AIMer v2.1 and Beyond

June 2025

Seongkwang Kim

Samsung SDS

PQC Competitions

NIST PQC Competition (2016.11 - 2025.3)

- 1st round (2017.11 - 2019.1)
  - 49 KEM submissions, 20 digital signature submissions
- 2nd round (2019.1 - 2020.7)
  - 17 KEM (including PKE) schemes, and 9 digital signature schemes
- 3rd round (2020.7 - 2022.7)
  - KEM: Classic McEliece, Kyber, NTRU, Saber, BIKE, FrodoKEM, HQC, NTRU Prime, SIKE
  - DS: Dilithium, Falcon, Rainbow, GeMSS, Picnic, SPHINCS+

NIST PQC Competition (2016.11 - 2025.3)

- 3rd round selection (2022.7)
  - KEM: Kyber (ML-KEM)
  - DS: Dilithium (ML-DSA), Falcon (FN-DSA), SPHINCS+ (SLH-DSA)
- 4th round (2022.7 - 2025.3)
  - KEM: Classic McEliece, HQC, BIKE, SIKE
  - 4th round selection (2025.3): HQC
- Documents
  - FIPS published: ML-KEM (FIPS 203), ML-DSA (FIPS 204), SLH-DSA (FIPS 205)
  - FIPS not yet published: FN-DSA (maybe soon), HQC (in 2 years)
  - Other works: transition (IR 8547), recommendations for KEM (SP 800-227), Short SLH-DSA

KpqC Competition (2021.11 - 2025.1)

- 1st round (2022.11 - 2023.12)
  - 7 KEM submissions, 9 DS submissions
- 2nd round (2023.12 - 2025.1)
  - KEM: NTRU+, PALOMA, REDOG, SMAUG-T
  - DS: AIMer, HAETAE, MQ-Sign, NCC-Sign
- Selected algorithms
  - KEM: NTRU+, SMAUG-T
  - DS: AIMer, HAETAE

NIST Call for Additional Signature Schemes (2022.9 - present)

- 1st round (2023.6 - 2024.10)
  - 6 code-based, 1 isogeny-based, 7 lattice-based, 7 MPCitH-based, 10 MQ-based, 4 symmetric-based, 5 others
- 2nd round (2024.10 - present)
  - 2 code-based, 1 isogeny-based, 1 lattice-based, 5 MPCitH-based, 4 MQ-based, 1 symmetric-based

Preliminaries

Additive Secret Sharing

- Each party shares the input value additively; for input x, P_i has $x^{(i)}$ such that

$$\sum_{i=1}^{n} x^{(i)} = x$$

- Addition is naturally compatible:

$$x + y = \sum_{i=1}^{n} x^{(i)} + \sum_{i=1}^{n} y^{(i)}.$$

Additive Secret Sharing

- Each party shares the input value additively; for input x, P_i has $x^{(i)}$ such that

$$\sum_{i=1}^{n} x^{(i)} = x$$

- Multiplication needs a multiplication triple.
  1. P_i has $(a^{(i)}, b^{(i)}, c^{(i)})$ such that $ab = c$
  2. P_i broadcasts $A^{(i)} = x^{(i)} - a^{(i)}$
  3. P_i broadcasts $B^{(i)} = y^{(i)} - b^{(i)}$
  4. P_i computes

$$\begin{aligned} z^{(i)} &= c^{(i)} + Ab^{(i)} + Ba^{(i)} + AB \\ &= c^{(i)} + (x-a)b^{(i)} + (y-b)a^{(i)} + (x-a)(y-b) = (xy)^{(i)} \end{aligned}$$

SPDZ Protocol

- Properties:
  - Maliciously-secure generic MPC in the preprocessing model
  - Additive secret sharing with IT-MAC

SPDZ Protocol

- Properties:
  - Maliciously-secure generic MPC in the preprocessing model
  - Additive secret sharing with IT-MAC
- Information-theoretic message authentication code (IT-MAC)
  - $\gamma(x) = \alpha \cdot x$
  - Each party shares $(\langle x \rangle, \langle \alpha \rangle, \langle \gamma(x) \rangle)$
  - Each party shares triple $(\langle a \rangle, \langle b \rangle, \langle c \rangle)$ and its MAC values

SPDZ Protocol

- Offline Phase (Preprocessing): Generate multiplication triples and its MACs using HE

SPDZ Protocol

- Offline Phase (Preprocessing): Generate multiplication triples and its MACs using HE
- Sacrificing technique:
  - Want to check multiplication triple $(\langle a \rangle, \langle b \rangle, \langle c \rangle)$ is honestly generated
  - Use another triple $(\langle f \rangle, \langle g \rangle, \langle h \rangle)$

SPDZ Protocol

- Offline Phase (Preprocessing): Generate multiplication triples and its MACs using HE
- Sacrificing technique:
    - Want to check multiplication triple $(\langle a\rangle, \langle b\rangle, \langle c\rangle)$ is honestly generated
    - Use another triple $(\langle f\rangle, \langle g\rangle, \langle h\rangle)$

  1. Randomly sample t
  2. Open $C = t \cdot \langle a\rangle - \langle f\rangle$ and $D = \langle b\rangle - \langle g\rangle$
  3. Evaluate $t \cdot \langle c\rangle - \langle h\rangle - D \cdot \langle f\rangle - C \cdot \langle g\rangle - CD$ and check whether it is zero

SPDZ Protocol

- Offline Phase (Preprocessing): Generate multiplication triples and its MACs using HE
- Sacrificing technique:
  - Want to check multiplication triple $(\langle a \rangle, \langle b \rangle, \langle c \rangle)$ is honestly generated
  - Use another triple $(\langle f \rangle, \langle g \rangle, \langle h \rangle)$

  1. Randomly sample t
  2. Open $C = t \cdot \langle a \rangle - \langle f \rangle$ and $D = \langle b \rangle - \langle g \rangle$
  3. Evaluate $t \cdot \langle c \rangle - \langle h \rangle - D \cdot \langle f \rangle - C \cdot \langle g \rangle - CD$ and check whether it is zero

  - If $c = ab + \varepsilon$ and $h = fg + \varepsilon'$, then

$$tc - h - (b - g)f - (ta - f)g - (b - g)(ta - f) = t\varepsilon - \varepsilon'$$

SPDZ Protocol

- Online Phase (Linear):
  - $\langle \gamma(mx + ny + k) \rangle = m \cdot \langle \gamma(x) \rangle + n \cdot \langle \gamma(y) \rangle + k \cdot \langle \alpha \rangle$

SPDZ Protocol

- Online Phase (Linear):
  - $\langle\gamma(mx+ny+k)\rangle = m\cdot\langle\gamma(x)\rangle + n\cdot\langle\gamma(y)\rangle + k\cdot\langle\alpha\rangle$
- Online Phase (Multiplication):
  1. Open $A = x - a$, $B = y - b$.
  2. Compute local share and MAC share of xy:

$$\langle xy\rangle = \langle c\rangle + A\langle b\rangle + B\langle a\rangle + AB,$$
$$\langle\gamma(xy)\rangle = \langle\gamma(c)\rangle + A\langle\gamma(b)\rangle + B\langle\gamma(a)\rangle + AB\langle\alpha\rangle$$

SPDZ Protocol

- Online Phase (Linear):
  - $\langle \gamma(mx + ny + k) \rangle = m \cdot \langle \gamma(x) \rangle + n \cdot \langle \gamma(y) \rangle + k \cdot \langle \alpha \rangle$
- Online Phase (Multiplication):
  1. Open $A = x - a$, $B = y - b$.
  2. Compute local share and MAC share of xy:

$$
\begin{aligned}
\langle xy \rangle &= \langle c \rangle + A\langle b \rangle + B\langle a \rangle + AB, \\
\langle \gamma(xy) \rangle &= \langle \gamma(c) \rangle + A\langle \gamma(b) \rangle + B\langle \gamma(a) \rangle + AB\langle \alpha \rangle
\end{aligned}
$$

- MAC Check: Commit $(\langle \alpha \rangle, \langle z \rangle, \langle \gamma(z) \rangle)$ and open it to check the sum of $\langle \gamma(z) \rangle - \alpha \langle z \rangle$ is zero.

MPC-in-the-Head

MPC-in-the-Head (MPCitH)

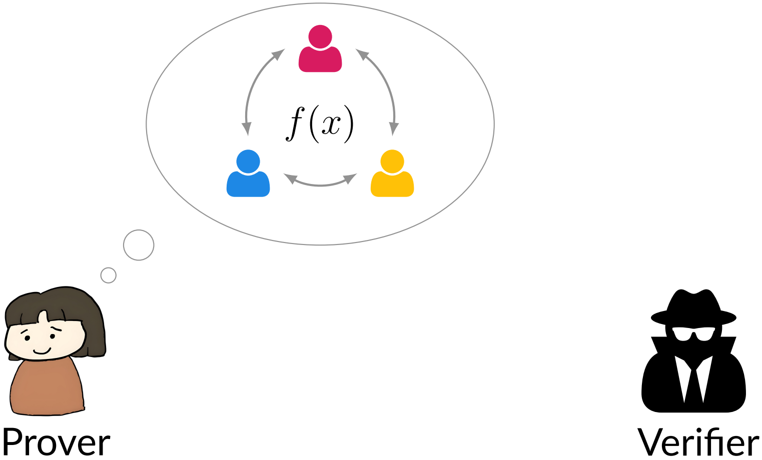
- MPCitH paradigm is to build a ZKP system by simulating an MPC protocol computing a one-way function
- Characteristics of the MPCitH-based digital signature is:
  - ✓ Security relying only on the one-wayness of the one-way function (no trapdoor)
  - ✓ Trade-off between time & size
  - ✓ Small public key and secret key
  - ✗ Relatively large signature size and sign/verify time

MPC-in-the-Head (MPCitH)

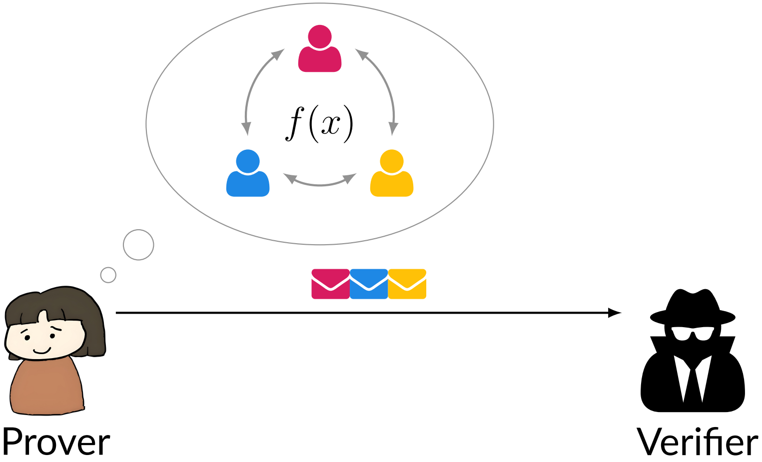
Prover

Verifier

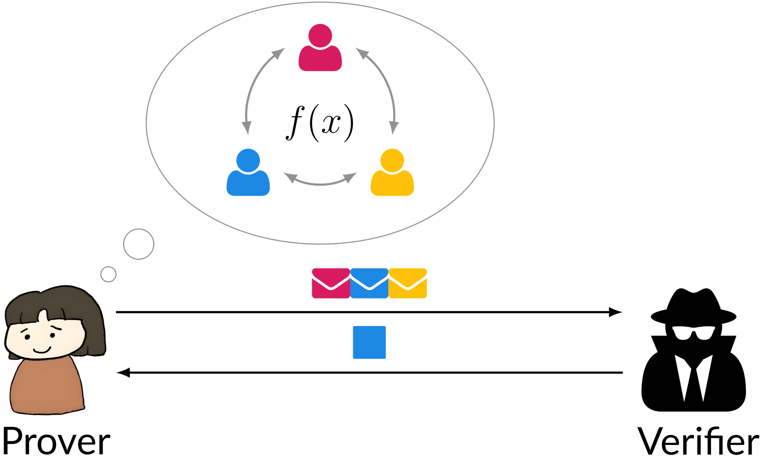
MPC-in-the-Head (MPCitH)



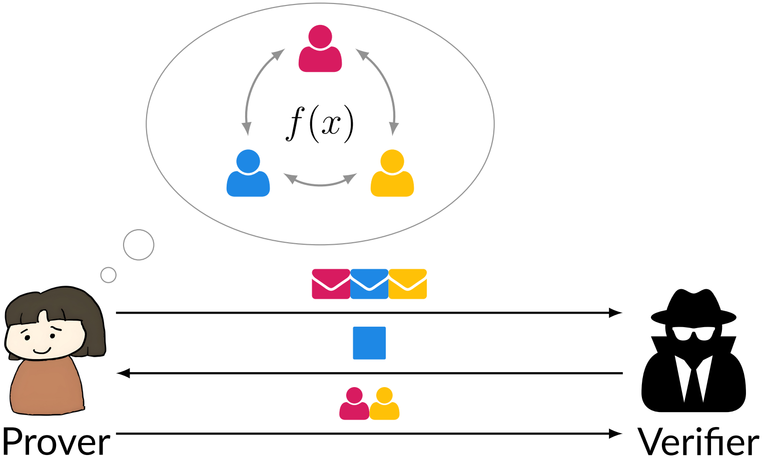
MPC-in-the-Head (MPCitH)



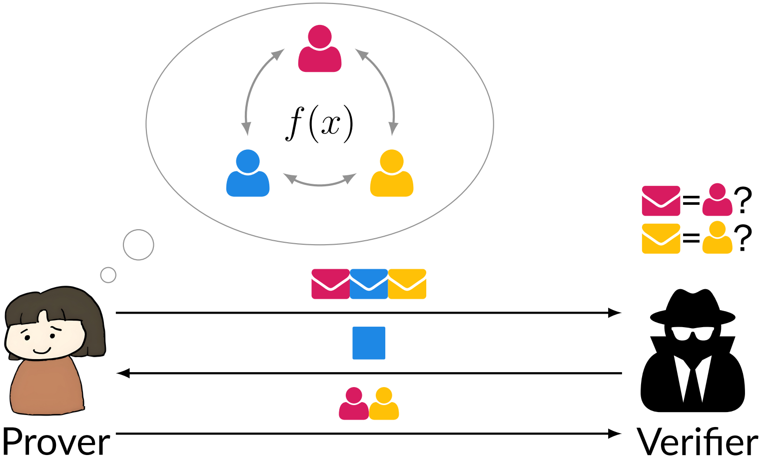
MPC-in-the-Head (MPCitH)



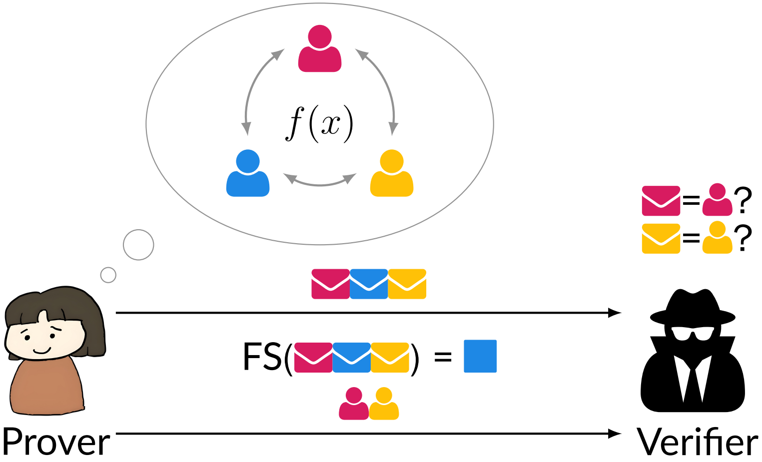
MPC-in-the-Head (MPCitH)



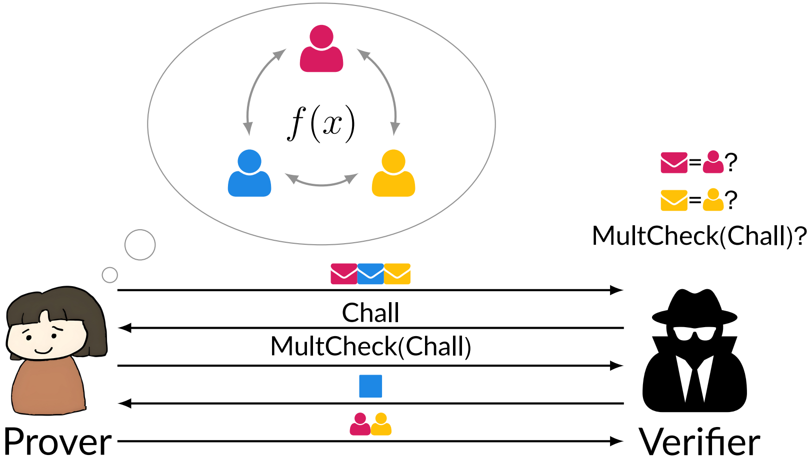
MPC-in-the-Head (MPCitH)



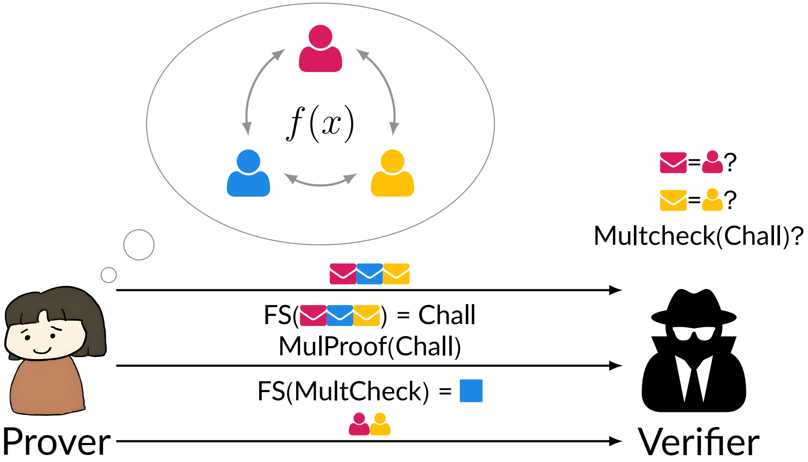
MPCitH-based Signature



Recent MPCitH



Recent MPCitH-based Signature



Toy Example

Phase	Variable	Real Value	Share					Correction
			Party 1	Party 2	Party 3	Party 4	Party 5	
Phase 1	x	3	5	6	1	3	9	1
	y	6	10	0	6	7	5	0
	z	7	9	4	1	2	7	6
	a	2	0	2	6	2	3	-
	b	5	8	4	3	0	1	-
	c	10	4	6	3	7	7	5
	com	-	$h(\mathsf{sd}_1)$	$h(\mathsf{sd}_2)$	$h(\mathsf{sd}_3)$	$h(\mathsf{sd}_4)$	$h(\mathsf{sd}_5)$	-

Phase 1

- N parties generate the shares of the another multiplication triples (a, b, c) which satisfies $ab = c$
- Each party commits to their own seeds and sends the corrections

Toy Example

Phase	Variable	Real Value	Share					Correction
			Party 1	Party 2	Party 3	Party 4	Party 5	
Phase 1	x	3	$5+1$	6	1	3	9	1
	y	6	$10+0$	0	6	7	5	0
	z	7	$9+6$	4	1	2	7	6
	a	2	0	2	6	2	3	-
	b	5	8	4	3	0	1	-
	c	10	$4+5$	6	3	7	7	5
	com	-	$h(\mathsf{sd}_1)$	$h(\mathsf{sd}_2)$	$h(\mathsf{sd}_3)$	$h(\mathsf{sd}_4)$	$h(\mathsf{sd}_5)$	-

Phase 1

- N parties generate the shares of the another multiplication triples (a, b, c) which satisfies $ab = c$
- Each party commits to their own seeds and sends the corrections

Toy Example

Phase	Variable	Real Value	Share					Correction
			Party 1	Party 2	Party 3	Party 4	Party 5	
Phase 1	x	3	$5+1$	6	1	3	9	1
	y	6	$10+0$	0	6	7	5	0
	z	7	$9+6$	4	1	2	7	6
	a	2	0	2	6	2	3	-
	b	5	8	4	3	0	1	-
	c	10	$4+5$	6	3	7	7	5
	com	-	$h(\mathsf{sd}_1)$	$h(\mathsf{sd}_2)$	$h(\mathsf{sd}_3)$	$h(\mathsf{sd}_4)$	$h(\mathsf{sd}_5)$	-
Phase 2	Random challenge $\varepsilon = 5$ from the verifier							

Phase 2

- Verifier sends random challenge ε to parties

Toy Example

Phase	Variable	Real Value	Share Party 1	Share Party 2	Share Party 3	Share Party 4	Share Party 5	Correction
Phase 1	x	3	$5+1$	6	1	3	9	1
	y	6	$10+0$	0	6	7	5	0
	z	7	$9+6$	4	1	2	7	6
	a	2	0	2	6	2	3	-
	b	5	8	4	3	0	1	-
	c	10	$4+5$	6	3	7	7	5
	com	-	$h(\mathsf{sd}_1)$	$h(\mathsf{sd}_2)$	$h(\mathsf{sd}_3)$	$h(\mathsf{sd}_4)$	$h(\mathsf{sd}_5)$	-
Phase 2	Random challenge $\varepsilon = 5$ from the verifier							
Phase 3	α	6	4	10	0	6	4	-
	β	0	7	4	9	7	6	-
	v	0	4	5	9	3	1	-

Phase 3

- The parties locally set $\alpha^{(i)} = \varepsilon \cdot x^{(i)} + a^{(i)}, \beta^{(i)} = y^{(i)} + b^{(i)}$ and broadcast them

Toy Example

Phase	Variable	Real Value	Share					Correction
			Party 1	Party 2	Party 3	Party 4	Party 5	
Phase 1	x	3	$5+1$	6	1	3	9	1
	y	6	$10+0$	0	6	7	5	0
	z	7	$9+6$	4	1	2	7	6
	a	2	0	2	6	2	3	-
	b	5	8	4	3	0	1	-
	c	10	$4+5$	6	3	7	7	5
	com	-	$h(\mathsf{sd}_1)$	$h(\mathsf{sd}_2)$	$h(\mathsf{sd}_3)$	$h(\mathsf{sd}_4)$	$h(\mathsf{sd}_5)$	-
Phase 2	Random challenge $\varepsilon = 5$ from the verifier							
Phase 3	α	6	4	10	0	6	4	-
	β	0	7	4	9	7	6	-
	v	0	4	5	9	3	1	-

Phase 3

- The parties locally set

$$v^{(i)} = \begin{cases} \varepsilon \cdot z^{(i)} - c^{(i)} + \alpha \cdot b^{(i)} + \beta \cdot a^{(i)} - \alpha \cdot \beta & \text{if } i = 1 \\ \varepsilon \cdot z^{(i)} - c^{(i)} + \alpha \cdot b^{(i)} + \beta \cdot a^{(i)} & \text{otherwise} \end{cases}$$

Toy Example

Phase	Variable	Real Value	Share					Correction
			Party 1	Party 2	Party 3	Party 4	Party 5	
Phase 1	x	3	$5+1$	6	1	3	9	1
	y	6	$10+0$	0	6	7	5	0
	z	7	$9+6$	4	1	2	7	6
	a	2	0	2	6	2	3	-
	b	5	8	4	3	0	1	-
	c	10	$4+5$	6	3	7	7	5
	com	-	$h(\mathsf{sd}_1)$	$h(\mathsf{sd}_2)$	$h(\mathsf{sd}_3)$	$h(\mathsf{sd}_4)$	$h(\mathsf{sd}_5)$	-
Phase 2	Random challenge $\varepsilon = 5$ from the verifier							
Phase 3	α	6	4	10	0	6	4	-
	β	0	7	4	9	7	6	-
	v	0	4	5	9	3	1	-

Phase 3

- Each party opens $v^{(i)}$ to compute v
- If $ab = c$ and $xy = z$, then $v = 0$

Toy Example

Phase	Variable	Real Value	Share					Correction
			Party 1	Party 2	Party 3	Party 4	Party 5	
Phase 1	x	3	$5+1$	6	1	3	9	1
	y	6	$10+0$	0	6	7	5	0
	z	7	$9+6$	4	1	2	7	6
	a	2	0	2	6	2	3	-
	b	5	8	4	3	0	1	-
	c	10	$4+5$	6	3	7	7	5
	com	-	$h(\mathsf{sd}_1)$	$h(\mathsf{sd}_2)$	$h(\mathsf{sd}_3)$	$h(\mathsf{sd}_4)$	$h(\mathsf{sd}_5)$	-
Phase 2	Random challenge $\varepsilon = 5$ from the verifier							
Phase 3	α	6	4	10	0	6	4	-
	β	0	7	4	9	7	6	-
	v	0	4	5	9	3	1	-
Phase 4	Random challenge $\bar{i} = 4$ from the verifier							

Phase 4

- Verifier sends a hidden party index $\bar{i}$ to parties

Toy Example

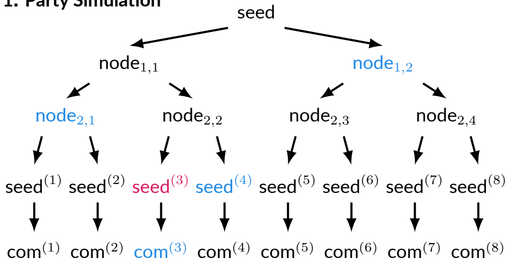
Phase	Variable	Real Value	Share					Correction
			Party 1	Party 2	Party 3	Party 4	Party 5	
Phase 1	x	3	$5+1$	6	1	3	9	1
	y	6	$10+0$	0	6	7	5	0
	z	7	$9+6$	4	1	2	7	6
	a	2	0	2	6	2	3	-
	b	5	8	4	3	0	1	-
	c	10	$4+5$	6	3	7	7	5
	com	-	$h(\mathsf{sd}_1)$	$h(\mathsf{sd}_2)$	$h(\mathsf{sd}_3)$	$h(\mathsf{sd}_4)$	$h(\mathsf{sd}_5)$	-
Phase 2	Random challenge $\varepsilon = 5$ from the verifier							
Phase 3	α	6	4	10	0	6	4	-
	β	0	7	4	9	7	6	-
	v	0	4	5	9	3	1	-
Phase 4	Random challenge $\bar{i} = 4$ from the verifier							
Phase 5	Open all parties except $\bar{i}$-th party and check consistency							

Phase 5

- Each party $i \in [N] \backslash \{\bar{i}\}$ sends $x^{(i)}, y^{(i)}, z^{(i)}, a^{(i)}, b^{(i)}$, and $c^{(i)}$ to verifier
- Verifier checks the consistency of the received shares

Detailed MPCitH

1. Party Simulation



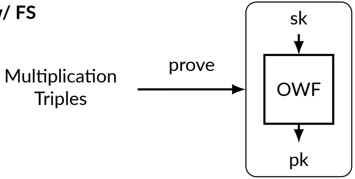
2. Multiplication triple generation

$\mathsf{PRG}(\mathsf{seed}^{(1)}) =$
$(w_1^{(1)}, \dots, w_C^{(1)}, a_1^{(1)}, \dots, a_C^{(1)}, b_1^{(1)}, \dots, b_C^{(1)}, c^{(1)})$

$\vdots$

$\mathsf{PRG}(\mathsf{seed}^{(N)}) =$
$(w_1^{(N)}, \dots, w_C^{(N)}, a_1^{(N)}, \dots, a_C^{(N)}, b_1^{(N)}, \dots, b_C^{(N)}, c^{(N)})$

3. Proof w/ FS

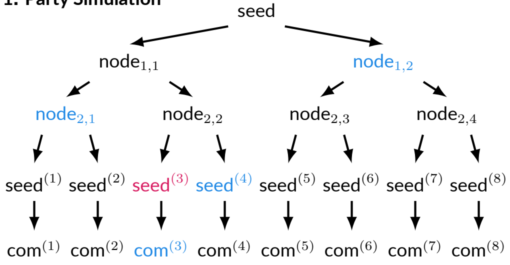


4. Party Opening

Choose i using FS!

Detailed MPCitH

1. Party Simulation



2. Multiplication triple generation

$\mathsf{PRG}(\mathsf{seed}^{(1)}) =$
$(w_1^{(1)}, \dots, w_C^{(1)}, a_1^{(1)}, \dots, a_C^{(1)}, b_1^{(1)}, \dots, b_C^{(1)}, c^{(1)})$
$\vdots$
$\mathsf{PRG}(\mathsf{seed}^{(N)}) =$
$(w_1^{(N)}, \dots, w_C^{(N)}, a_1^{(N)}, \dots, a_C^{(N)}, b_1^{(N)}, \dots, b_C^{(N)}, c^{(N)})$

3. Proof w/ FS

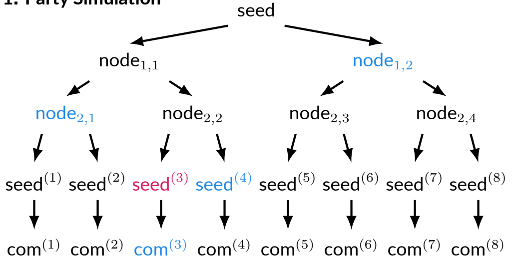
Proving $x \cdot y = z$
$\alpha^{(i)} = \epsilon \cdot x^{(i)} + a^{(i)}$
$\beta^{(i)} = y^{(i)} + b^{(i)}$
Broadcast α and β
Check $\sum_i (\epsilon z^{(i)} - c^{(i)} + \alpha b^{(i)} + \beta a^{(i)} - \alpha\beta) = 0$
where $ab = c$

4. Party Opening

Choose i using FS!

Detailed MPCitH

1. Party Simulation



2. Multiplication triple generation

PRG(seed$^{(1)}$) $=$
$(w_1^{(1)}, \ldots, w_C^{(1)}, a_1^{(1)}, \ldots, a_C^{(1)}, b_1^{(1)}, \ldots, b_C^{(1)}, c^{(1)})$

$\vdots$

PRG(seed$^{(N)}$) $=$
$(w_1^{(N)}, \ldots, w_C^{(N)}, a_1^{(N)}, \ldots, a_C^{(N)}, b_1^{(N)}, \ldots, b_C^{(N)}, c^{(N)})$

3. Proof w/ FS

Proving $x_j \cdot y_j = z_j$

$\alpha_j^{(i)} = \epsilon_j \cdot x_j^{(i)} + a_j^{(i)}$

$\beta_j^{(i)} = y_j^{(i)} + b_j^{(i)}$

Broadcast α_j and β_j

Check $\sum_i (\sum_j (\epsilon_j z_j^{(i)} + \alpha_j b_j^{(i)} + \beta a_j^{(i)} - \alpha_j \beta_j) - c^{(i)}) = 0$

where $\sum_j a_j b_j = c$

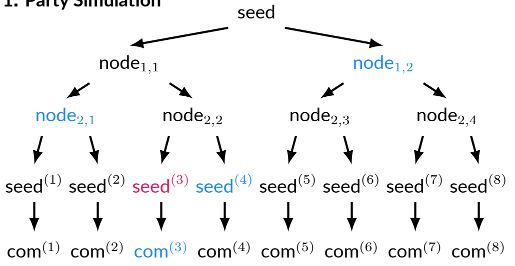
4. Party Opening

Choose i using FS!

AIMer

AIMer v1.0

1. Party Simulation



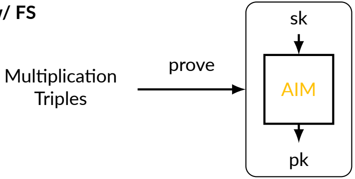
2. Multiplication triple generation

$\mathrm{PRG}(\mathrm{seed}^{(1)}) =$
$(w_1^{(1)}, \ldots, w_C^{(1)}, a_1^{(1)}, \ldots, a_C^{(1)}, b_1^{(1)}, \ldots, b_C^{(1)}, c^{(1)})$

$\vdots$

$\mathrm{PRG}(\mathrm{seed}^{(N)}) =$
$(w_1^{(N)}, \ldots, w_C^{(N)}, a_1^{(N)}, \ldots, a_C^{(N)}, b_1^{(N)}, \ldots, b_C^{(N)}, c^{(N)})$

3. Proof w/ FS

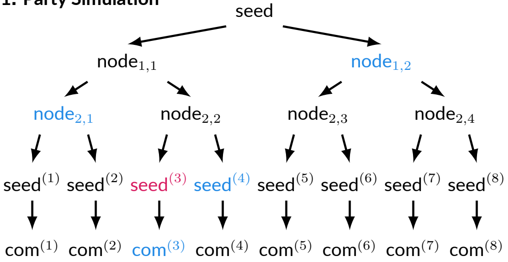


4. Party Opening

Choose i using FS!

AIMer v2.0

1. Party Simulation



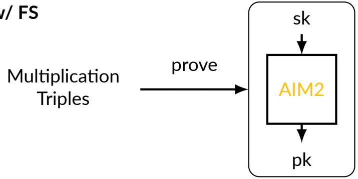
2. Multiplication triple generation

$$\mathrm{PRG}(\mathsf{seed}^{(1)}) = (w_1^{(1)}, \ldots, w_C^{(1)}, a_1^{(1)}, \ldots, a_C^{(1)}, b_1^{(1)}, \ldots, b_C^{(1)}, c^{(1)})$$

$$\vdots$$

$$\mathrm{PRG}(\mathsf{seed}^{(N)}) = (w_1^{(N)}, \ldots, w_C^{(N)}, a_1^{(N)}, \ldots, a_C^{(N)}, b_1^{(N)}, \ldots, b_C^{(N)}, c^{(N)})$$

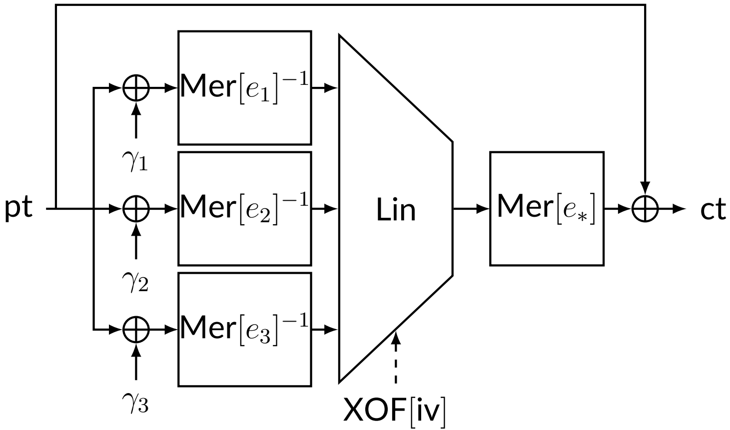
3. Proof w/ FS



4. Party Opening

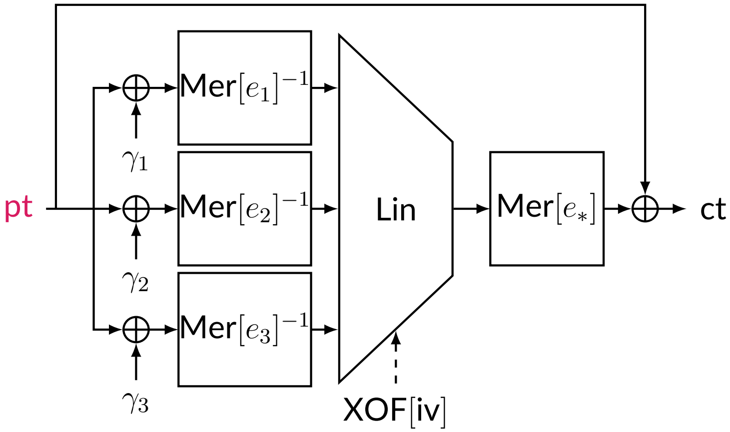
Choose i using FS!

AIM2



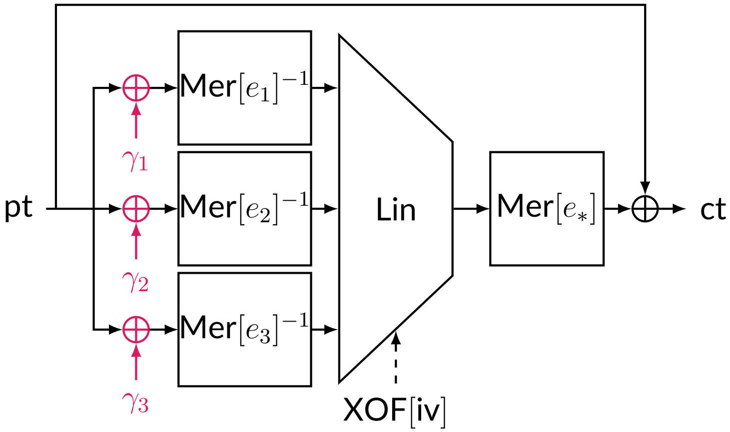
pt
Mer[e_1]$^{-1}$
Mer[e_2]$^{-1}$
Mer[e_3]$^{-1}$
γ_1
γ_2
γ_3
Lin
XOF[iv]
Mer[e_*]
ct

AIM2



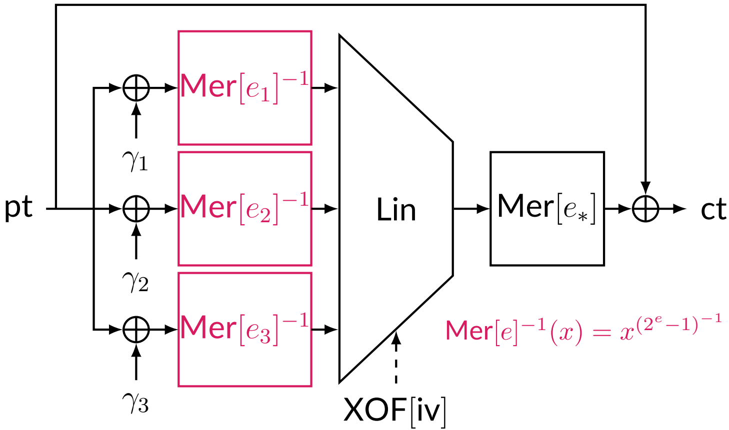
pt
Mer$[e_1]^{-1}$
Mer$[e_2]^{-1}$
Mer$[e_3]^{-1}$
γ_1
γ_2
γ_3
Lin
XOF[iv]
Mer$[e_*]$
ct

AIM2



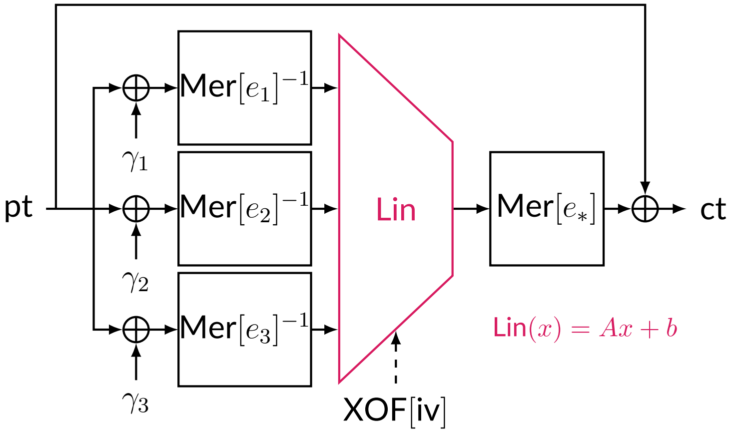
pt
$\mathrm{Mer}[e_1]^{-1}$
$\mathrm{Mer}[e_2]^{-1}$
$\mathrm{Mer}[e_3]^{-1}$
γ_1
γ_2
γ_3
Lin
XOF[iv]
$\mathrm{Mer}[e_*]$
ct

AIM2



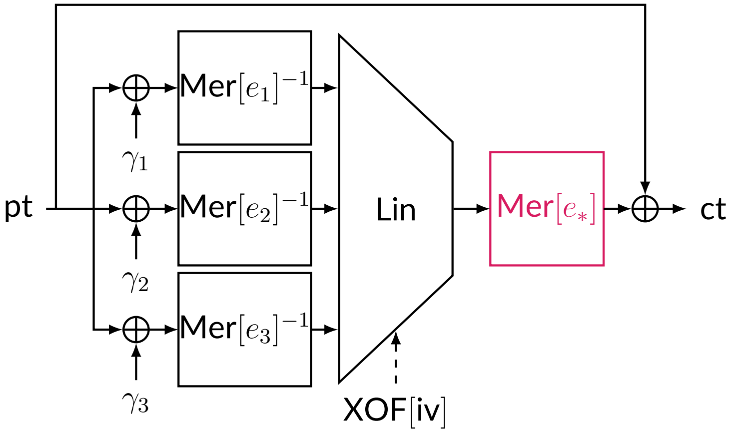
pt
γ_1
γ_2
γ_3
$\mathsf{Mer}[e_1]^{-1}$
$\mathsf{Mer}[e_2]^{-1}$
$\mathsf{Mer}[e_3]^{-1}$
Lin
XOF[iv]
$\mathsf{Mer}[e_*]$
ct
$\mathsf{Mer}[e]^{-1}(x) = x^{(2^e-1)^{-1}}$

AIM2



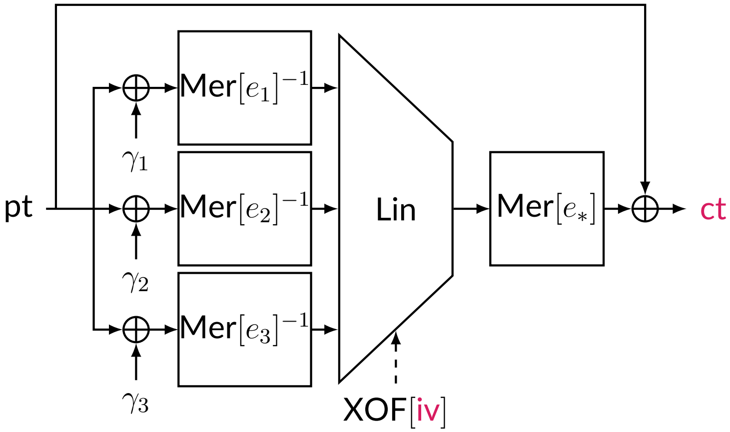
pt
γ_1
γ_2
γ_3
$\mathsf{Mer}[e_1]^{-1}$
$\mathsf{Mer}[e_2]^{-1}$
$\mathsf{Mer}[e_3]^{-1}$
Lin
XOF[iv]
$\mathsf{Mer}[e_*]$
ct
$\mathsf{Lin}(x) = Ax + b$

AIM2



pt
γ_1
γ_2
γ_3
$\mathsf{Mer}[e_1]^{-1}$
$\mathsf{Mer}[e_2]^{-1}$
$\mathsf{Mer}[e_3]^{-1}$
Lin
XOF[iv]
$\mathsf{Mer}[e_*]$
ct

AIM2



pt
γ_1
γ_2
γ_3
$\text{Mer}[e_1]^{-1}$
$\text{Mer}[e_2]^{-1}$
$\text{Mer}[e_3]^{-1}$
Lin
XOF[iv]
$\text{Mer}[e_*]$
ct

Advantage & Limitation

- Advantages
  1. Short key size
  2. Security only relies on symmetric primitives
  3. Most efficient among schemes relying only on symmetric primitives
- Limitations
  1. Modest performance
  2. Relatively new primitive
     - * But multiple cryptanalysts have admitted that AIM2 is secure against state-of-the-art cryptanalytic techniques.

Security

- Security of AIMer is reduced to preimage resistance of AIM2
- Conventional symmetric key cryptanalysis cannot be applied to AIM2
    - Single input-output assumption
- We prevent algebraic attacks with the utmost effort
    - Sufficient security margin despite of radical assumption
    - We brute-forced all the derivable quadratic system of AIM2
    - All the attacks done for symmetric primitives with large S-boxes are considered

Performance

AIMer enjoys balanced performance (all-rounder).

Scheme	Size (B)			Time (cycle)		
	sk	pk	sig	KeyGen	Sign	Verify
Dilithium	2,528	1,312	2,420			
Falcon	1,281	897	666			
SPHINCS+-f	64	32	17.1K			
HAETAE	1,408	992	1,474			
NCC-Sign-tri	2,400	1,760	2,912			
MQ-Sign-LR	161K	328K	134			
AIMer-f	48	32	5,888			

SUPERCOP result (Zen 4), Category 1 or 2, median speed

Performance

AIMer enjoys balanced performance (all-rounder).

Scheme	Size (B)			Time (cycle)		
	sk	pk	sig	KeyGen	Sign	Verify
Dilithium	2,528	1,312	2,420	62K	149K	70K
Falcon	1,281	897	666	15.6M*	331K*	63K*
SPHINCS+-f	64	32	17.1K	1.23M*	5.65M*	6.26M*
HAETAE	1,408	992	1,474	437K	1.13M	100K
NCC-Sign-tri	2,400	1,760	2,912	197K	295K	196K
MQ-Sign-LR	161K	328K	134	5.60M*	67K*	35K*
AIMer-f	48	32	5,888	40K	889K	898K

* Not intend to be constant-time

SUPERCOP result (Zen 4), Category 1 or 2, median speed

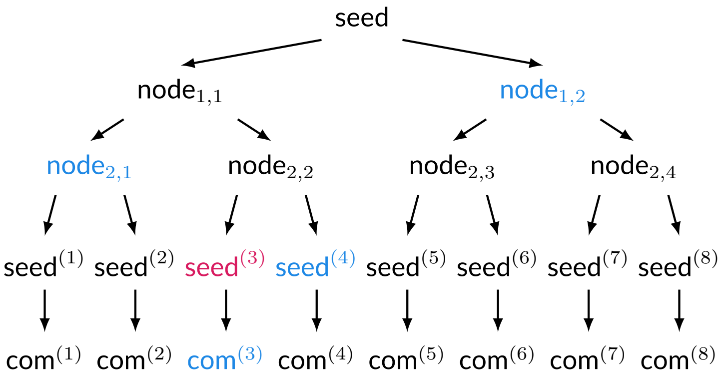
Implementations

- Github repository at (`https://github.com/samsungsds-research-papers/AIMer`)
- Reference (C standalone)
- Optimized (AVX2)
- ARM64 + SHA3 (only in Apple M series)
- Constrained memory ($\leq$ 110 KB)
- ARM Cortex-M4 (in `pqm4` library)

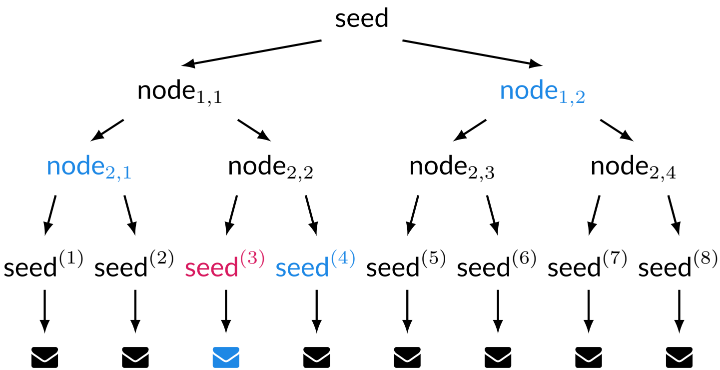
Relaxed Vector Commitment for Shorter Signatures (Eurocrypt 2025)

Vector Commitment

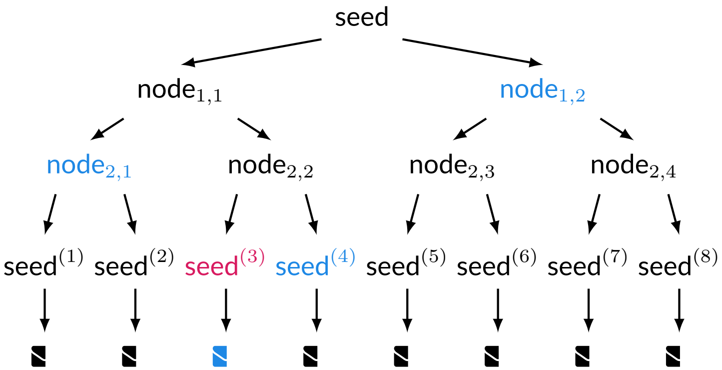
Vector Commitment



Vector Commitment



Vector Semi-Commitment

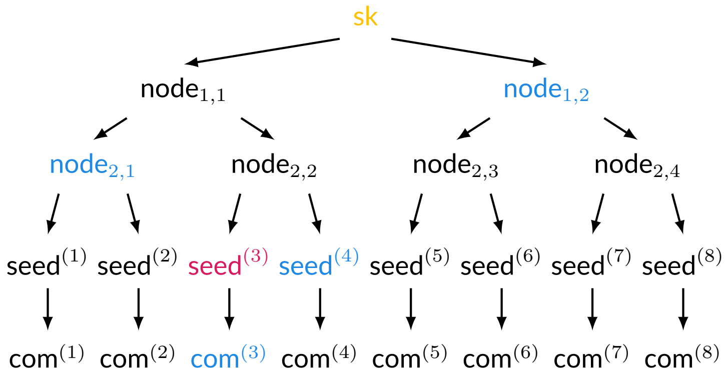


Application of VSC (rMPCitH)

1. Halved commitment size
2. GGM tree $\rightarrow$ correlated GGM tree

Application of VSC (rMPCitH)

1. Halved commitment size
2. GGM tree $\rightarrow$ correlated GGM tree

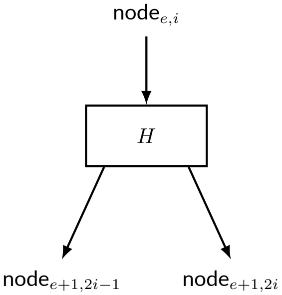


Application of VSC (rMPCitH)

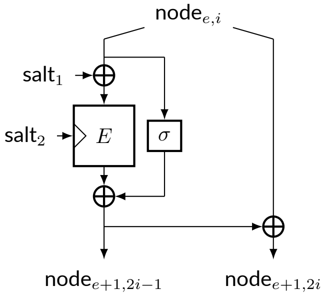
1. Halved commitment size
2. GGM tree $\rightarrow$ correlated GGM tree
3. Random oracle model $\rightarrow$ ideal cipher model

Application of VSC (rMPCitH)

1. Halved commitment size
2. GGM tree $\rightarrow$ correlated GGM tree
3. Random oracle model $\rightarrow$ ideal cipher model



Double-length PRG



IC-VSC

Performance

Scheme	$\|pk\|$ (B)	$\|sig\|$ (B)	Sign (Kc)	Verify (Kc)
Dilithium2	1,312	2,420	162	57
SPHINCS$^+$-128f*	32	17,088	38,216	2,158
SPHINCS$^+$-128s*	32	7,856	748,053	799
SDitH-Hypercube-gf256	132	8,496	20,820	10,935
FAEST-128f	32	6,336	2,387	2,344
FAEST-128s	32	5,006	20,926	20,936
AIMer-v2.0-128f	32	5,888	788	752
AIMer-v2.0-128s	32	4,160	5,926	5,812
rAIMer-128f	32	4,848	421	395
rAIMer-128s	32	3,632	2,826	2,730

*: -SHAKE256-simple

Conclusion

- MPC-in-the-Head is a paradigm to construct ZKP from MPC, which does not require a trapdoor
- AIM2 is a one-way function designed for efficiency in MPCitH paradigm and security against algebraic attacks
- AIMer is a digital signature scheme proving one-way function AIM within the MPCitH paradigm
- Research on MPCitH-based (including TCitH, VOLEitH) signature is not yet finished

Thank you!
Check out our website!

Attribution

- Illustrations at the very beginning was created using fontawesome latex package (`https://github.com/xdanaux/fontawesome-latex`).
- SUPERCOP result can be found in `https://bench.cr.yp.to/results-sign/amd64-hertz.html`.