Efficacy and Mitigation of the Cryptanalysis on AIM

Seongkwang Kim¹
Jincheol Ha²
Mincheol Son²
Byeonghak Lee¹

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

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

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

- In AlMer, $pk = (iv, ct)$ and $sk = (pk, pt)$
Symmetric Primitive AIM

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

- Mersenne S-box
  - \( \text{Mer}[e](x) = x^{2^e-1} \)
  - 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
  - $\text{Mer}[e](x) = x^{2^e-1}$
  - 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
  - $(A_{iv}, b_{iv}) \leftarrow \text{XOF}(iv)$
  - $\text{Lin}(x) = A_{iv} \cdot x + b_{iv}$
Symmetric Primitive AIM

- Mersenne S-box
  - \( \text{Mer}[e](x) = x^{2^e-1} \)
  - 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
  - \((A_{iv}, b_{iv}) \leftarrow \text{XOF}(iv)\)
  - \(\text{Lin}(x) = A_{iv} \cdot x + b_{iv}\)

<table>
<thead>
<tr>
<th>Scheme</th>
<th>(\lambda)</th>
<th>(n)</th>
<th>(\ell)</th>
<th>(e_1)</th>
<th>(e_2)</th>
<th>(e_3)</th>
<th>(e_*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIM-I</td>
<td>128</td>
<td>128</td>
<td>2</td>
<td>3</td>
<td>27</td>
<td>-</td>
<td>5</td>
</tr>
<tr>
<td>AIM-III</td>
<td>192</td>
<td>192</td>
<td>2</td>
<td>5</td>
<td>29</td>
<td>-</td>
<td>7</td>
</tr>
<tr>
<td>AIM-V</td>
<td>256</td>
<td>256</td>
<td>3</td>
<td>3</td>
<td>53</td>
<td>7</td>
<td>5</td>
</tr>
</tbody>
</table>
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:
  \[ x \rightarrow x^{2^2-1} \rightarrow x^{2^3-1} \rightarrow x^{2^6-1} \rightarrow x^{2^{12}-1} \rightarrow x^{2^{24}-1} \rightarrow x^{2^{27}-1} \]
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

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

<table>
<thead>
<tr>
<th></th>
<th>Previous Cost</th>
<th>Current Cost</th>
<th>AES Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIM-I</td>
<td>149.0</td>
<td>146.3</td>
<td>143</td>
</tr>
<tr>
<td>AIM-III</td>
<td>214.4</td>
<td>211.8</td>
<td>207</td>
</tr>
<tr>
<td>AIM-V</td>
<td>280.0</td>
<td>276.7</td>
<td>272</td>
</tr>
</tbody>
</table>

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

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

Fast Exhaustive Search Attack (Liu et al.)

\[ \text{AIM}[\text{iv}](\text{pt}) = \text{ct} \]

\[ \iff F(x) = y \& \deg F = d \]
Fast Exhaustive Search Attack (Liu et al.)

- 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.

$$AIM[iv](pt) = ct \iff F(x) = y \& \deg F = d$$
Fast Exhaustive Search Attack (Liu et al.)

- 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.

$\text{AIM[iv]}(\text{pt}) = \text{ct}$

$\iff F(x) = y \ \& \ \text{deg} F = d$
Fast Exhaustive Search Attack (Liu et al.)

- Boolean polynomial system can be brute-force searched with $4d \log n \times 2^n$ 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)

<table>
<thead>
<tr>
<th></th>
<th>n</th>
<th>Deg</th>
<th>Log(Time) [bits]</th>
<th>Log(Mem) [bits]</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIM-I</td>
<td>128</td>
<td>10</td>
<td>136.2 ($-$10.1)</td>
<td>61.7</td>
</tr>
<tr>
<td>AIM-III</td>
<td>192</td>
<td>14</td>
<td>200.7 ($-$11.1)</td>
<td>84.3</td>
</tr>
<tr>
<td>AIM-V</td>
<td>256</td>
<td>15</td>
<td>265.0 ($-$11.7)</td>
<td>95.1</td>
</tr>
</tbody>
</table>

$\text{AIM}[\text{iv}](\text{pt}) = \text{ct} \iff F(x) = y \land \deg F = d$
Easier System to Solve (Liu)

\[ w = pt^{-1} \]
\[ \text{Mer}[e_i](pt) = w \cdot pt^{2e_i} \]
Easier System to Solve (Liu)

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

\[ w = pt^{-1} \]
\[ Mer[e_i](pt) = w \cdot pt^{2e_i} \]
Easier System to Solve (Liu)

• 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

\[ w = pt^{-1} \]
\[ \text{Mer}[e_i](pt) = w \cdot pt^{2e_i} \]
Easier System to Solve (Liu)

- 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)

\[
\begin{array}{|c|c|c|}
\hline
n & \text{Log(Time*) [bits]} & \text{Log(Time**) [bits]} \\
\hline
\text{AIM-I} & 128 & 124.8 (-18.8) & 158.3 (+14.4) \\
\text{AIM-III} & 192 & 157.5 (-54.3) & 226.5 (+14.7) \\
\text{AIM-V} & 256 & 188.9 (-87.8) & 290.2 (+13.5) \\
\hline
\end{array}
\]

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

\[ w = pt^{-1} \]
\[ \text{Mer}[e_i](pt) = w \cdot pt^{2e_i} \]
Efficient Exhaustive Search (Saarinen)

\[ w = \text{pt}^{-1} \]

\[ \text{Mer}[e_i](\text{pt}) = w \cdot \text{pt}^{2^e_i} \]
Efficient Exhaustive Search (Saarinen)

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

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

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

• Using LFSR for $\mathbb{F}_{2^n}$, exhaustive search on $x^{-1}$ is easy:
  $$x \ll_{\text{LFSR}} 1 = x \cdot \alpha \text{ in } \mathbb{F}_2[\alpha]/(f(\alpha))$$
  $$(x \ll_{\text{LFSR}} 1)^{-1} = x^{-1} \gg_{\text{LFSR}} 1$$

• The common inverse $w$ reduces the number of multiplications $\rightarrow$ Low complexity

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

• Using LFSR for $\mathbb{F}_{2^n}$, exhaustive search on $x^{-1}$ is easy:
  
  $x \ll_{\text{LFSR}} 1 = x \cdot \alpha$ in $\mathbb{F}_2[\alpha]/(f(\alpha))$

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

• The common inverse $w$ reduces the number of multiplications $\rightarrow$ Low complexity

• The result of Saarinen (new estimation)

\[
w = pt^{-1}
\]

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

<table>
<thead>
<tr>
<th></th>
<th>n</th>
<th>#mult</th>
<th>Log(Time) [bits]</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIM-I</td>
<td>128</td>
<td>3</td>
<td>145.0 ($-1.3$)</td>
</tr>
<tr>
<td>AIM-III</td>
<td>192</td>
<td>3</td>
<td>210.2 ($-1.6$)</td>
</tr>
<tr>
<td>AIM-V</td>
<td>256</td>
<td>4</td>
<td>275.5 ($-1.2$)</td>
</tr>
</tbody>
</table>
Inputs to parallel S-boxes are all the same
Structural Vulnerability (Zhang et al.)

• Find some $d|\left(2^n - 1\right)$ such that

\[\begin{align*}
\text{Mer}[e_1](\text{pt}) &= \left(\text{pt}^d\right)^{s_1} \cdot \text{pt}^{2^t_1} \\
\text{Mer}[e_2](\text{pt}) &= \left(\text{pt}^d\right)^{s_2} \cdot \text{pt}^{2^t_2} \\
\text{Mer}[e_3](\text{pt}) &= \left(\text{pt}^d\right)^{s_3} \cdot \text{pt}^{2^t_3}
\end{align*}\]

Inputs to parallel S-boxes are all the same
Structural Vulnerability (Zhang et al.)

- Find some $d|(2^n - 1)$ such that
  \[
  \begin{align*}
  \text{Mer}[e_1](pt) &= (pt^d)^{s_1} \cdot pt^{2t_1} \\
  \text{Mer}[e_2](pt) &= (pt^d)^{s_2} \cdot pt^{2t_2} \\
  \text{Mer}[e_3](pt) &= (pt^d)^{s_3} \cdot pt^{2t_3}
  \end{align*}
  \]

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

Inputs to parallel S-boxes are all the same
Structural Vulnerability (Zhang et al.)

- Find some $d|(2^n - 1)$ such that
  
  \[
  \begin{align*}
  \text{Mer}[e_1](pt) &= (pt^d)^{s_1} \cdot pt^{2^t_1} \\
  \text{Mer}[e_2](pt) &= (pt^d)^{s_2} \cdot pt^{2^t_2} \\
  \text{Mer}[e_3](pt) &= (pt^d)^{s_3} \cdot pt^{2^t_3}
  \end{align*}
  \]

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

Inputs to parallel S-boxes are all the same
Structural Vulnerability (Zhang et al.)

- Find some $d | (2^n - 1)$ such that
  
  \[
  \begin{align*}
  \text{Mer}[e_1](\text{pt}) &= (\text{pt}^d)^{s_1} \cdot \text{pt}^{2t_1} \\
  \text{Mer}[e_2](\text{pt}) &= (\text{pt}^d)^{s_2} \cdot \text{pt}^{2t_2} \\
  \text{Mer}[e_3](\text{pt}) &= (\text{pt}^d)^{s_3} \cdot \text{pt}^{2t_3}
  \end{align*}
  \]

- When $\text{pt}^d$ 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)

<table>
<thead>
<tr>
<th></th>
<th>$n$</th>
<th>$d$</th>
<th>Log(Time) [bits]</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIM-I</td>
<td>128</td>
<td>5</td>
<td>146.0 $(-0.3)$</td>
</tr>
<tr>
<td>AIM-III</td>
<td>192</td>
<td>45</td>
<td>210.4 $(-1.4)$</td>
</tr>
<tr>
<td>AIM-V</td>
<td>256</td>
<td>3</td>
<td>277.0 $(+0.3)$</td>
</tr>
</tbody>
</table>
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.

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

<table>
<thead>
<tr>
<th></th>
<th>FES (Liu et al.)</th>
<th>Easier System (Liu)</th>
<th>Efficient Search (Saarinen)</th>
<th>Linearization (Zhang et al.)</th>
<th>Exhaustive Search</th>
<th>AES Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIM-I</td>
<td>136.2 (−6.8)</td>
<td>158.3 (+17.7)</td>
<td>145.0 (+2.0)</td>
<td>146.0 (+3.0)</td>
<td>146.3</td>
<td>143</td>
</tr>
<tr>
<td>AIM-III</td>
<td>200.7 (−5.3)</td>
<td>226.5 (+19.5)</td>
<td>210.2 (+3.2)</td>
<td>210.4 (+3.4)</td>
<td>211.8</td>
<td>207</td>
</tr>
<tr>
<td>AIM-V</td>
<td>265.0 (−7.0)</td>
<td>290.2 (+18.2)</td>
<td>275.5 (+3.5)</td>
<td>277.0 (+5.0)</td>
<td>276.7</td>
<td>272</td>
</tr>
</tbody>
</table>
AIM2 and Analysis
AIM2: Secure Patch for Algebraic Attacks

- Inverse Mersenne S-box
  - $\text{Mer}[e]^{-1}(x) = x^a$
  - $a = (2^e - 1)^{-1} \mod (2^n - 1)$
  - More resistant to algebraic attacks
AIM2: Secure Patch for Algebraic Attacks

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

- Larger exponents
  - To mitigate fast exhaustive search
AIM2: Secure Patch for Algebraic Attacks

- Inverse Mersenne S-box
  - $\text{Mer}[e]^{-1}(x) = x^a$
  - $a = (2^e - 1)^{-1} \mod (2^n - 1)$
  - 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

- Inverse Mersenne S-box
  - $\text{Mer}[e]^{-1}(x) = x^a$
  - $a = (2^e - 1)^{-1} \mod (2^n - 1)$
  - 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
    $$(x^a)^b \text{ vs } (x^a + c)^b$$

<table>
<thead>
<tr>
<th>Scheme</th>
<th>$\lambda$</th>
<th>$n$</th>
<th>$\ell$</th>
<th>$e_1$</th>
<th>$e_2$</th>
<th>$e_3$</th>
<th>$e_*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIM2-I</td>
<td>128</td>
<td>128</td>
<td>2</td>
<td>49</td>
<td>91</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>AIM2-III</td>
<td>192</td>
<td>192</td>
<td>2</td>
<td>17</td>
<td>47</td>
<td>-</td>
<td>5</td>
</tr>
<tr>
<td>AIM2-V</td>
<td>256</td>
<td>256</td>
<td>3</td>
<td>11</td>
<td>141</td>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>
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 $a_{\alpha \beta \gamma}$:
    \[
    \sum_{\alpha,\gamma \in \mathbb{F}_2^n, \beta = (\beta_1, \ldots, \beta_n) \in \mathbb{F}_2^n} a_{\alpha \beta \gamma} x^{\alpha} t_i^{\beta} z^{\gamma} = 0
    \]
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 $a_{\alpha \beta \gamma}$:

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

  • 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 $a_{\alpha \beta \gamma}$:
    \[
    \sum_{\alpha, \gamma \in \mathbb{F}_2^n, \beta = (\beta_1, ..., \beta_\ell) \in \mathbb{F}_2^{\ell n}} a_{\alpha \beta \gamma} x^{\alpha} t_i^{\beta_i} z^\gamma = 0
    \]
    \[\text{with} \quad \text{hw}(\alpha) + \text{hw}(\beta) + \text{hw}(\gamma) \leq 2\]
  - 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 $a_{\alpha \beta \gamma}$:
    \[
    \sum_{\alpha, \gamma \in \mathbb{F}_2^n, \beta = (\beta_1, \ldots, \beta_\ell) \in \mathbb{F}_2^{\ell \cdot n}} a_{\alpha \beta \gamma} x^\alpha t_i^\beta z^\gamma = 0
    \]
    
  - 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

<table>
<thead>
<tr>
<th></th>
<th>#var</th>
<th>#eq</th>
<th>Log(Time) [bits]</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIM2-I</td>
<td>256</td>
<td>384</td>
<td>207.9 (+60.9)</td>
</tr>
<tr>
<td></td>
<td>384</td>
<td>1536</td>
<td>185.3 (+38.3)</td>
</tr>
<tr>
<td>AIM2-III</td>
<td>384</td>
<td>576</td>
<td>301.9 (+89.6)</td>
</tr>
<tr>
<td></td>
<td>576</td>
<td>2304</td>
<td>262.4 (+50.1)</td>
</tr>
<tr>
<td>AIM2-V</td>
<td>768</td>
<td>1536</td>
<td>503.7 (+226.0)</td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td>4608</td>
<td>411.4 (+133.7)</td>
</tr>
</tbody>
</table>
Algebraic Analysis on AIM2

• Brute-force search of intermediate variables in a S-box
  • Variable: $x \in \mathbb{F}_{2^n}$, $t = \text{Mer}[e]^{-1}(x)$, 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: $x \in \mathbb{F}_{2^n}$, $t = \text{Mer}[e]^{-1}(x)$, 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: \( x \in \mathbb{F}_{2^n} \), \( t = \text{Mer}[e]^{-1}(x) \), 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

<table>
<thead>
<tr>
<th></th>
<th>(( e_1 ), Deg)</th>
<th>(( e_2 ), Deg)</th>
<th>(( e_3 ), Deg)</th>
<th>(( e_* ), Deg)</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIM2-I</td>
<td>(49,16)</td>
<td>(91,15)</td>
<td></td>
<td>(3,15)</td>
<td>( \geq 176.2 ) (+29.2)</td>
</tr>
<tr>
<td>AIM2-III</td>
<td>(17,17)</td>
<td>(47,17)</td>
<td></td>
<td>(5,26)</td>
<td>( \geq 214.4 ) (+2.1)</td>
</tr>
<tr>
<td>AIM2-V</td>
<td>(11,31)</td>
<td>(141,23)</td>
<td>(7,25)</td>
<td>(3,29)</td>
<td>( \geq 310.4 ) (+32.7)</td>
</tr>
</tbody>
</table>
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

• 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 $\rightarrow$ 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 $\rightarrow$ full bound)
  • Implementation-friendly specification
Performance Comparison

Graph showing signing time comparison in ms (AVX2) for v1.0 and v2.0.
Performance Comparison

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

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
  • AlMer v2.0 which contains AIM2 enjoys up to 29% faster signing

• Remark
  • We submitted AlMer 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
  • AlMer v2.0 which contains AIM2 enjoys up to 29% faster signing

• Remark
  • We submitted AlMer 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 AlMer on ARM Cortex-M4 in an optimized form
    • Preliminary result: memory usage ≤ 110 KB for all parameter sets
  • We are improving the puncturable PRF in AlMer, and adopting AES-based PRG
    • Preliminary result: 4.8 KB (128f), 3.6 KB (128s)
Thank you!
Check out our website!