Efficacy and Mitigation of the Cryptanalysis on AIM

Seongkwang Kim[1]
Jincheol Ha[2]
Mincheol Son[2]
Byeonghak Lee[1]

[1] Samsung SDS, Seoul, Korea
[2] KAIST, Daejeon, Korea

The 5th NIST PQC Standardization Conference

SAMSUNG SDS KAIST

Overview

- Cryptanalysis on AIM
  - AIMer is a NIST PQC round 1 candidate based on MPC-in-the-Head paradigm and symmetric primitive AIM
  - AIM has been analyzed recently up to 15-bit security degradation
  - We re-analyze the complexity of exhaustive search on AIM, and re-calculate the amount of the security degradation

Overview

- Cryptanalysis on AIM
  - AIMer is a NIST PQC round 1 candidate based on MPC-in-the-Head paradigm and symmetric primitive AIM
  - AIM has been analyzed recently up to 15-bit security degradation
  - We re-analyze the complexity of exhaustive search on AIM, and re-calculate the amount of the security degradation

- AIM2 and AIMer v2.0
  - To mitigate the analyses, we propose a new symmetric primitive AIM2 which inherits the design rationale of AIM
  - We extensively analyze the security of AIM2
  - Despite of the patch, AIMer v2.0 enjoys faster performance

Symmetric Primitive AIM

Symmetric Primitive AIM

- $\mathrm{AIM}: \{0,1\}^n \times \mathbb{F}_{2^n} \rightarrow \mathbb{F}_{2^n}$ is the one-way function in AIMer v1.0
- It was designed to be efficiently proved by BN++

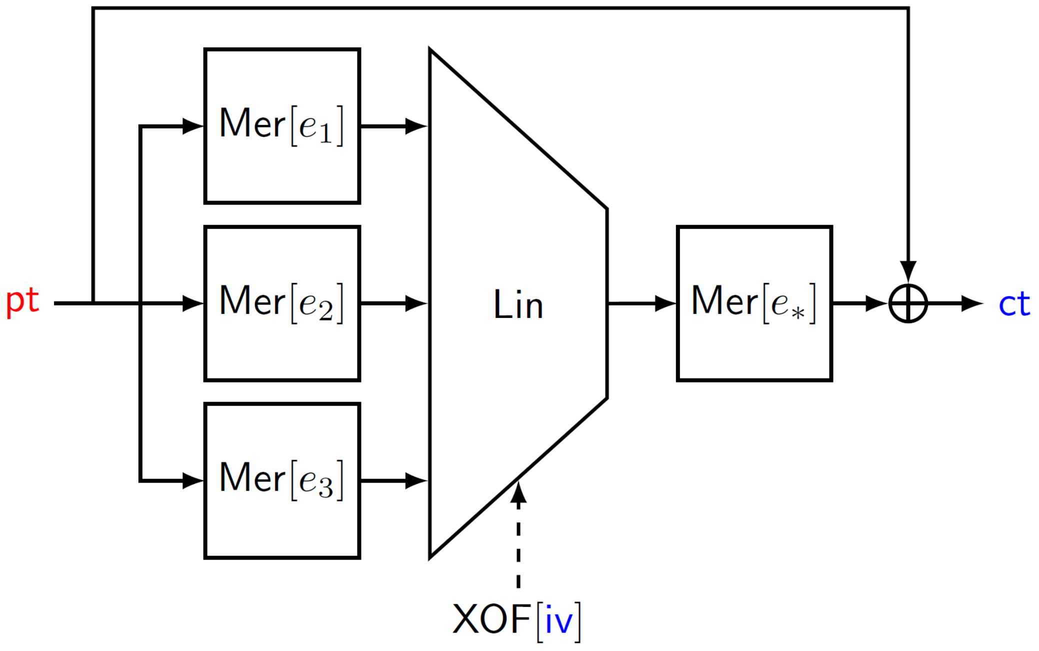
Symmetric Primitive AIM

- $\mathrm{AIM}: \{0,1\}^n \times \mathbb{F}_{2^n} \rightarrow \mathbb{F}_{2^n}$ is the one-way function in AIMer v1.0
- It was designed to be efficiently proved by BN++
- Given a single pair $(\mathrm{iv}, \mathrm{ct})$ such that $\mathrm{iv} \leftarrow_{\$} \{0,1\}^n$ and $\mathrm{AIM}(\mathrm{iv}, \mathrm{pt}) = \mathrm{ct}$, it should be hard to find $\mathrm{pt}^* \in \mathbb{F}_{2^n}$ such that

$$\mathrm{AIM}(\mathrm{iv}, \mathrm{pt}^*) = \mathrm{ct}$$

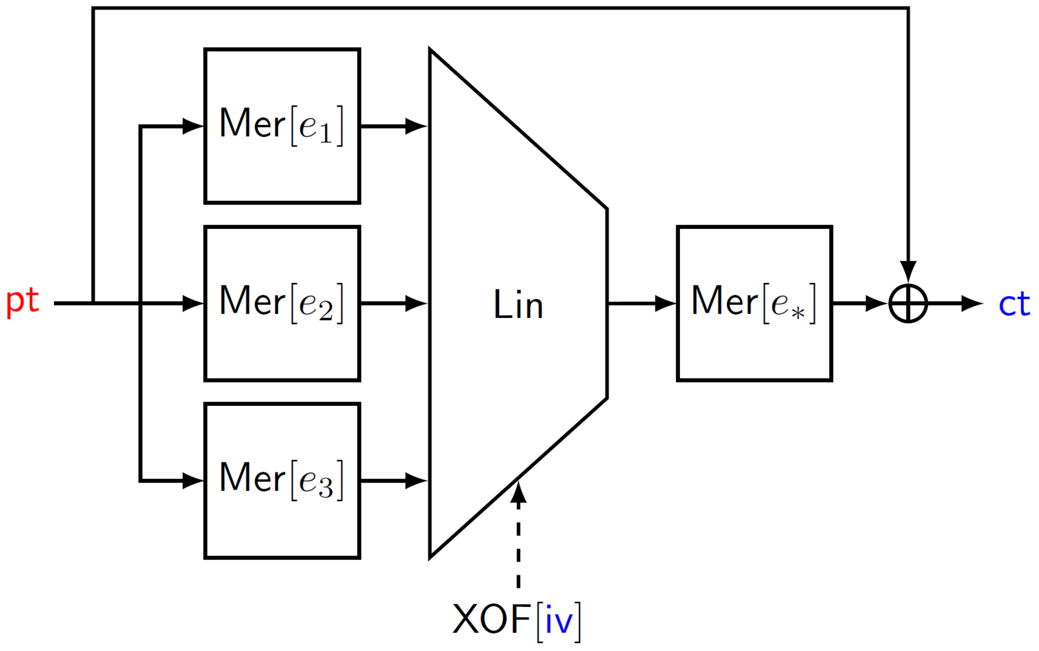
- In AIMer, $pk = (\mathrm{iv}, \mathrm{ct})$ and $sk = (pk, \mathrm{pt})$

Symmetric Primitive AIM



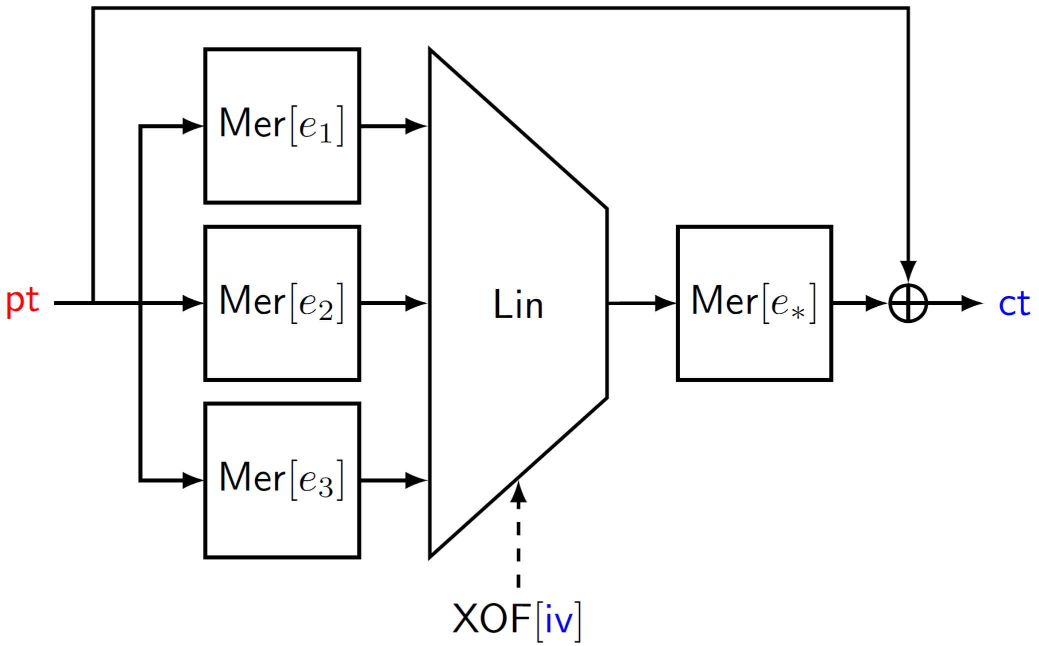
- Mersenne S-box
  - $\mathrm{Mer}[e](x) = x^{2^e - 1}$
  - Invertible, high-degree, quadratic relation
  - Requires a single multiplication
  - Produces $3n$ quadratic equations

Symmetric Primitive AIM



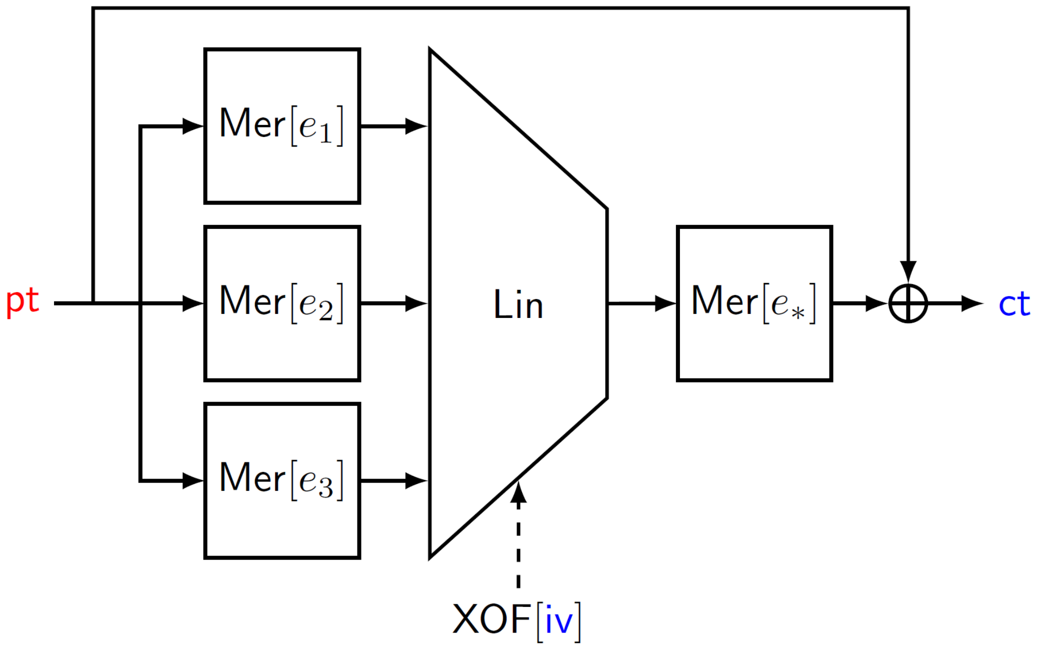
- Mersenne S-box
  - $\mathrm{Mer}[e](x) = x^{2^e-1}$
  - Invertible, high-degree, quadratic relation
  - Requires a single multiplication
  - Produces $3n$ quadratic equations
- Repetitive structure
  - Parallel application of S-boxes
  - Feed-forward construction
  - Fully exploit the BN++ optimizations

Symmetric Primitive AIM



- Mersenne S-box
  - $\mathrm{Mer}[e](x) = x^{2^e-1}$
  - Invertible, high-degree, quadratic relation
  - Requires a single multiplication
  - Produces $3n$ quadratic equations
- Repetitive structure
  - Parallel application of S-boxes
  - Feed-forward construction
  - Fully exploit the BN++ optimizations
- Randomized structure
  - $(A_{\mathrm{iv}}, b_{\mathrm{iv}}) \leftarrow \mathrm{XOF}(\mathrm{iv})$
  - $\mathrm{Lin}(x) = A_{\mathrm{iv}} \cdot x + b_{\mathrm{iv}}$

Symmetric Primitive AIM



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

- Mersenne S-box
  - $\mathrm{Mer}[e](x) = x^{2^e-1}$
  - Invertible, high-degree, quadratic relation
  - Requires a single multiplication
  - Produces $3n$ quadratic equations
- Repetitive structure
  - Parallel application of S-boxes
  - Feed-forward construction
  - Fully exploit the BN++ optimizations
- Randomized structure
  - $(A_{\mathrm{iv}}, b_{\mathrm{iv}}) \leftarrow \mathrm{XOF}(\mathrm{iv})$
  - $\mathrm{Lin}(x) = A_{\mathrm{iv}} \cdot x + b_{\mathrm{iv}}$

Analyses on AIM

Exhaustive Search on AIM

- In the conference version, the complexity of exhaustive search on AIM was overestimated
- The reason is the addition chain structure of AIM
- For example, AIM-I requires only 6 multiplications for evaluating 2 S-boxes

$$x \to x^{2^2-1} \to x^{2^3-1} \to x^{2^6-1} \to x^{2^{12}-1} \to x^{2^{24}-1} \to x^{2^{27}-1}$$

Exhaustive Search on AIM

- In the conference version, the complexity of exhaustive search on AIM was overestimated
- The reason is the addition chain structure of AIM
- For example, AIM-I requires only 6 multiplications for evaluating 2 S-boxes

$$x \to x^{2^2-1} \to x^{2^3-1} \to x^{2^6-1} \to x^{2^{12}-1} \to x^{2^{24}-1} \to x^{2^{27}-1}$$

	Previous Cost	**Current Cost**	AES Cost
AIM-I	149.0	146.3	143
AIM-III	214.4	211.8	207
AIM-V	280.0	276.7	272

Table. Complexity of exhaustive search attack on AIM and AES in log

Recent Analyses on AIM

- Recent analysis on AIM
  - [LMOM23] Fast exhaustive search, claiming up to 15-bit security degradation
  - [Liu23] Less costly algebraic attack, but not broken
  - [Sar23] Efficient exhaustive search by implementation, unknown amount of security degradation
  - [ZWYGC23] Guess & determine + linearization attack, claiming up to 6-bit security degradation

[LMOM23] F. Liu, M. Mahzoun, M. Øygarden, and W. Meier. *Algebraic Attacks on RAIN and AIM Using Equivalent Representations*. IACR Transactions on Symmetric Cryptology 2023(4): 166-186.
[Liu23] F. Liu. *Mind Multiple Power Maps: Algebraic Cryptanalysis of Full AIM for Post-quantum Signature Scheme AIMer*. In private communication. 2023.
[Sar23] M. O. Saarinen. *Round 1 (Additional Signatures) OFFICIAL COMMENT: AIMer*. https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/BI2ilXblNy0.
[ZWYGC23] K. Zhang, Q. Wang, Y. Yu, C. Guo, and H. Cui. *Algebraic Attacks on Round-Reduced RAIN and Full AIM-III*. Asiacrypt 2023.

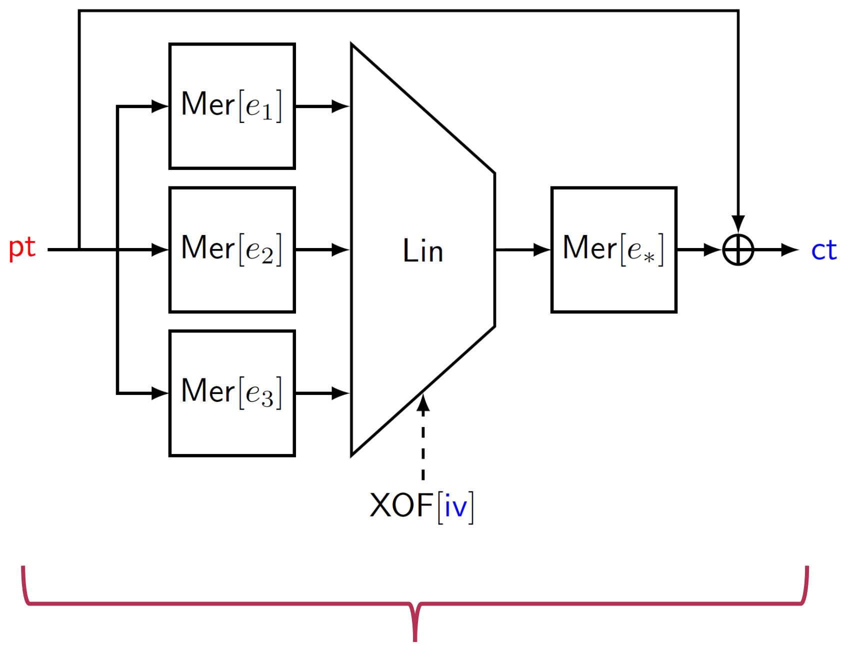
Recent Analyses on AIM

- Recent analysis on AIM
  - [LMOM23] Fast exhaustive search, claiming up to 15-bit security degradation
  - [Liu23] Less costly algebraic attack, but not broken
  - [Sar23] Efficient exhaustive search by implementation, unknown amount of security degradation
  - [ZWYGC23] Guess & determine + linearization attack, claiming up to 6-bit security degradation

- Mainly, there are two vulnerabilities in the structure of AIM
  - Low degree representation in n variables ⇒ Fast exhaustive search attack
  - Common input to the parallel Mersenne S-boxes ⇒ Structural vulnerability

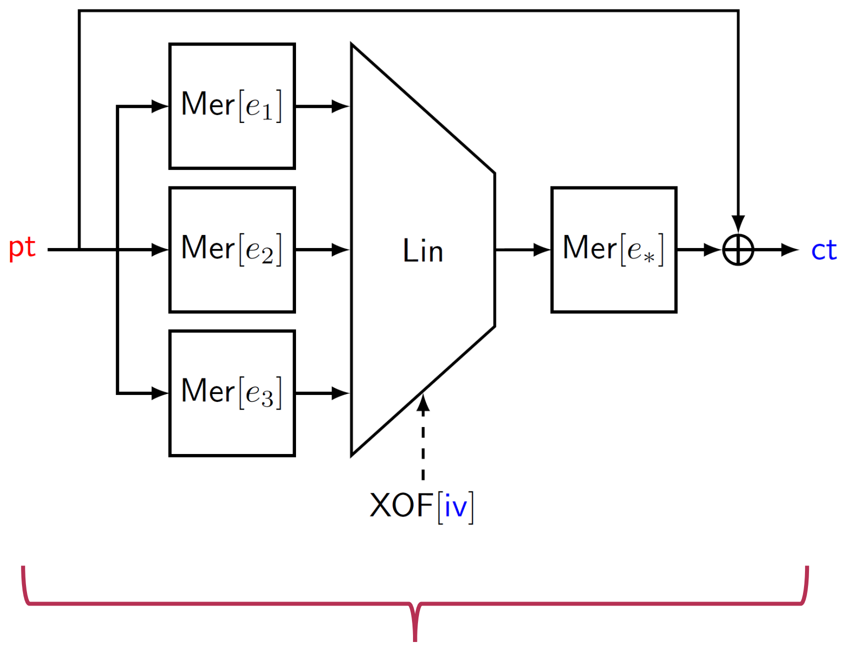
[LMOM23] F. Liu, M. Mahzoun, M. Øygarden, and W. Meier. *Algebraic Attacks on RAIN and AIM Using Equivalent Representations*. IACR Transactions on Symmetric Cryptology 2023(4): 166-186.
[Liu23] F. Liu. *Mind Multiple Power Maps: Algebraic Cryptanalysis of Full AIM for Post-quantum Signature Scheme AIMer*. In private communication. 2023.
[Sar23] M. O. Saarinen. *Round 1 (Additional Signatures) OFFICIAL COMMENT: AIMer*. https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/BI2ilXblNy0.
[ZWYGC23] K. Zhang, Q. Wang, Y. Yu, C. Guo, and H. Cui. *Algebraic Attacks on Round-Reduced RAIN and Full AIM-III*. Asiacrypt 2023.

Fast Exhaustive Search Attack (Liu et al.)



$$\mathrm{AIM[iv](pt)} = \mathrm{ct}$$
$$\Leftrightarrow F(x) = y \ \&\ \deg F = d$$

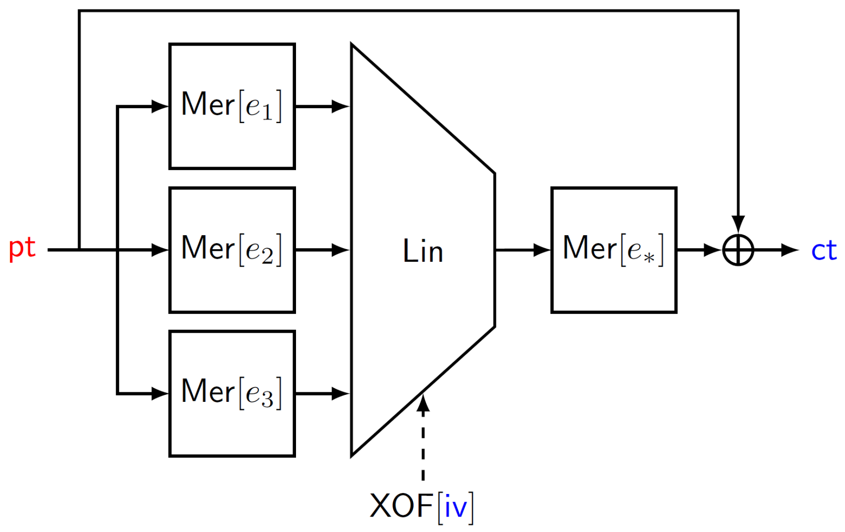
Fast Exhaustive Search Attack (Liu et al.)



$$\mathrm{AIM[iv](pt)} = \mathrm{ct}$$
$$\Leftrightarrow F(x) = y \ \&\ \deg F = d$$

- Boolean polynomial system can be brute-force searched with $4d \log n \, 2^n$ computation and $O(n^{d+2})$ memory if d is small enough

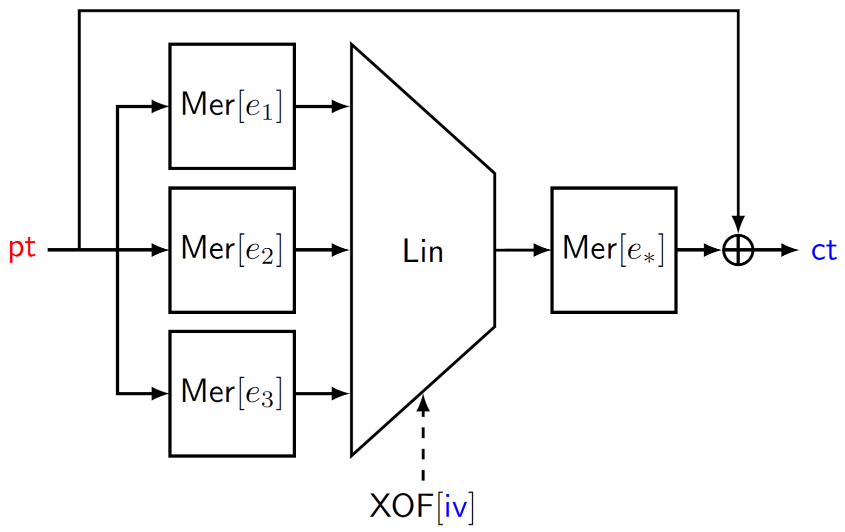
Fast Exhaustive Search Attack (Liu et al.)



$$\mathrm{AIM}[\mathrm{iv}](\mathrm{pt}) = \mathrm{ct}$$
$$\Leftrightarrow F(x) = y \;\&\; \deg F = d$$

- Boolean polynomial system can be brute-force searched with $4d \log n \, 2^n$ computation and $O(n^{d+2})$ memory if d is small enough
- If degree d is small enough, this fast exhaustive search is faster than naive brute-force search

Fast Exhaustive Search Attack (Liu et al.)

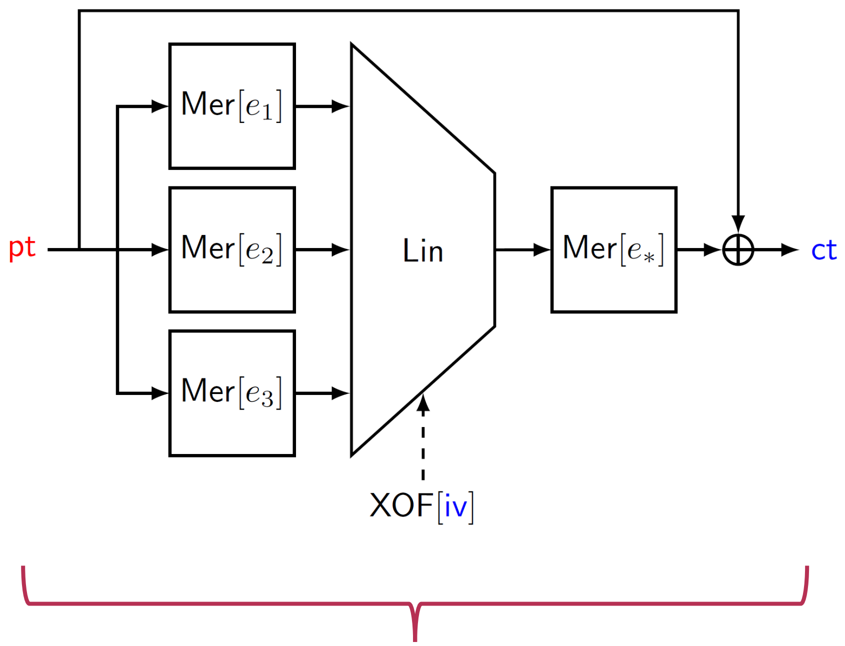


$$\mathrm{AIM[iv](pt) = ct}$$
$$\Leftrightarrow F(x) = y \ \& \ \deg F = d$$

- Boolean polynomial system can be brute-force searched with $4d \log n \, 2^n$ computation and $O(n^{d+2})$ memory if d is small enough
- If degree d is small enough, this fast exhaustive search is faster than naive brute-force search
- The result of Liu et al. (updated security degradation)

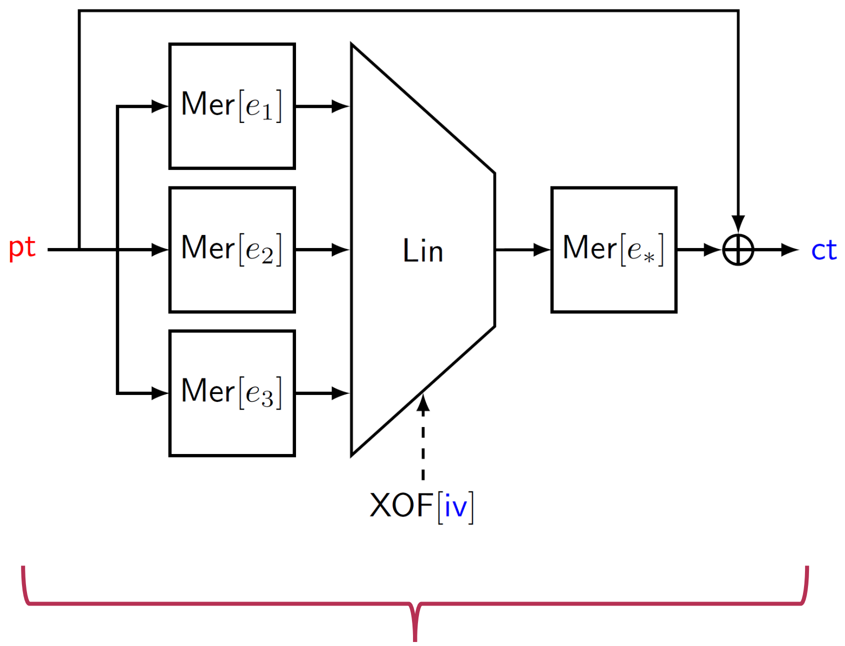
	n	Deg	Log(Time) [bits]	Log(Mem) [bits]
AIM-I	128	10	136.2 (−10.1)	61.7
AIM-III	192	14	200.7 (−11.1)	84.3
AIM-V	256	15	265.0 (−11.7)	95.1

Easier System to Solve (Liu)



$$w = \text{pt}^{-1}$$
$$\text{Mer}[e_i](\text{pt}) = w \cdot \text{pt}^{2^{e_i}}$$

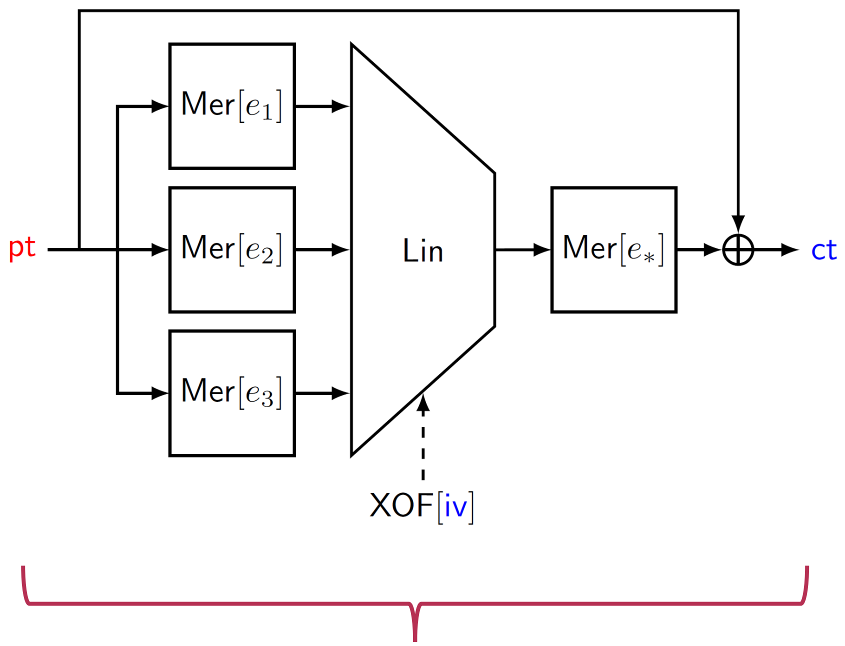
Easier System to Solve (Liu)



$$w = \mathrm{pt}^{-1}$$
$$\mathrm{Mer}[e_i](\mathrm{pt}) = w \cdot \mathrm{pt}^{2^{e_i}}$$

- Introducing w makes a system of $5n$ quadratic, $5n$ cubic equations in $2n$ variables

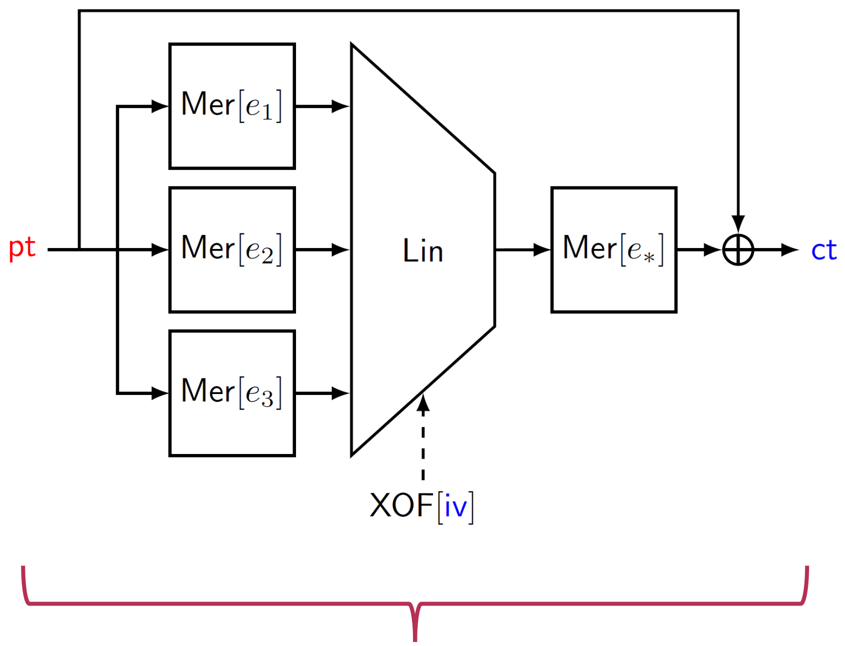
Easier System to Solve (Liu)



$$w = \text{pt}^{-1}$$
$$\text{Mer}[e_i](\text{pt}) = w \cdot \text{pt}^{2^{e_i}}$$

- Introducing w makes a system of $5n$ quadratic, $5n$ cubic equations in $2n$ variables
- If XL algorithm always generate linearly independent equations, then this attack works

Easier System to Solve (Liu)



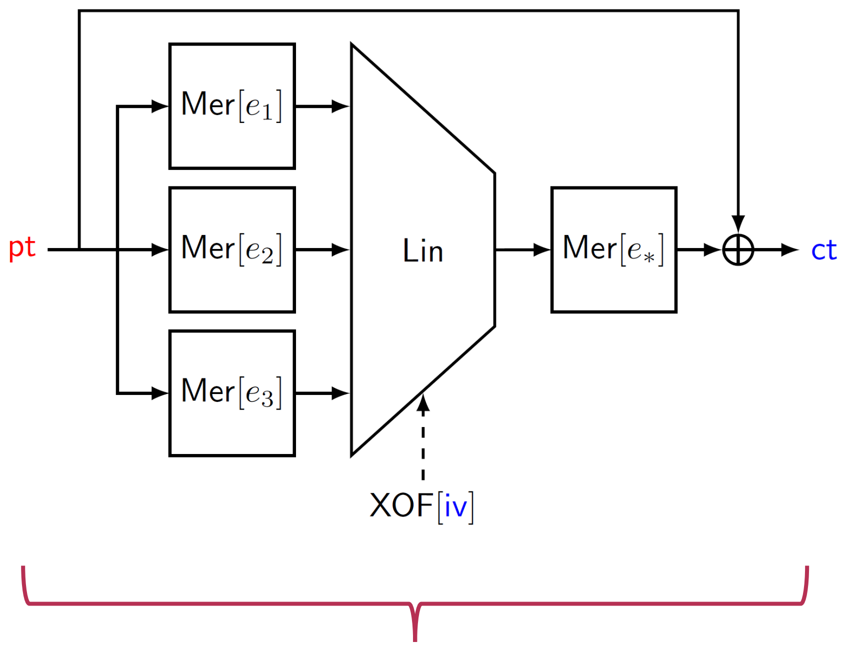
$$w = \mathrm{pt}^{-1}$$
$$\mathrm{Mer}[e_i](\mathrm{pt}) = w \cdot \mathrm{pt}^{2^{e_i}}$$

- Introducing w makes a system of $5n$ quadratic, $5n$ cubic equations in $2n$ variables
- If XL algorithm always generate linearly independent equations, then this attack works
- The result of Liu (our estimation)

	n	Log(Time*) [bits]	Log(Time**) [bits]
AIM-I	128	124.8 (−18.8)	158.3 (+14.4)
AIM-III	192	157.5 (−54.3)	226.5 (+14.7)
AIM-V	256	188.9 (−87.8)	290.2 (+13.5)

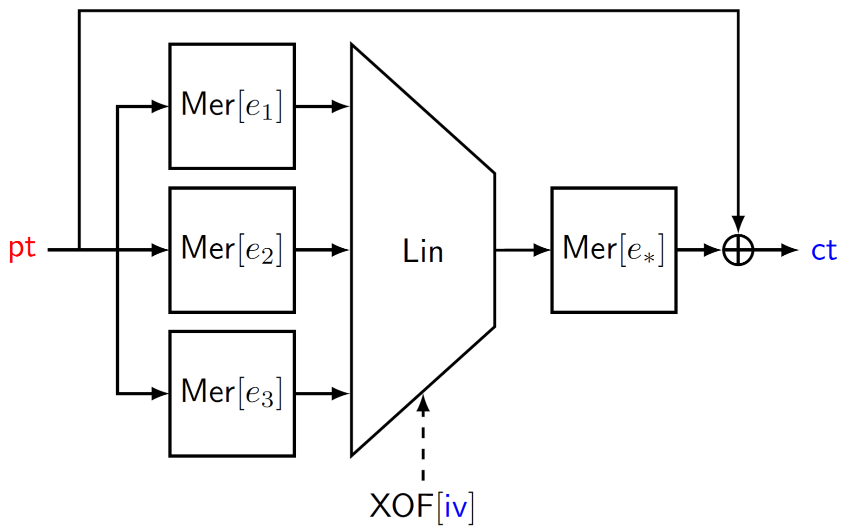
*Assumption: Every equations generated by XL are linearly independent (unrealistic)
**Assumption: XL finishes at the degree of regularity

Efficient Exhaustive Search (Saarinen)



$$w = \mathrm{pt}^{-1}$$
$$\mathrm{Mer}[e_i](\mathrm{pt}) = w \cdot \mathrm{pt}^{2^{e_i}}$$

Efficient Exhaustive Search (Saarinen)

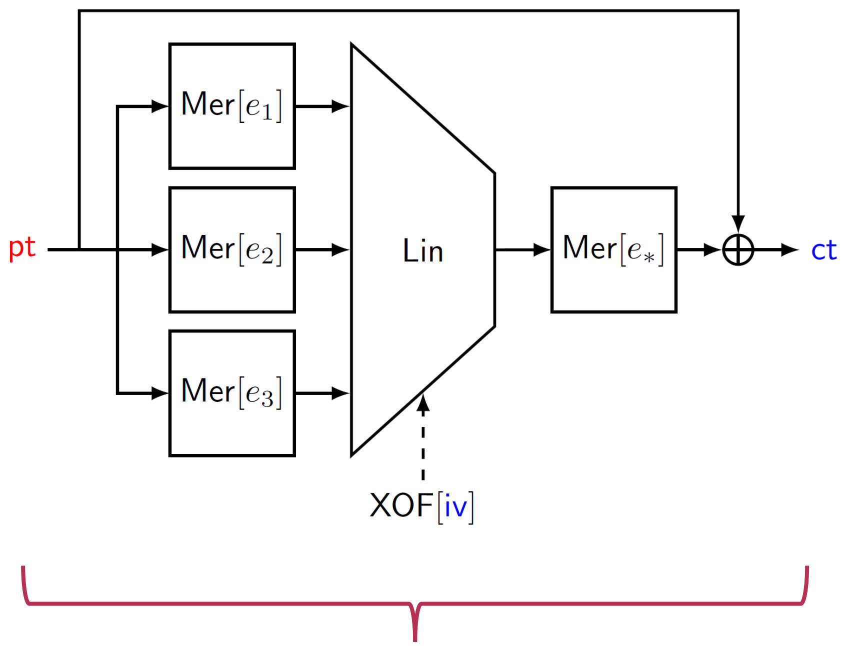


$$w = \mathrm{pt}^{-1}$$
$$\mathrm{Mer}[e_i](\mathrm{pt}) = w \cdot \mathrm{pt}^{2^{e_i}}$$

- Using LFSR for $\mathbb{F}_{2^n}$, exhaustive search on x^{-1} is easy:

$$x \ll_{\mathrm{LFSR}} 1 = x \cdot \alpha \ \text{ in } \mathbb{F}_2[\alpha]/(f(\alpha))$$
$$(x \ll_{\mathrm{LFSR}} 1)^{-1} = x^{-1} \gg_{\mathrm{LFSR}} 1$$

Efficient Exhaustive Search (Saarinen)



$w = \mathrm{pt}^{-1}$

$\mathrm{Mer}[e_i](\mathrm{pt}) = w \cdot \mathrm{pt}^{2^{e_i}}$

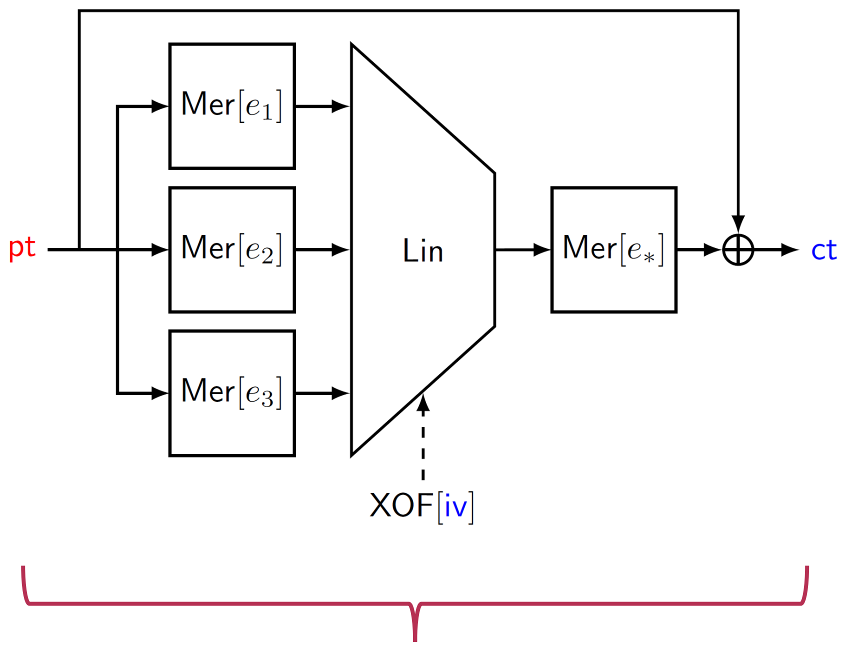
- Using LFSR for $\mathbb{F}_{2^n}$, exhaustive search on x^{-1} is easy:

$$x \ll_{\mathrm{LFSR}} 1 = x \cdot \alpha \ \text{ in } \mathbb{F}_2[\alpha]/(f(\alpha))$$

$$(x \ll_{\mathrm{LFSR}} 1)^{-1} = x^{-1} \gg_{\mathrm{LFSR}} 1$$

- The common inverse w reduces the number of multiplications → Low complexity

Efficient Exhaustive Search (Saarinen)



$w = \mathrm{pt}^{-1}$

$\mathrm{Mer}[e_i](\mathrm{pt}) = w \cdot \mathrm{pt}^{2^{e_i}}$

- Using LFSR for $\mathbb{F}_{2^n}$, exhaustive search on x^{-1} is easy:

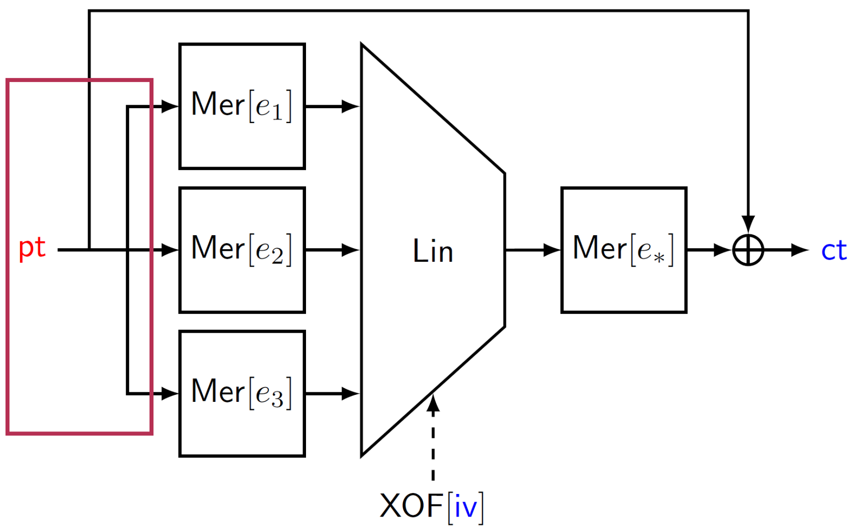
$$x \ll_{\mathrm{LFSR}} 1 = x \cdot \alpha \ \text{ in } \mathbb{F}_2[\alpha]/(f(\alpha))$$

$$(x \ll_{\mathrm{LFSR}} 1)^{-1} = x^{-1} \gg_{\mathrm{LFSR}} 1$$

- The common inverse w reduces the number of multiplications → Low complexity
- The result of Saarinen (new estimation)

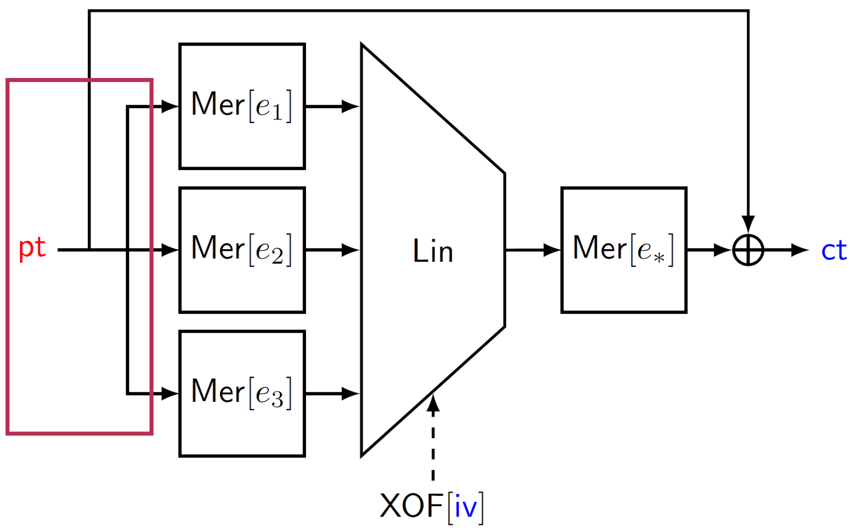
	n	#mult	Log(Time) [bits]
AIM-I	128	3	145.0 (−1.3)
AIM-III	192	3	210.2 (−1.6)
AIM-V	256	4	275.5 (−1.2)

Structural Vulnerability (Zhang et al.)



Inputs to parallel S-boxes are all the same

Structural Vulnerability (Zhang et al.)

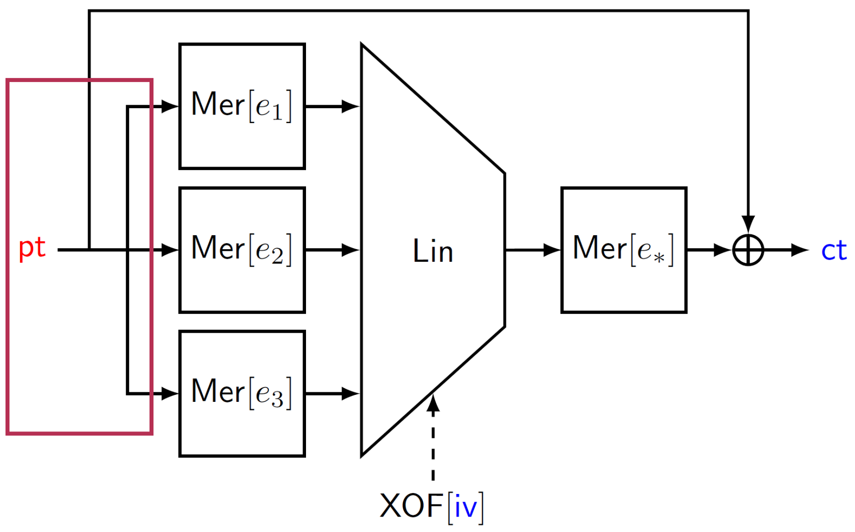


Inputs to parallel S-boxes are all the same

- Find some $d|(2^n - 1)$ such that

$$\begin{cases} \mathrm{Mer}[e_1](\mathrm{pt}) = \left(\mathrm{pt}^d\right)^{s_1} \cdot \mathrm{pt}^{2^{t_1}} \\ \mathrm{Mer}[e_2](\mathrm{pt}) = \left(\mathrm{pt}^d\right)^{s_2} \cdot \mathrm{pt}^{2^{t_2}} \\ \mathrm{Mer}[e_3](\mathrm{pt}) = \left(\mathrm{pt}^d\right)^{s_3} \cdot \mathrm{pt}^{2^{t_3}} \end{cases}$$

Structural Vulnerability (Zhang et al.)



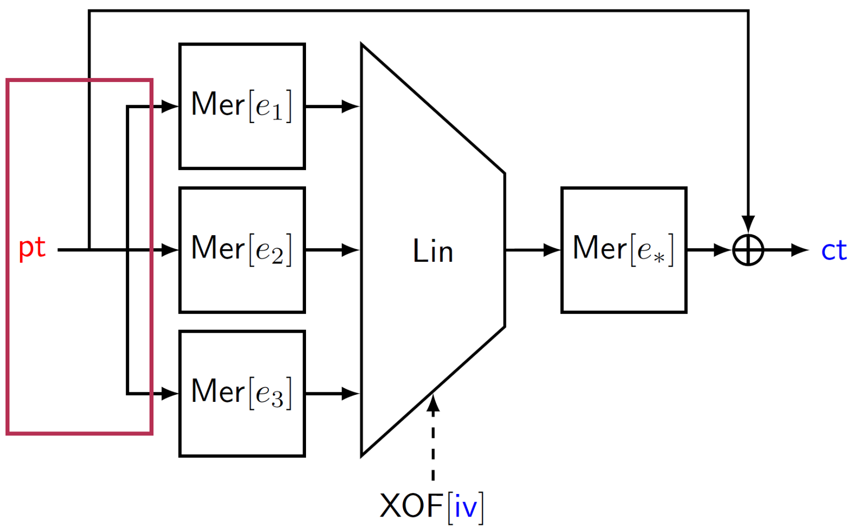
Inputs to parallel S-boxes are all the same

- Find some $d|(2^n - 1)$ such that

$$\begin{cases} \mathrm{Mer}[e_1](\mathrm{pt}) = \left(\mathrm{pt}^d\right)^{s_1} \cdot \mathrm{pt}^{2^{t_1}} \\ \mathrm{Mer}[e_2](\mathrm{pt}) = \left(\mathrm{pt}^d\right)^{s_2} \cdot \mathrm{pt}^{2^{t_2}} \\ \mathrm{Mer}[e_3](\mathrm{pt}) = \left(\mathrm{pt}^d\right)^{s_3} \cdot \mathrm{pt}^{2^{t_3}} \end{cases}$$

- When pt^d is guessed, above system becomes linear

Structural Vulnerability (Zhang et al.)



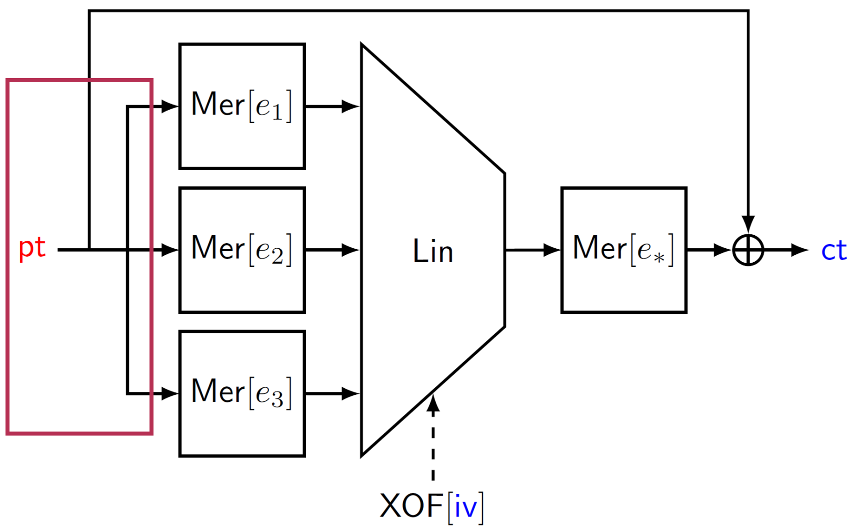
Inputs to parallel S-boxes are all the same

- Find some $d|(2^n - 1)$ such that

$$\begin{cases} \mathrm{Mer}[e_1](\mathrm{pt}) = \left(\mathrm{pt}^d\right)^{s_1} \cdot \mathrm{pt}^{2^{t_1}} \\ \mathrm{Mer}[e_2](\mathrm{pt}) = \left(\mathrm{pt}^d\right)^{s_2} \cdot \mathrm{pt}^{2^{t_2}} \\ \mathrm{Mer}[e_3](\mathrm{pt}) = \left(\mathrm{pt}^d\right)^{s_3} \cdot \mathrm{pt}^{2^{t_3}} \end{cases}$$

- When pt^d is guessed, above system becomes linear
- A few bits of complexity are dismissed as a constant in the big-O notation

Structural Vulnerability (Zhang et al.)



Inputs to parallel S-boxes are all the same

- Find some $d|(2^n - 1)$ such that

$$\begin{cases} \mathrm{Mer}[e_1](\mathrm{pt}) = \left(\mathrm{pt}^d\right)^{s_1} \cdot \mathrm{pt}^{2^{t_1}} \\ \mathrm{Mer}[e_2](\mathrm{pt}) = \left(\mathrm{pt}^d\right)^{s_2} \cdot \mathrm{pt}^{2^{t_2}} \\ \mathrm{Mer}[e_3](\mathrm{pt}) = \left(\mathrm{pt}^d\right)^{s_3} \cdot \mathrm{pt}^{2^{t_3}} \end{cases}$$

- When pt^d is guessed, above system becomes linear
- A few bits of complexity are dismissed as a constant in the big-O notation
- The result of Zhang et al. (our estimation)

	n	d	Log(Time) [bits]
AIM-I	128	5	146.0 (−0.3)
AIM-III	192	45	210.4 (−1.4)
AIM-V	256	3	277.0 (+0.3)

Summary of Analyses on AIM

- The main vulnerabilities of AIM are:
  - Low algebraic degree
  - No domain separation
- By our complexity estimations, the amount of security degradation is clarified or reduced
- Some turn out to be not as powerful as claimed

	FES (Liu et al.)	Easier System (Liu)	Efficient Search (Saarinen)	Linearization (Zhang et al.)	**Exhaustive Search**	AES Cost
AIM-I	136.2 (−10.1)	158.3 (+14.4)	145.0 (−1.3)	146.0 (−0.3)	146.3	143
AIM-III	200.7 (−11.1)	226.5 (+14.7)	210.2 (−1.6)	210.4 (−1.4)	211.8	207
AIM-V	265.0 (−11.7)	290.2 (+13.5)	275.5 (−1.2)	277.0 (+0.3)	276.7	272

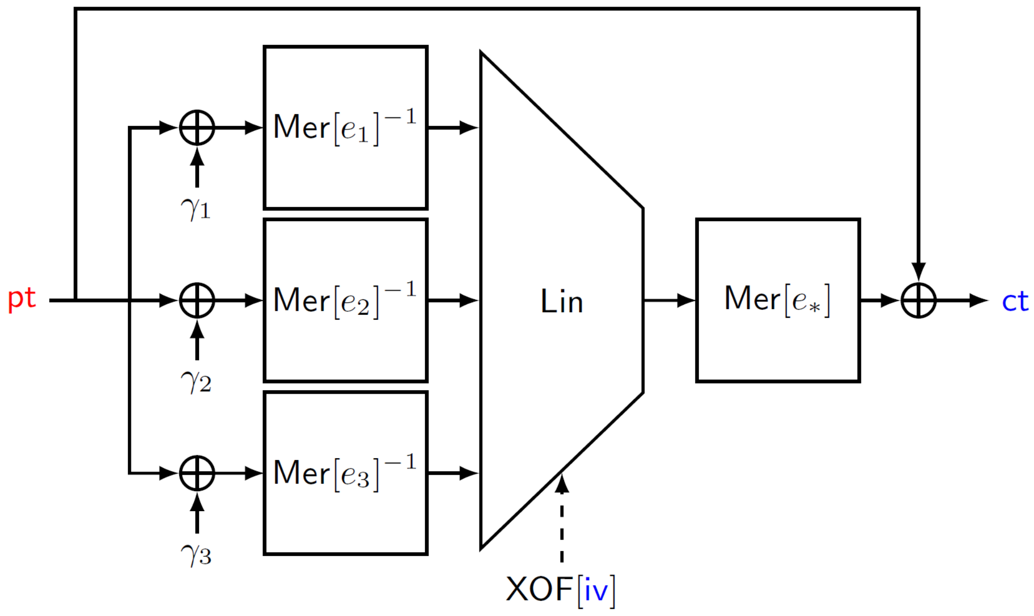
Summary of Analyses on AIM

- The main vulnerabilities of AIM are:
  - Low algebraic degree
  - No domain separation
- By our complexity estimations, the amount of security degradation is clarified or reduced
- Some turn out to be not as powerful as claimed

	FES (Liu et al.)	Easier System (Liu)	Efficient Search (Saarinen)	Linearization (Zhang et al.)	Exhaustive Search	**AES Cost**
AIM-I	136.2 (−6.8)	158.3 (+17.7)	145.0 (+2.0)	146.0 (+3.0)	146.3	143
AIM-III	200.7 (−5.3)	226.5 (+19.5)	210.2 (+3.2)	210.4 (+3.4)	211.8	207
AIM-V	265.0 (−7.0)	290.2 (+18.2)	275.5 (+3.5)	277.0 (+5.0)	276.7	272

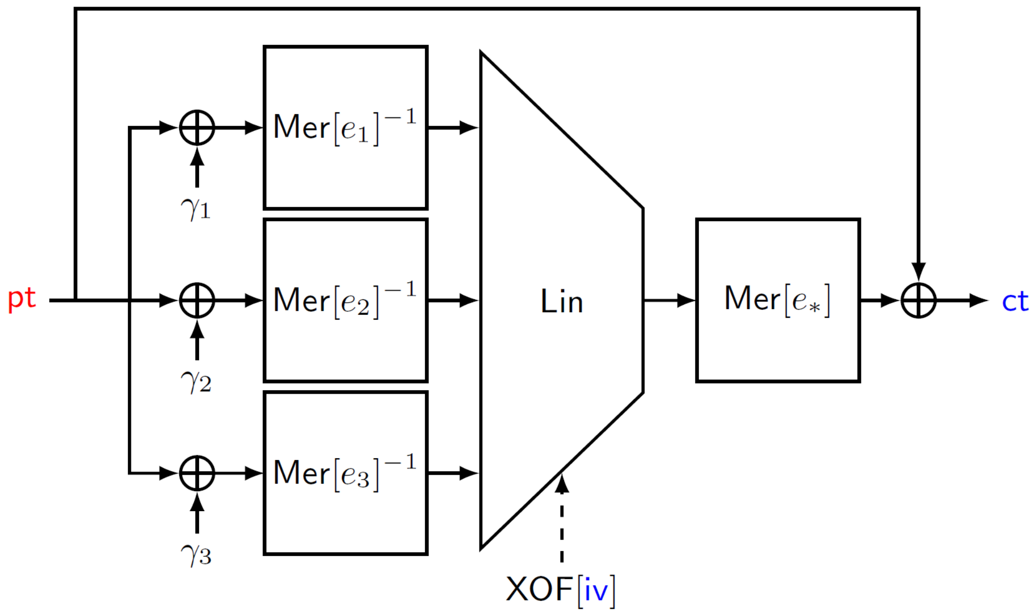
AIM2 and Analysis

AIM2: Secure Patch for Algebraic Attacks



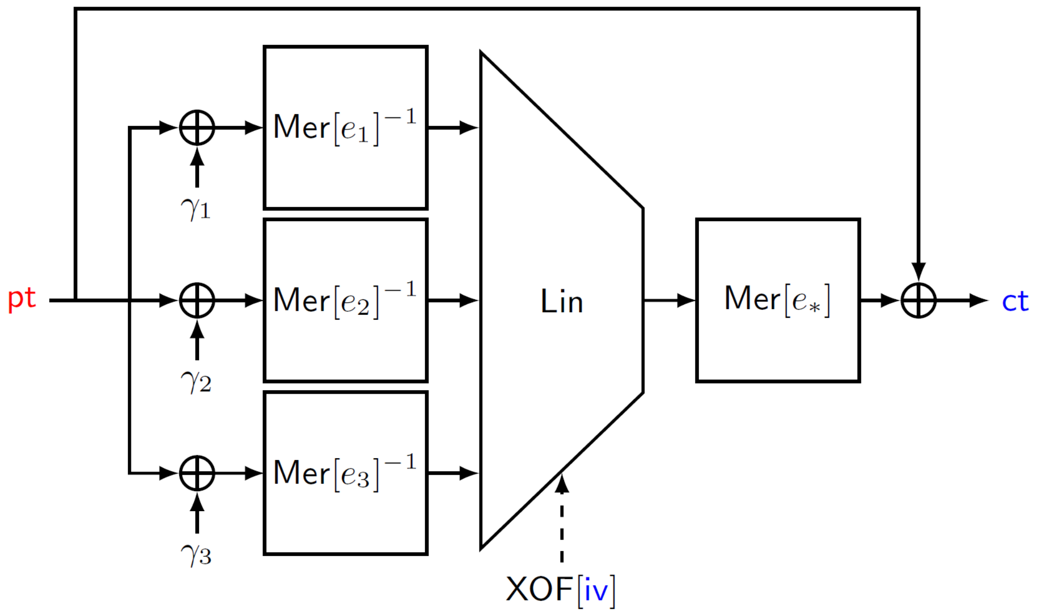
- Inverse Mersenne S-box
  - $\mathrm{Mer}[e]^{-1}(x) = x^a$
  - $a = (2^e - 1)^{-1} \bmod (2^n - 1)$
  - More resistant to algebraic attacks

AIM2: Secure Patch for Algebraic Attacks



- Inverse Mersenne S-box
  - $\mathrm{Mer}[e]^{-1}(x) = x^a$
  - $a = (2^e - 1)^{-1} \bmod (2^n - 1)$
  - More resistant to algebraic attacks
- Larger exponents
  - To mitigate fast exhaustive search

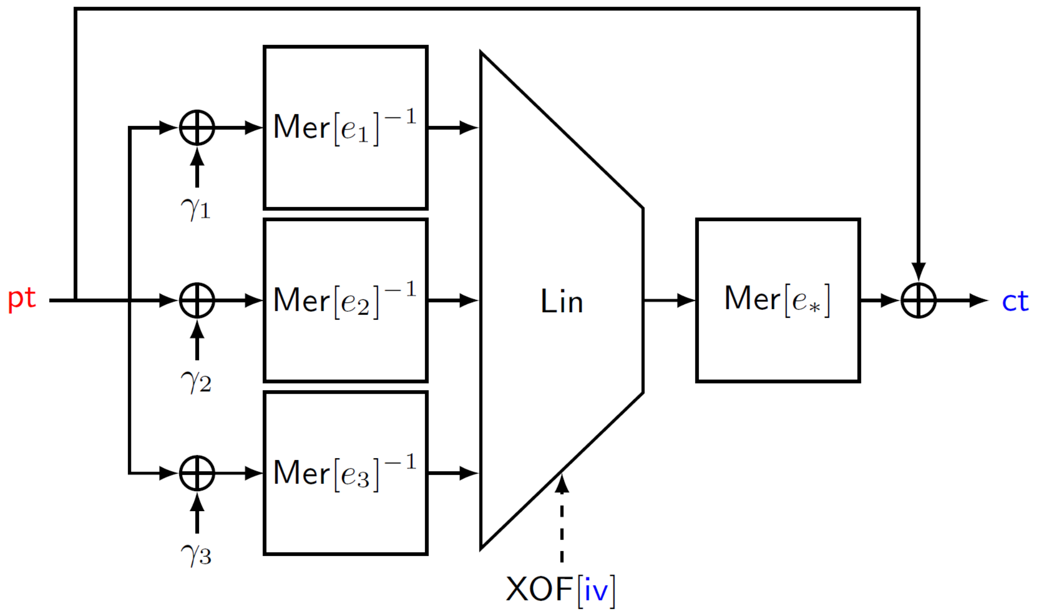
AIM2: Secure Patch for Algebraic Attacks



- Inverse Mersenne S-box
  - $\mathrm{Mer}[e]^{-1}(x) = x^a$
  - $a = (2^e - 1)^{-1} \bmod (2^n - 1)$
  - More resistant to algebraic attacks
- Larger exponents
  - To mitigate fast exhaustive search
- Fixed constant addition
  - To differentiate inputs of S-boxes
  - Increase the degree of composite power function

$$(x^a)^b \text{ vs } (x^a + c)^b$$

AIM2: Secure Patch for Algebraic Attacks



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

- Inverse Mersenne S-box
  - $\mathrm{Mer}[e]^{-1}(x) = x^a$
  - $a = (2^e - 1)^{-1} \bmod (2^n - 1)$
  - More resistant to algebraic attacks
- Larger exponents
  - To mitigate fast exhaustive search
- Fixed constant addition
  - To differentiate inputs of S-boxes
  - Increase the degree of composite power function

$$(x^a)^b \text{ vs } (x^a + c)^b$$

Algebraic Analysis on AIM2

- Brute-force search of quadratic equations
  - Variables: x (input), t_i (output of i-th S-box), z (input of the last S-box) in $\mathbb{F}_2^n$

Algebraic Analysis on AIM2

- Brute-force search of quadratic equations
  - Variables: x (input), t_i (output of i-th S-box), z (input of the last S-box) in $\mathbb{F}_2^n$
  - Set up an equation with indeterminate $a_{\alpha\beta\gamma}$:

$$\sum_{\substack{\alpha,\gamma\in\mathbb{F}_2^n,\beta=(\beta_1,\ldots,\beta_\ell)\in\mathbb{F}_2^{\ell n} \\ hw(\alpha)+hw(\beta)+hw(\gamma)\le 2}} a_{\alpha\beta\gamma} x^\alpha t_i^{\beta_i} z^\gamma = 0$$

Algebraic Analysis on AIM2

- Brute-force search of quadratic equations
  - Variables: x (input), t_i (output of i-th S-box), z (input of the last S-box) in $\mathbb{F}_2^n$
  - Set up an equation with indeterminate $a_{\alpha\beta\gamma}$:

$$\sum_{\substack{\alpha,\gamma\in\mathbb{F}_2^n,\beta=(\beta_1,\ldots,\beta_\ell)\in\mathbb{F}_2^{\ell n}\\ hw(\alpha)+hw(\beta)+hw(\gamma)\leq 2}} a_{\alpha\beta\gamma}x^{\alpha}t_i^{\beta_i}z^{\gamma}=0$$

  - Randomly sample x, compute corresponding t_i and z, and substitute them

Algebraic Analysis on AIM2

- Brute-force search of quadratic equations
  - Variables: x (input), t_i (output of i-th S-box), z (input of the last S-box) in $\mathbb{F}_2^n$
  - Set up an equation with indeterminate $a_{\alpha\beta\gamma}$:

$$\sum_{\substack{\alpha,\gamma\in\mathbb{F}_2^n,\beta=(\beta_1,\ldots,\beta_\ell)\in\mathbb{F}_2^{\ell n} \\ hw(\alpha)+hw(\beta)+hw(\gamma)\leq 2}} a_{\alpha\beta\gamma}x^\alpha t_i^{\beta_i} z^\gamma = 0$$

  - Randomly sample x, compute corresponding t_i and z, and substitute them
  - Repeat the previous step sufficiently many times, and solve the linear system w.r.t. $a_{\alpha\beta\gamma}$

Algebraic Analysis on AIM2

- Brute-force search of quadratic equations
  - Variables: x (input), t_i (output of i-th S-box), z (input of the last S-box) in $\mathbb{F}_2^n$
  - Set up an equation with indeterminate $a_{\alpha\beta\gamma}$:

$$\sum_{\substack{\alpha,\gamma\in\mathbb{F}_2^n,\beta=(\beta_1,\ldots,\beta_\ell)\in\mathbb{F}_2^{\ell n}\\ hw(\alpha)+hw(\beta)+hw(\gamma)\leq 2}} a_{\alpha\beta\gamma}x^\alpha t_i^{\beta_i}z^\gamma = 0$$

  - Randomly sample x, compute corresponding t_i and z, and substitute them
  - Repeat the previous step sufficiently many times, and solve the linear system w.r.t. $a_{\alpha\beta\gamma}$
- The resulting system and complexity

	#var	#eq	Log(Time) [bits]
AIM2-I	256	384	207.9 (+60.9)
	384	1536	185.3 (+38.3)
AIM2-III	384	576	301.9 (+89.6)
	576	2304	262.4 (+50.1)
AIM2-V	768	1536	503.7 (+226.0)
	1024	4608	411.4 (+133.7)

Algebraic Analysis on AIM2

- Brute-force search of intermediate variables in a S-box
  - Variable: $x \in \mathbb{F}_{2^n}$, $t = \mathrm{Mer}[e]^{-1}(x)$, and $y = x^a$
  - Goal: For any $a \in \mathbb{Z}_{2^n-1}$, prove that introducing y does not generate an easy system to solve

Algebraic Analysis on AIM2

- Brute-force search of intermediate variables in a S-box
  - Variable: $x \in \mathbb{F}_{2^n}$, $t = \mathrm{Mer}[e]^{-1}(x)$, and $y = x^a$
  - Goal: For any $a \in \mathbb{Z}_{2^n-1}$, prove that introducing y does not generate an easy system to solve

- Result: Either of followings are checked by theoretically or experimentally
  1. The variable t is of high degree with respect to y
  2. The system does not generate sufficiently many quadratic equations
  3. The system only involves y-variables

Algebraic Analysis on AIM2

- Brute-force search of intermediate variables in a S-box
  - Variable: $x \in \mathbb{F}_{2^n}$, $t = \mathrm{Mer}[e]^{-1}(x)$, and $y = x^a$
  - Goal: For any $a \in \mathbb{Z}_{2^n-1}$, prove that introducing y does not generate an easy system to solve

- Result: Either of followings are checked by theoretically or experimentally
  1. The variable t is of high degree with respect to y
  2. The system does not generate sufficiently many quadratic equations
  3. The system only involves y-variables

	(e_1, Deg)	(e_2, Deg)	(e_3, Deg)	(e_*, Deg)	Complexity
AIM2-I	$(49,16)$	$(91,15)$	-	$(3,15)$	≥ 176.2 $(+29.2)$
AIM2-III	$(17,17)$	$(47,17)$	-	$(5,26)$	≥ 214.4 $(+2.1)$
AIM2-V	$(11,31)$	$(141,23)$	$(7,25)$	$(3,29)$	≥ 310.4 $(+32.7)$

Other Analysis on AIM2

- Exhaustive search
  - Saarinen's method is the fastest (by <1 bit)
  - Sliding 2 LFSRs standing for pt and pt^{-1}
  - Fast exhaustive search is not allowed since there is no low-degree system

Other Analysis on AIM2

- Exhaustive search
  - Saarinen's method is the fastest (by <1 bit)
  - Sliding 2 LFSRs standing for pt and pt^{-1}
  - Fast exhaustive search is not allowed since there is no low-degree system
- DC/LC
  - Almost same as AIM

Other Analysis on AIM2

- Exhaustive search
  - Saarinen's method is the fastest (by <1 bit)
  - Sliding 2 LFSRs standing for pt and pt^{-1}
  - Fast exhaustive search is not allowed since there is no low-degree system
- DC/LC
  - Almost same as AIM
- Grover's algorithm
  - MITM approach can reduce the depth of circuit
  - But AIM2 still costs more than AES

Other Analysis on AIM2

- Exhaustive search
  - Saarinen's method is the fastest (by <1 bit)
  - Sliding 2 LFSRs standing for pt and pt^{-1}
  - Fast exhaustive search is not allowed since there is no low-degree system
- DC/LC
  - Almost same as AIM
- Grover's algorithm
  - MITM approach can reduce the depth of circuit
  - But AIM2 still costs more than AES
- Quantum attacks
  - Complexities change but not critically
  - Always slower than Grover's algorithm

AIMer version 2.0

AIMer version 2.0

- Change of Specification
  - Symmetric primitive: AIM → AIM2
  - Prehashing now supported
  - Halved salt size
  - Reduced number of parameter sets (e.g., 128f, 128s)

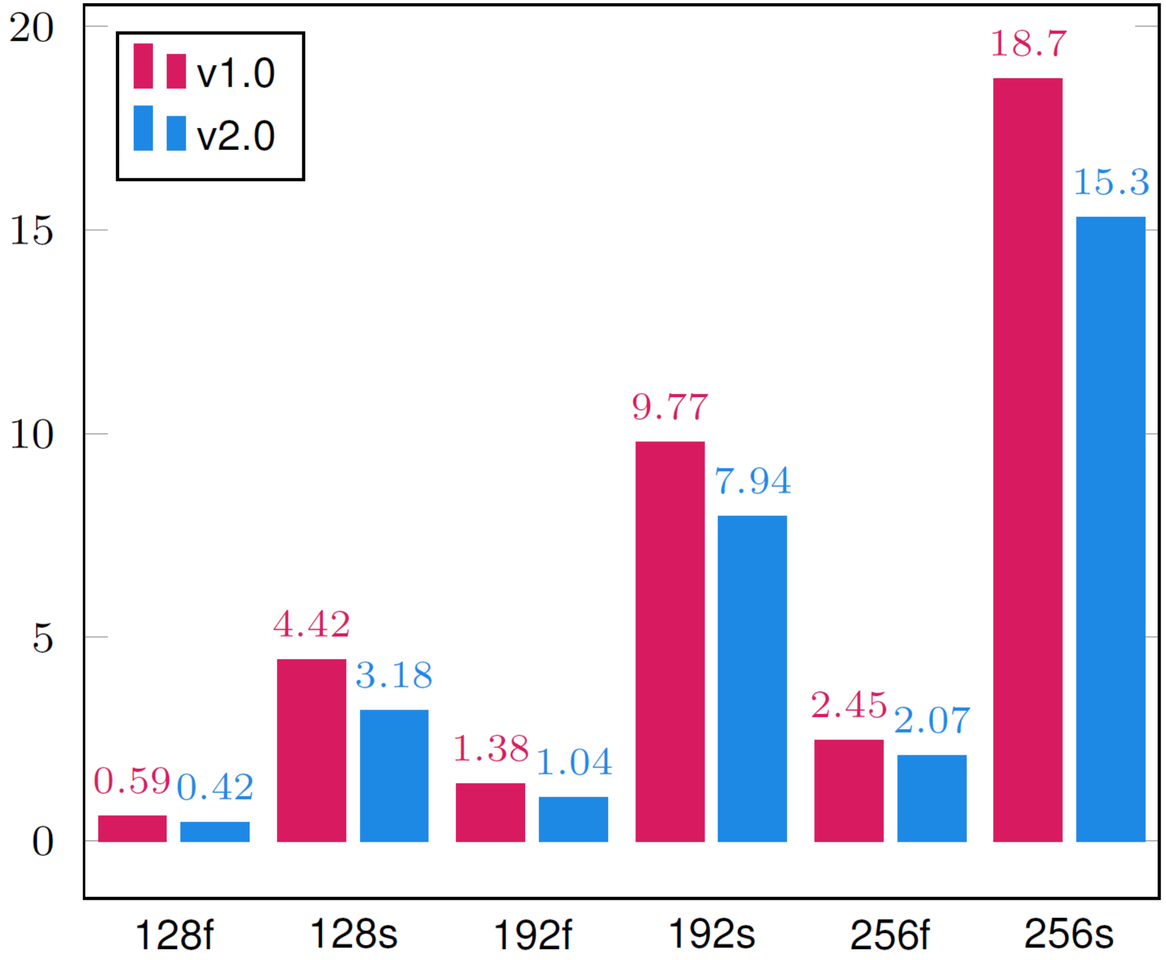
AIMer version 2.0

- Change of Specification
  - Symmetric primitive: AIM → AIM2
  - Prehashing now supported
  - Halved salt size
  - Reduced number of parameter sets (e.g., 128f, 128s)
- Change of Implementation
  - More readable reference code
  - Additional ARM64 implementation
  - Up to 29% faster signing on AVX2 than v1.0
  - Up to 96% less memory usage in verification

AIMer version 2.0

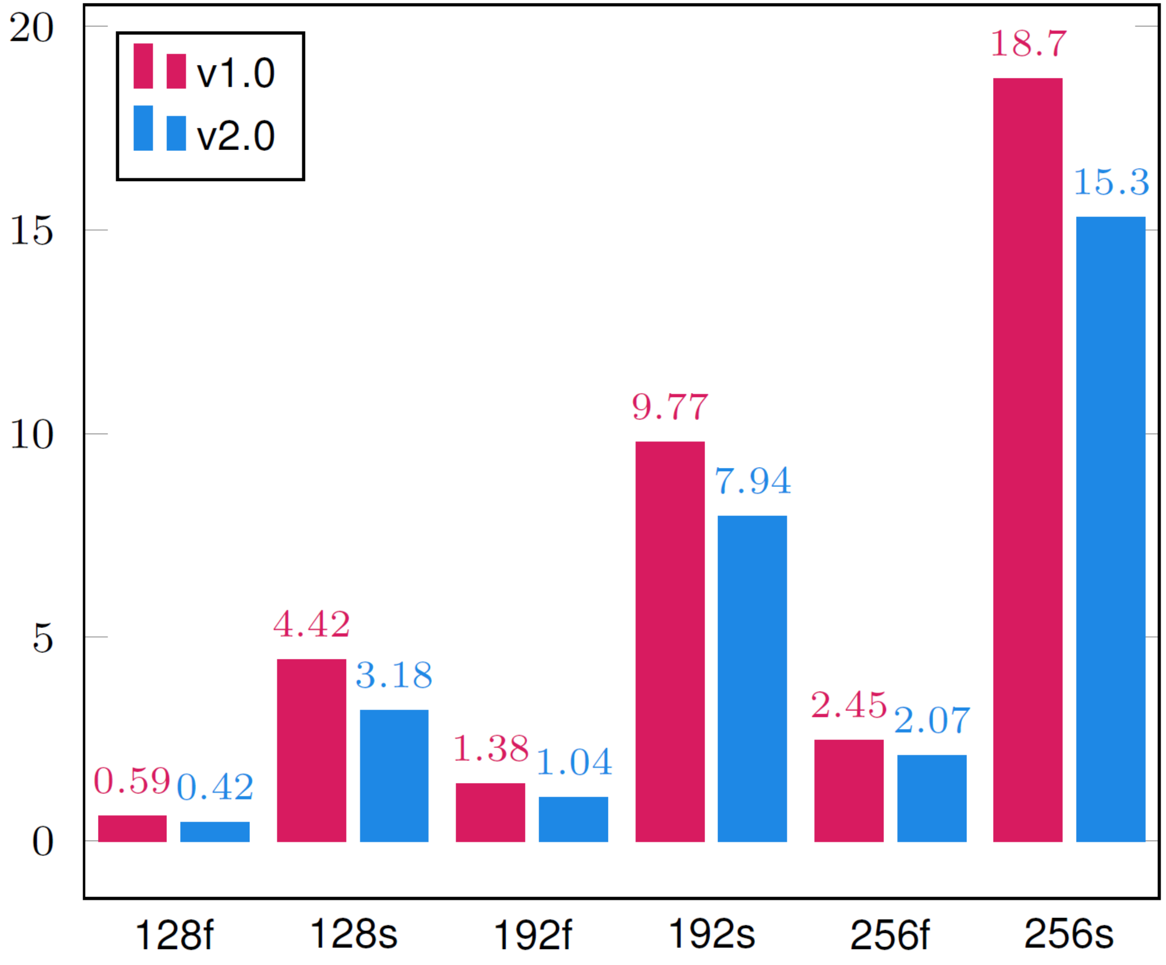
- Change of Specification
  - Symmetric primitive: AIM → AIM2
  - Prehashing now supported
  - Halved salt size
  - Reduced number of parameter sets (e.g., 128f, 128s)
- Change of Implementation
  - More readable reference code
  - Additional ARM64 implementation
  - Up to 29% faster signing on AVX2 than v1.0
  - Up to 96% less memory usage in verification
- Editorial Change
  - Improved EUF-CMA security proof (birthday bound → full bound)
  - Implementation-friendly specification

Performance Comparison



Signing time comparison in ms (AVX2)

Performance Comparison



Signing time comparison in ms (AVX2)

Scheme	pk (B)	sig (B)	Sign (ms)	Verify (ms)
Dilithium2	1312	2420	0.10	0.03
Falcon-512	897	690	0.27	0.04
SPHINCS+-128s	32	7856	315.74	0.35
SPHINCS+-128f	32	17088	16.32	0.97
AIMer v1.0	32	5904	0.59	0.53
AIMer v1.0	32	4176	4.42	4.31
AIMer v2.0	32	5888	0.42	0.41
AIMer v2.0	32	4160	3.18	3.13

Measured on Intel Xeon E5-1650 v3 @ 3.50 GHz with 128 GB RAM, TurboBoost and Hyper-threading disabled, gcc 7.5.0 with -O3 option

Conclusion

- Summary
  - We re-analyze the efficacy of recent analyses on AIM
  - We patched AIM to AIM2 to mitigate the analyses
  - AIMer v2.0 which contains AIM2 enjoys up to 29% faster signing

Conclusion

- Summary
  - We re-analyze the efficacy of recent analyses on AIM
  - We patched AIM to AIM2 to mitigate the analyses
  - AIMer v2.0 which contains AIM2 enjoys up to 29% faster signing
- Remark
  - We submitted AIMer to KpqC and NIST PQC competition
  - Our website: https://aimer-signature.org
  - We are waiting for **third-party analysis**!

Conclusion

- Summary
  - We re-analyze the efficacy of recent analyses on AIM
  - We patched AIM to AIM2 to mitigate the analyses
  - AIMer v2.0 which contains AIM2 enjoys up to 29% faster signing
- Remark
  - We submitted AIMer to KpqC and NIST PQC competition
  - Our website: https://aimer-signature.org
  - We are waiting for **third-party analysis**!
- Work in progress
  - We are implementing AIMer on ARM Cortex-M4 in an optimized form
    - Preliminary result: memory usage $\leq$ 110 KB for all parameter sets
  - We are improving the puncturable PRF in AIMer, and adopting AES-based PRG
    - Preliminary result: 4.8 KB (128f), 3.6 KB (128s)

Thank you!
Check out our website!