{parent})$
PERK	MPC-in-the-Head	$\text{com}_{e,i} = \text{SHA3-}\lambda(\text{salt} \mid \text{index} \mid \text{seed}_{e,i} \mid \text{dom})$
RYDE	MPC-in-the-Head	$\text{com}_{e,i} = \text{Hash3}(\text{salt} \mid \text{bin32}(\text{idx}) \mid \text{seed}_{e,i})$
MIRATH	MPC-in-the-Head	$\text{salt} = \text{SHAKE256}(\text{M}_{\text{digest}} \mid R \mid \text{seed}_{\text{sk}})$
MAYO	Multivariate	$\text{com}[e][i] = \text{SHA3-}\lambda(0\text{x}03 \mid \text{salt} \mid i \mid \text{seed}_{e,i})$

Countermeasures

1. Double Hashing/Computation

- Pre-hash secret inputs
- Perform critical computations twice and verify output
- How much overhead is allowed?

2. Code Structure

- Inlining eliminates call and return branches
- Move secret data outside the output/offset range

3. Hardware/Compiler Defense

- Control Flow Integrity
 - Validate execution paths to detect skipped instructions
- Hardware fault detectors

Conclusion

- Demonstrated on PQM4 implementation
 - Single instruction-skip
 - Breaks the one-way property
 - Applicable to other sponge constructions like Ascon-XOF [Nat25a]
- Offset-Read & Known-Prefix Attack
 - Read output from offset
 - Recovery even for large inputs
- Applicability
 - 6 different NIST Round-2 candidates

Conclusion

- Demonstrated on PQM4 implementation
 - Single instruction-skip
 - Breaks the one-way property
 - Applicable to other sponge constructions like Ascon-XOF [Nat25a]
- Offset-Read & Known-Prefix Attack
 - Read output from offset
 - Recovery even for large inputs
- Applicability
 - 6 different NIST Round-2 candidates

→ Test one attack across different algorithms, to find vulnerabilities in implementations that apply widely.



FDTC - October 11, 2026

Mona Sobhani, Sönke Jendral, Mats Näslund, Elena Dubrova
KTH Royal Institute of Technology, Sweden

Sources

FDTC - October 11, 2026

- [Nat25a] National Institute of Standards and Technology. Ascon-based lightweight cryptography standards for constrained devices. Technical Report NIST SP 800-232, National Institute of Standards and Technology, Gaithersburg, MD, August 2025.