AIMer v2.1 and Beyond

2025 KMS Spring Meeting

Seongkwang Kim[1] Jincheol Ha[2] Mincheol Son[2]
Byeonghak Lee[1] Dukjae Moon[1] Joohee Lee[3] Sangyub Lee[1]
Jihoon Kwon[1] Jihoon Cho[1] Hyojin Yoon[1] Jooyoung Lee[2]

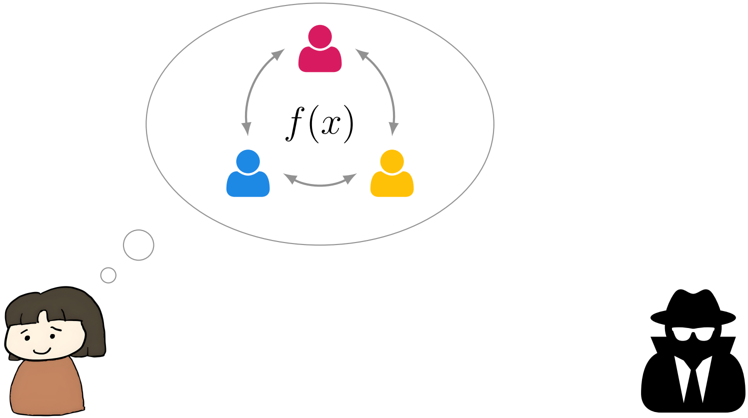
[1]Samsung SDS [2]KAIST [3]Sungshin Women's University

MPC-in-the-Head (MPCitH)

Prover

Verifier

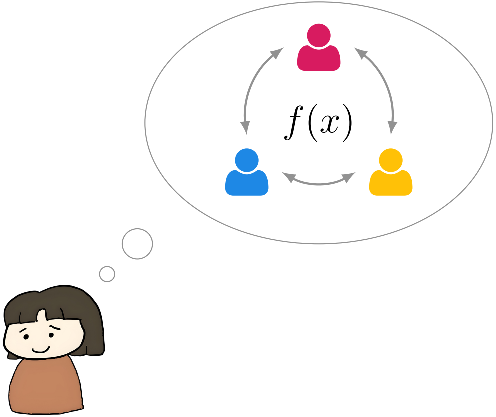
MPC-in-the-Head (MPCitH)



Prover

Verifier

MPC-in-the-Head (MPCitH)

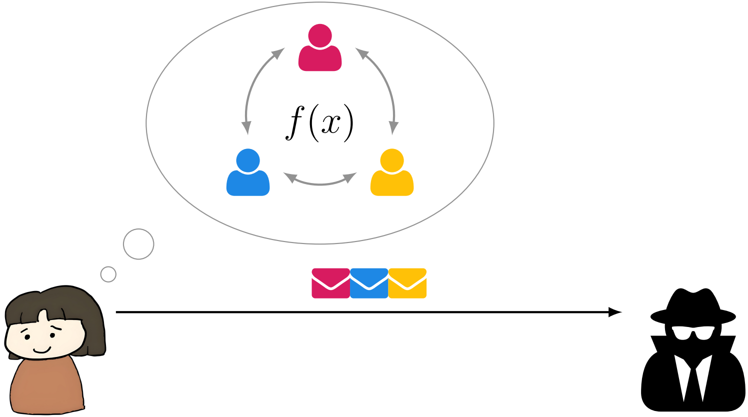


Which MPC?
Garbled circuit
GMW
Beaver triple

Prover

Verifier

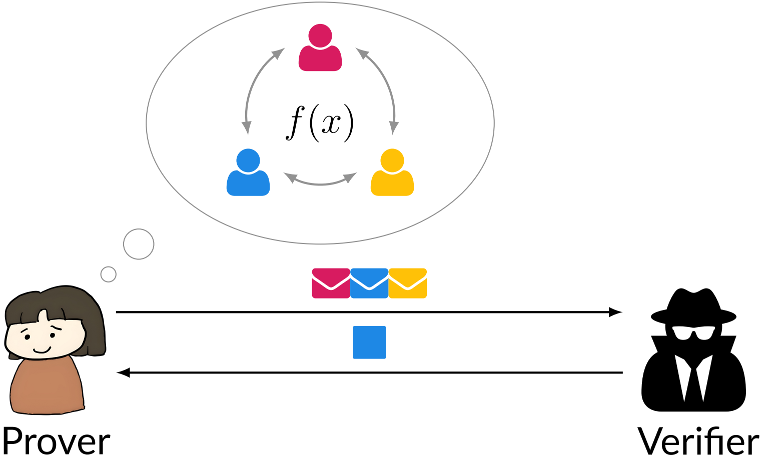
MPC-in-the-Head (MPCitH)



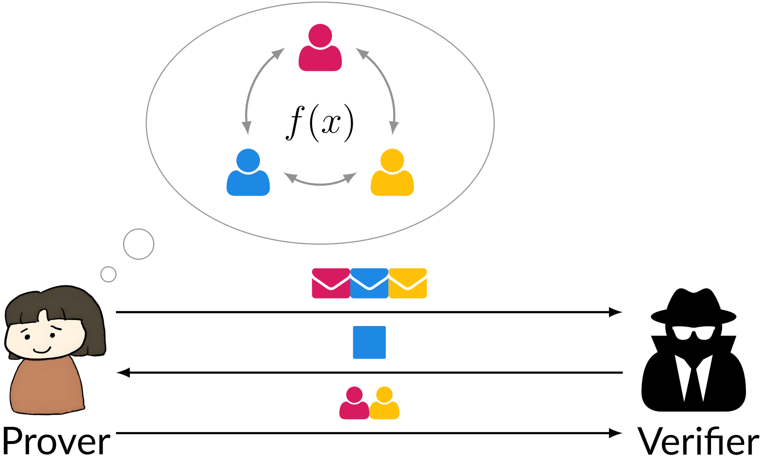
Prover

Verifier

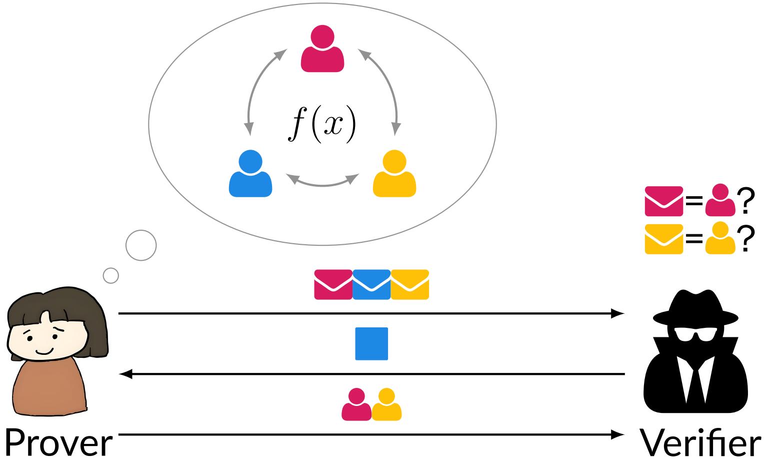
MPC-in-the-Head (MPCitH)



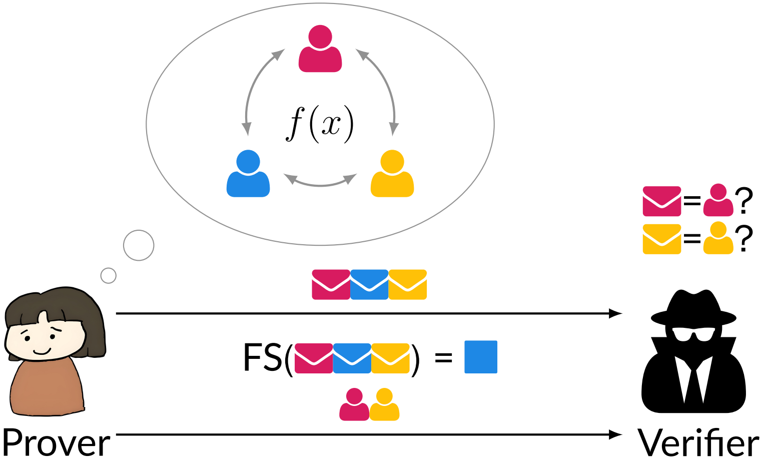
MPC-in-the-Head (MPCitH)



MPC-in-the-Head (MPCitH)



MPCitH-based Signature



$f(x)$

= ?

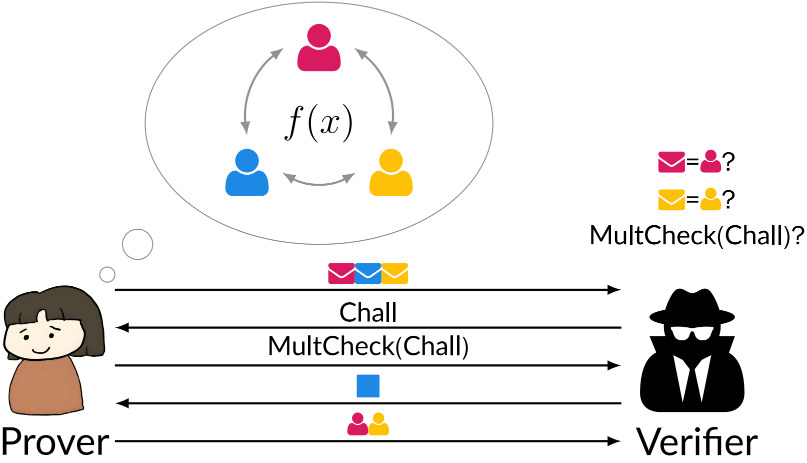
= ?

FS( ) =

Prover

Verifier

Recent MPCitH



$f(x)$

=?

=?

MultCheck(Chall)?

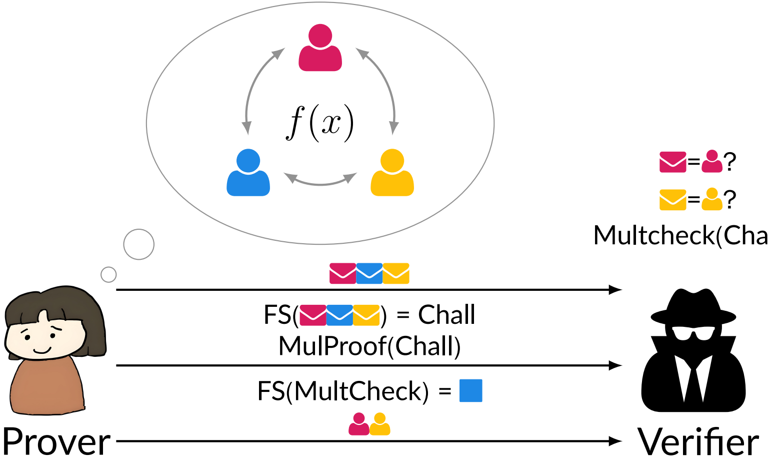
Chall

MultCheck(Chall)

Prover

Verifier

Recent MPCitH-based Signature



$f(x)$

✉=👤?

✉=👤?

Multcheck(Chall)?

✉✉✉

FS(✉✉✉) = Chall

MulProof(Chall)

FS(MultCheck) = ■

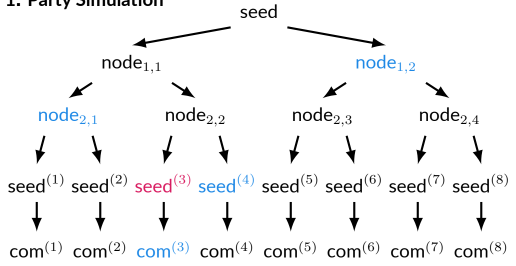
👤👤

Prover

Verifier

Detailed MPCitH

1. Party Simulation



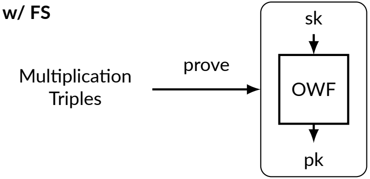
2. Multiplication triple generation

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

$$\vdots$$

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

3. Proof w/ FS

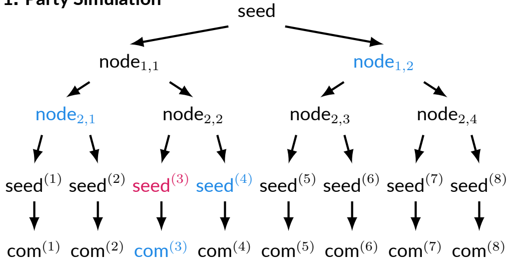


4. Party Opening

Choose i using FS!

Detailed MPCitH

1. Party Simulation



seed
$\text{node}_{1,1}$
$\text{node}_{1,2}$
$\text{node}_{2,1}$
$\text{node}_{2,2}$
$\text{node}_{2,3}$
$\text{node}_{2,4}$
$\text{seed}^{(1)}$ $\text{seed}^{(2)}$ $\text{seed}^{(3)}$ $\text{seed}^{(4)}$ $\text{seed}^{(5)}$ $\text{seed}^{(6)}$ $\text{seed}^{(7)}$ $\text{seed}^{(8)}$
$\text{com}^{(1)}$ $\text{com}^{(2)}$ $\text{com}^{(3)}$ $\text{com}^{(4)}$ $\text{com}^{(5)}$ $\text{com}^{(6)}$ $\text{com}^{(7)}$ $\text{com}^{(8)}$

2. Multiplication triple generation

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

3. Proof w/ FS

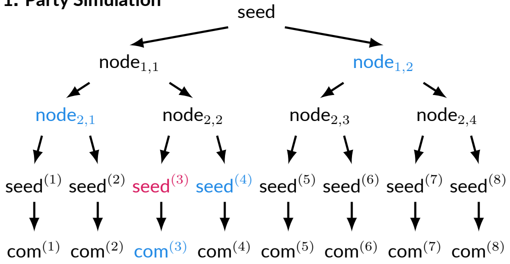
Proving $x \cdot y = z$
$\alpha^{(i)} = \epsilon \cdot x^{(i)} + a^{(i)}$
$\beta^{(i)} = y^{(i)} + b^{(i)}$
Broadcast α and β
Check $\sum_i (\epsilon z^{(i)} - c^{(i)} + \alpha b^{(i)} + \beta a^{(i)} - \alpha\beta) = 0$
where $ab = c$

4. Party Opening

Choose i using FS!

Detailed MPCitH

1. Party Simulation



seed

$\text{node}_{1,1}$ $\text{node}_{1,2}$

$\text{node}_{2,1}$ $\text{node}_{2,2}$ $\text{node}_{2,3}$ $\text{node}_{2,4}$

$\text{seed}^{(1)}$ $\text{seed}^{(2)}$ $\text{seed}^{(3)}$ $\text{seed}^{(4)}$ $\text{seed}^{(5)}$ $\text{seed}^{(6)}$ $\text{seed}^{(7)}$ $\text{seed}^{(8)}$

$\text{com}^{(1)}$ $\text{com}^{(2)}$ $\text{com}^{(3)}$ $\text{com}^{(4)}$ $\text{com}^{(5)}$ $\text{com}^{(6)}$ $\text{com}^{(7)}$ $\text{com}^{(8)}$

2. Multiplication triple generation

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

$\vdots$

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

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

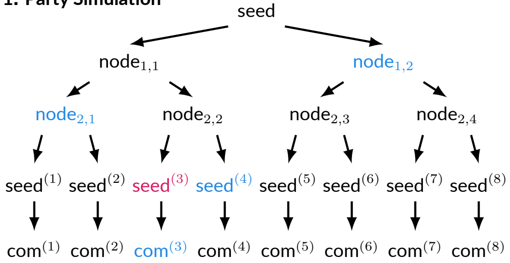
where $\sum_j a_j b_j = c$

4. Party Opening

Choose i using FS!

AIMer v1.0

1. Party Simulation


seed
$\text{node}_{1,1}$
$\text{node}_{1,2}$
$\text{node}_{2,1}$
$\text{node}_{2,2}$
$\text{node}_{2,3}$
$\text{node}_{2,4}$
$\text{seed}^{(1)}$ $\text{seed}^{(2)}$ $\text{seed}^{(3)}$ $\text{seed}^{(4)}$ $\text{seed}^{(5)}$ $\text{seed}^{(6)}$ $\text{seed}^{(7)}$ $\text{seed}^{(8)}$
$\text{com}^{(1)}$ $\text{com}^{(2)}$ $\text{com}^{(3)}$ $\text{com}^{(4)}$ $\text{com}^{(5)}$ $\text{com}^{(6)}$ $\text{com}^{(7)}$ $\text{com}^{(8)}$

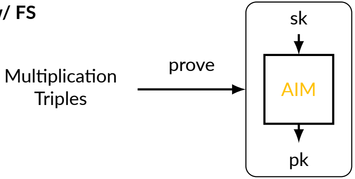
2. Multiplication triple generation

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

$$\vdots$$

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

3. Proof w/ FS

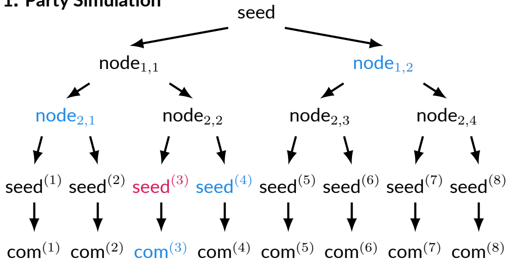


4. Party Opening

Choose i using FS!

AIMer v2.0

1. Party Simulation



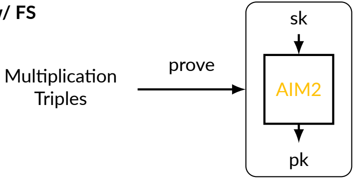
2. Multiplication triple generation

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

$$\vdots$$

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

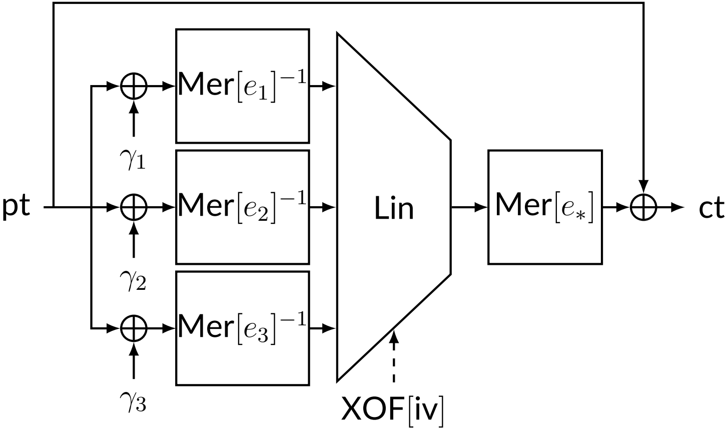
3. Proof w/ FS



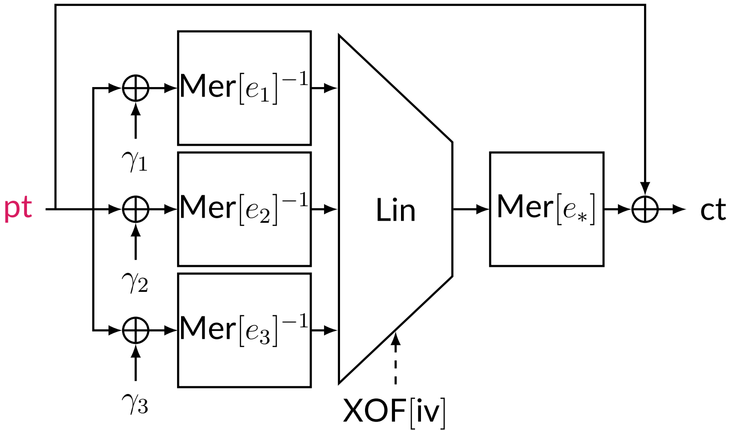
4. Party Opening

Choose i using FS!

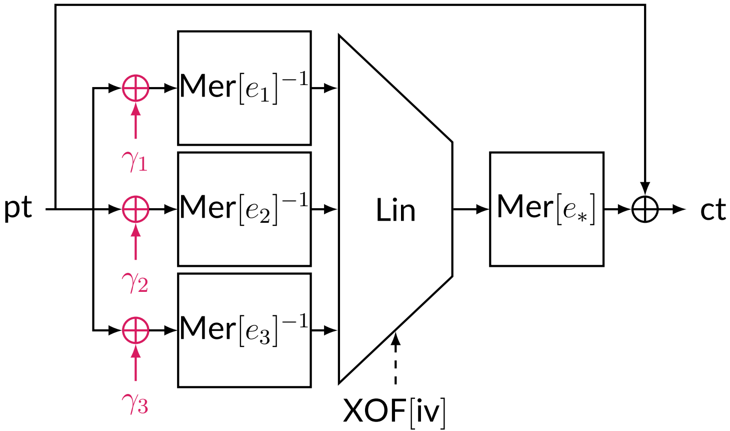
AIM2


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

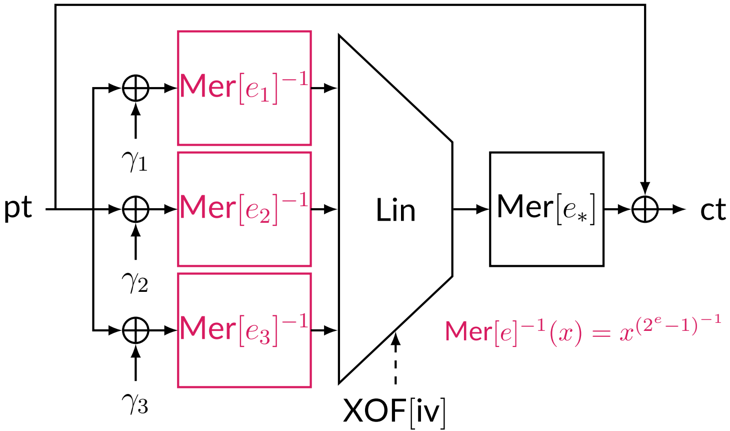
AIM2


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

AIM2

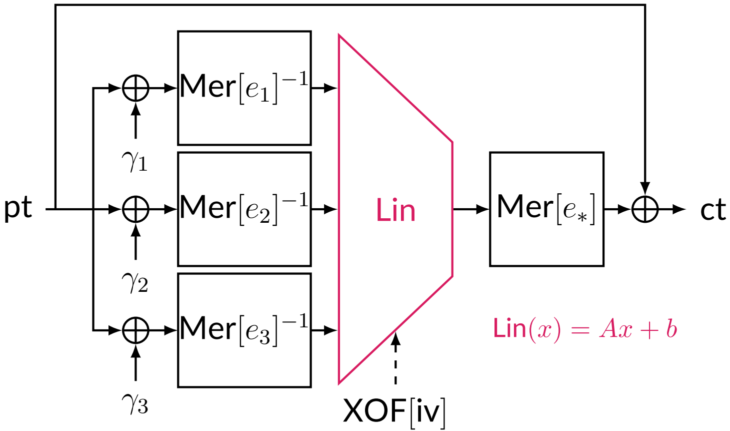

pt
γ_1
γ_2
γ_3
$\mathsf{Mer}[e_1]^{-1}$
$\mathsf{Mer}[e_2]^{-1}$
$\mathsf{Mer}[e_3]^{-1}$
Lin
XOF[iv]
$\mathsf{Mer}[e_*]$
ct

AIM2



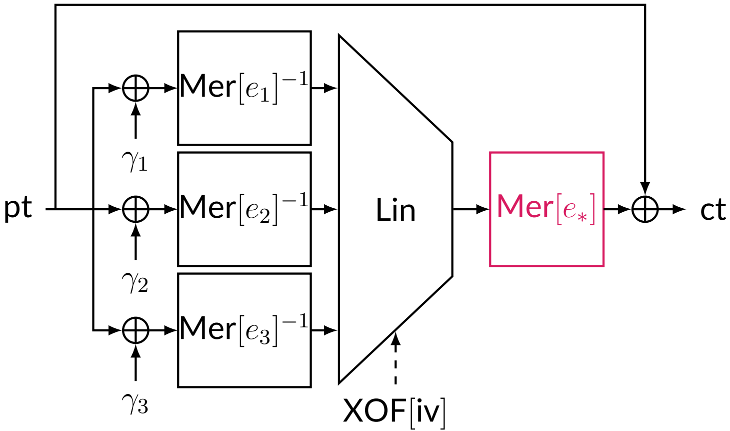
$\mathsf{Mer}[e]^{-1}(x) = x^{(2^e-1)^{-1}}$

AIM2

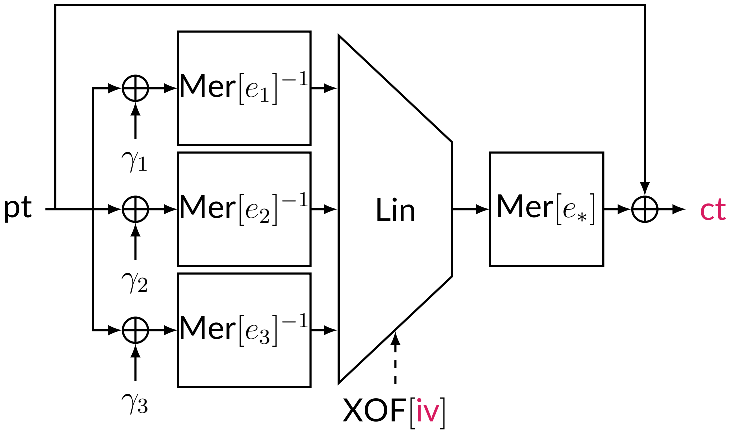


$\text{Lin}(x) = Ax + b$

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

Advantage & Limitation

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

Security

Scheme	Type	#Var	Variables	(#Eq, Deg)	Complexity		
					k	d_{reg}	Time (bits)
AIM2-I	S_1	n	t_1	$(n, 60)$	-	-	-
	S_2	$2n$	t_1, t_2	$(3n, 2)$	62	15	207.9
	S_{quad}	$3n$	x, t_1, t_2	$(12n, 2)$	0	16	185.3
AIM2-III	S_1	n	x	$(2n, 114)$	-	-	-
	S_2	$2n$	t_1, t_2	$(3n, 2)$	100	20	301.9
	S_{quad}	$3n$	x, t_1, t_2	$(12n, 2)$	0	22	262.4
AIM2-V	S_1	n	x	$(2n, 172)$	-	-	-
	S_2	$2n$	t_2, z	$(n, 2) + (2n, 38)$	253	30	513.5
	S_3	$3n$	t_1, t_2, t_3	$(6n, 2)$	2	47	503.7
	S_{quad}	$4n$	x, t_1, t_2, t_3	$(18n, 2)$	9	32	411.4

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

History: AIMer v0.9 (Oct. 2022)

History: AIMer v0.9 (Oct. 2022)

Algorithm		Implementation	Security
Symmetric	Protocol		
AIM	BN++	C standalone	Birthday-bound

History: AIMer v1.0 (Jun. 2023)

Algorithm		Implementation	Security
Symmetric	Protocol		
AIM	BN++ Merge hash Domain sep.	C standalone AVX2	Birthday-bound

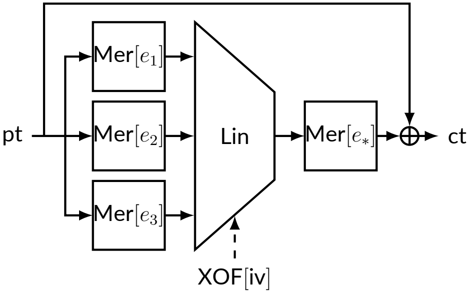
History: AIMer v1.0 (Sep. 2023)

Algorithm		Implementation	Security
Symmetric	Protocol		
~~AIM~~ Attack AIM2	BN++ Merge hash Domain sep.	C standalone AVX2	Birthday-bound

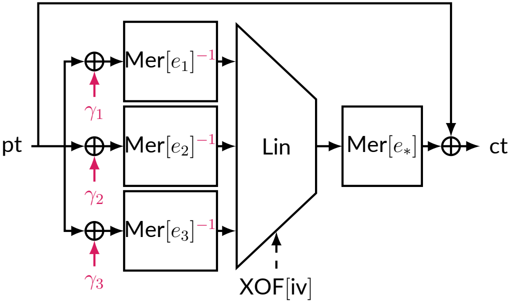
History: AIMer v2.0 (Feb. 2024)

Algorithm		Implementation	Security
Symmetric	Protocol		
~~AIM~~	BN++	C standalone	~~Birthday-bound~~
~~Attack~~	Merge hash	AVX2	Full-bound
AIM2	Domain sep.	ARM64	
	Half salt		
	Prehashing		

History: AIMer v2.0 (Feb. 2024)



AIM1



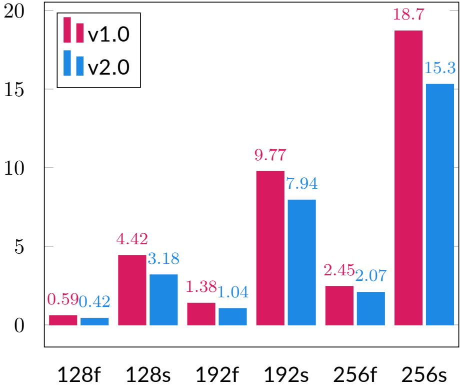
AIM2

History: AIMer v2.0 (Feb. 2024)

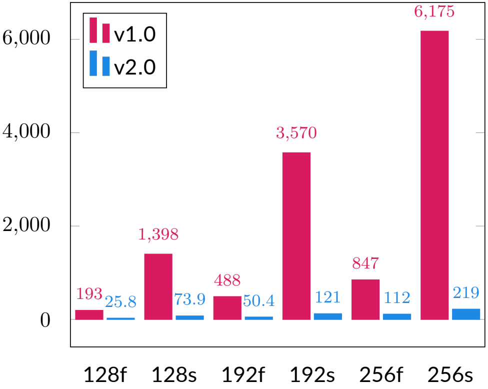
Scheme	λ	n	ℓ	e_1	e_2	e_3	e_*
AIM-I	128	128	2	3	27	-	5
AIM-III	192	192	2	5	29	-	7
AIM-V	256	256	3	3	53	7	5

Scheme	λ	n	ℓ	e_1	e_2	e_3	e_*
AIM2-I	128	128	2	49	91	-	3
AIM2-III	192	192	2	17	47	-	5
AIM2-V	256	256	3	11	141	7	3

History: AIMer v2.0 (Feb. 2024)



Signing time comparison in ms (AVX2)



Memory usage comparison for verification in KB

History: AIMer v2.1 (Aug. 2024)

Algorithm		Implementation	Security
Symmetric	Protocol		
~~AIM~~	BN++	C standalone	~~Birthday-bound~~
~~Attack~~	Merge hash	AVX2	Full-bound
AIM2	Domain sep.	ARM64 + SHA3	
	Half salt	ARM Cortex-M4	
	Prehashing	PQClean	
		Constrained mem.	
		TIMECOP	

Lesson Learned from Standardization

- Conservative security first
  - Old security assumption preferred
  - Simple security proof preferred

Lesson Learned from Standardization

- Conservative security first
  - Old security assumption preferred
  - Simple security proof preferred
- So many people are needed than expected
  - Algorithm makers, cryptanalysts, (quantum) provable security experts, side-channel analysts, implementation experts on many different platforms, languages, and protocols, …

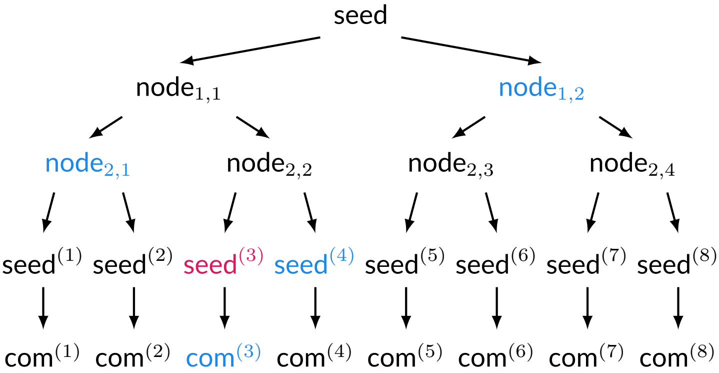
Lesson Learned from Standardization

- Conservative security first
  - Old security assumption preferred
  - Simple security proof preferred
- So many people are needed than expected
  - Algorithm makers, cryptanalysts, (quantum) provable security experts, side-channel analysts, implementation experts on many different platforms, languages, and protocols, …
- Proper marketing required
  - If security, efficiency, and simplicity of my scheme is the best, then anything does not matter
  - Otherwise, where can my scheme fit into?
  - Protocol (TLS, IPSec, SSH, DNSSEC), security assumption (lattice, isogeny, MQ, code), constrained resources, …

Relaxed Vector Commitment for Shorter Signatures (Eurocrypt 2025)

Vector Commitment

Vector Commitment



seed

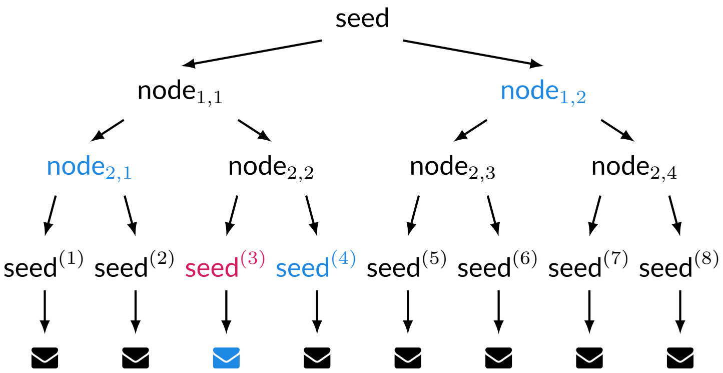
$\mathsf{node}_{1,1}$ $\mathsf{node}_{1,2}$

$\mathsf{node}_{2,1}$ $\mathsf{node}_{2,2}$ $\mathsf{node}_{2,3}$ $\mathsf{node}_{2,4}$

$\mathsf{seed}^{(1)}$ $\mathsf{seed}^{(2)}$ $\mathsf{seed}^{(3)}$ $\mathsf{seed}^{(4)}$ $\mathsf{seed}^{(5)}$ $\mathsf{seed}^{(6)}$ $\mathsf{seed}^{(7)}$ $\mathsf{seed}^{(8)}$

$\mathsf{com}^{(1)}$ $\mathsf{com}^{(2)}$ $\mathsf{com}^{(3)}$ $\mathsf{com}^{(4)}$ $\mathsf{com}^{(5)}$ $\mathsf{com}^{(6)}$ $\mathsf{com}^{(7)}$ $\mathsf{com}^{(8)}$

Vector Commitment



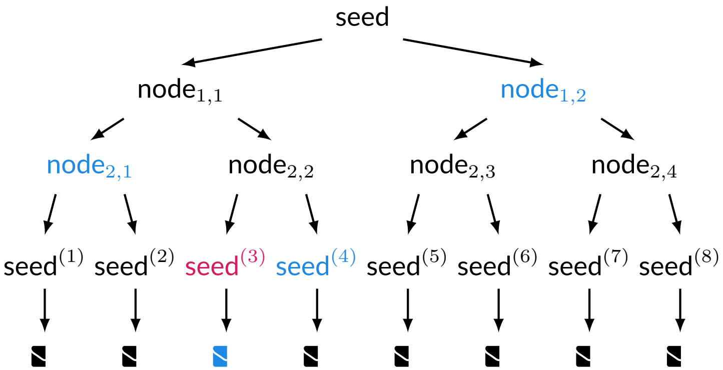
seed

$\text{node}_{1,1}$ $\text{node}_{1,2}$

$\text{node}_{2,1}$ $\text{node}_{2,2}$ $\text{node}_{2,3}$ $\text{node}_{2,4}$

$\text{seed}^{(1)}$ $\text{seed}^{(2)}$ $\text{seed}^{(3)}$ $\text{seed}^{(4)}$ $\text{seed}^{(5)}$ $\text{seed}^{(6)}$ $\text{seed}^{(7)}$ $\text{seed}^{(8)}$

Vector Semi-Commitment

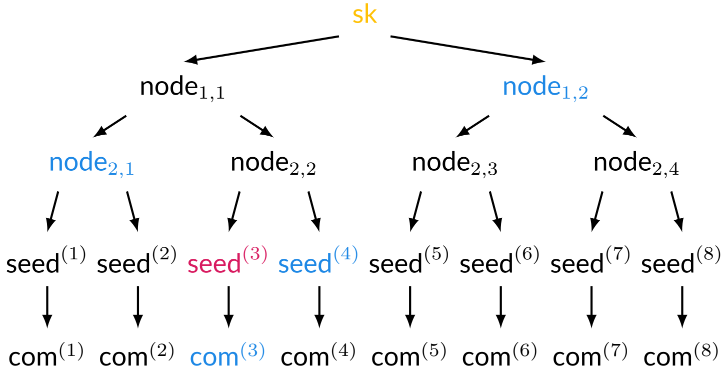


Application of VSC (rMPCitH)

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

Application of VSC (rMPCitH)

1. Halved commitment size
2. GGM tree → correlated GGM tree

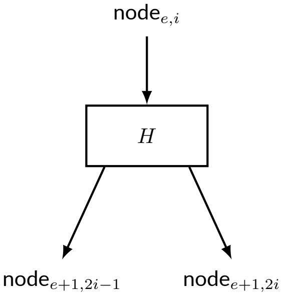


Application of VSC (rMPCitH)

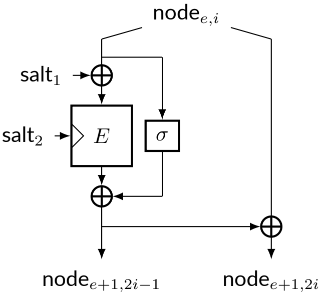
1. Halved commitment size
2. GGM tree $\rightarrow$ correlated GGM tree
3. Random oracle model $\rightarrow$ ideal cipher model

Application of VSC (rMPCitH)

1. Halved commitment size
2. GGM tree → correlated GGM tree
3. Random oracle model → ideal cipher model

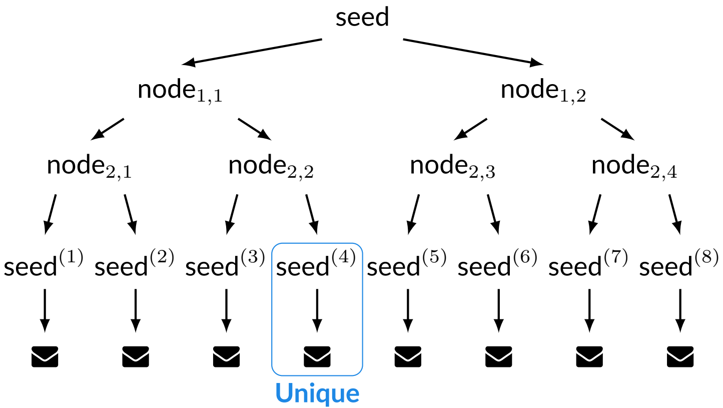


Double-length PRG

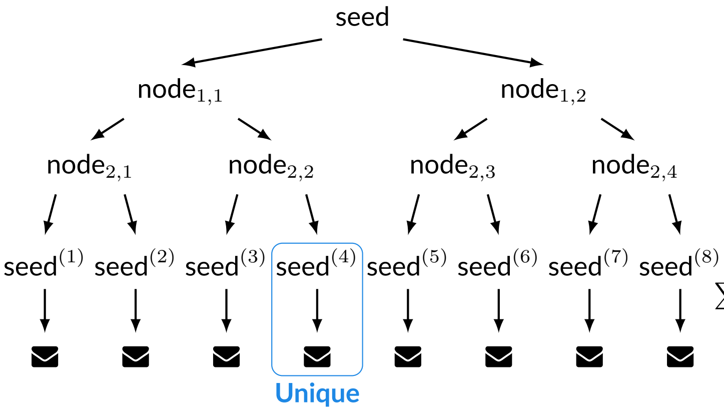


IC-VSC

Difference of Security Proof



Difference of Security Proof



seed

$\text{node}_{1,1}$ $\text{node}_{1,2}$

$\text{node}_{2,1}$ $\text{node}_{2,2}$ $\text{node}_{2,3}$ $\text{node}_{2,4}$

$\text{seed}^{(1)}$ $\text{seed}^{(2)}$ $\text{seed}^{(3)}$ $\text{seed}^{(4)}$ $\text{seed}^{(5)}$ $\text{seed}^{(6)}$ $\text{seed}^{(7)}$ $\text{seed}^{(8)}$

Unique

MultCheck Soundness

$$1/|\mathbb{F}|$$

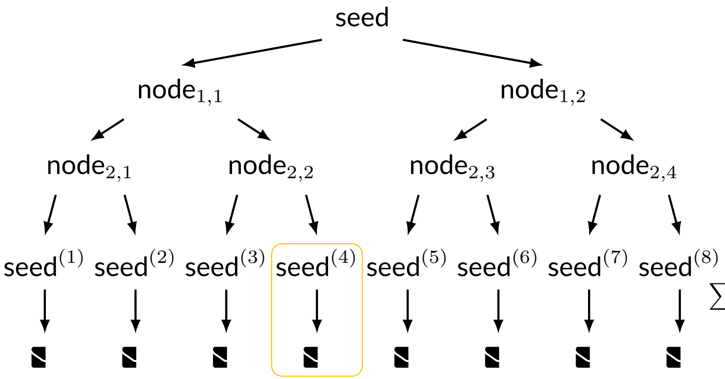
Corrupt Probability

$$1/N$$

KZ Attack Cost

$$\sum_{k=\tau_0}^{\tau} \left(\frac{1}{|\mathbb{F}|}\right)^{k} \left(1-\frac{1}{|\mathbb{F}|}\right)^{\tau-k} + \left(\frac{1}{N}\right)^{\tau-\tau_0}$$

Difference of Security Proof



seed

node$_{1,1}$ node$_{1,2}$

node$_{2,1}$ node$_{2,2}$ node$_{2,3}$ node$_{2,4}$

seed$^{(1)}$ seed$^{(2)}$ seed$^{(3)}$ seed$^{(4)}$ seed$^{(5)}$ seed$^{(6)}$ seed$^{(7)}$ seed$^{(8)}$

Few

$\nu = \lambda / \log \lambda$

MultCheck Soundness

$\nu^N / |\mathbb{F}|$??

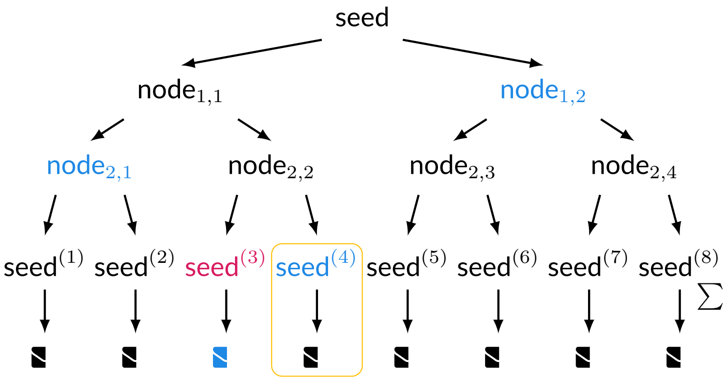
Corrupt Probability

$1/N$

KZ Attack Cost

$$\sum_{k=\tau_0}^{\tau} \left(\frac{\nu^N}{|\mathbb{F}|}\right)^k \left(1 - \frac{\nu^N}{|\mathbb{F}|}\right)^{\tau-k} + \left(\frac{1}{N}\right)^{\tau-\tau_0}$$??

Difference of Security Proof



Few

$\nu = \lambda / \log \lambda$

MultCheck Soundness

$\nu^2 N / 2|\mathbb{F}|$

Corrupt Probability

$1/N$

KZ Attack Cost

$$\sum_{k=\tau_0}^{\tau} \left(\frac{\nu^2 N}{2|\mathbb{F}|}\right)^k \left(1 - \frac{\nu^2 N}{2|\mathbb{F}|}\right)^{\tau-k} + \left(\frac{1}{N}\right)^{\tau-\tau_0}$$

Performance

Scheme	$\lvert pk \rvert$ (B)	$\lvert sig \rvert$ (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

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`.