Revisiting the Computation Analysis against Internal Encodings in White-Box Implementations

Yufeng Tang, Zheng Gong, Bin Li and Liangju Zhao

yuft@m.scnu.edu.cn, cis.gong@gmail.com
School of Computer Science, South China Normal University

CHES 2023, Prague, Czech Republic
September 14, 2023
Outline

1. White-Box Cryptography and Internal Encodings
2. Revisiting the Computation Analysis
3. Algebraic Degree Computation Analysis against Internal Encodings
4. Conclusion
Outline

1. White-Box Cryptography and Internal Encodings
2. Revisiting the Computation Analysis
3. Algebraic Degree Computation Analysis against Internal Encodings
4. Conclusion
White-Box Attack Context

- **Black-Box**: Traditional cryptanalysis techniques, exploiting the inputs/outputs of the cryptographic algorithm. (Differential/Linear Cryptanalysis)

- **Gray-Box**: Execution in hardware, utilizing power/electromagnetic information or fault injection. (Differential Power/Fault Analysis)

- **White-Box**: Execution in software, full access to the cryptographic implementation, extracting the secret key in the memory by reverse engineering.
Encoding Frameworks in White-Box Cryptography

Goal
Preventing the cryptographic algorithm from key extraction in the white-box model.

Encoding Framework

- **CEJO** (SAC’02, Chow et al.): Key-dependent look-up tables, applying linear/non-linear encodings.

- **Self-Equivalence** (SAC’20, Ranea and Preneel): Key-dependent affine transformation, applying affine self-equivalence encodings.

- **Implicit Function** (CRYPTO’22, Ranea et al.): Key-dependent affine system, applying quadratic-affine self-equivalence encodings.
• Applying input and output encodings $I$ and $O$, a table $T$ can be transformed into $T'$ as

$$T' = O \circ T \circ I^{-1}$$

• Using pairwise invertible encodings $I_R^{-1} \circ O_T = F_{id}$, the composition $T' \circ R'$ yields:

$$R' \circ T' = (O_R \circ R \circ I_R^{-1}) \circ (O_T \circ T \circ I_T^{-1}) = O_R \circ (R \circ T) \circ I_T^{-1},$$

• Combining all the round functions:

$$E'_k = O_E \circ E_k \circ I_E,$$

where $I_E$ and $O_E$ are input and output external encodings, respectively.

• The other encodings which can be canceled in the inner rounds are Internal encodings.
• **Diffused-Input-Blocked-Output Function** (DIBO, CHES’21, Carlet *et al.*): Formalizing the internal encoding in the CEJO framework, which consists of a linear mapping and a concatenation of 4/8-bit non-linear permutations.
Attacks on White-Box Implementation

Structural Attack
- Such as BGE (SAC’04, Billet et al.) and collision attack (SAC’13, Lepoint et al.).
- Also called algebraic attack: recovering the encodings and extracting the secret key.
- Requiring the encoding details and reverse engineering effort.

Automatic Attack
- Such as power/fault analysis.
- Without the reverse engineering and the knowledge of encoding details.
1. White-Box Cryptography and Internal Encodings

2. Revisiting the Computation Analysis

3. Algebraic Degree Computation Analysis against Internal Encodings

4. Conclusion
DCA (CHES’16, Bos et al.)

- **Noise-Free Computation Traces**
  The first/last-round computed values (accessed memory/register) by using DBI tools.

- **Differential Power Analysis**
  Dividing the traces in two distinct sets based on key guesses, computing the difference of two sets, distinguishing the correct key with the highest peak.
DCA does not explain the vulnerability of internal encodings.

- FSE’16, Sasdrich et al. proposed the **Spectral Analysis** (SA) which utilizes the Walsh transforms to distinguish the correct key.
  - No analysis of vulnerable encodings.

- ACNS’18, Bock et al. proposed the **Improved DCA** (IDCA) to enumerate the linear combinations of sensitive values.
  - The **Hamming Weight (HW)** of linear encodings cause the key leakage.
The Published Computation Analyses

- CHES’19, Rivain and Wang introduced the **Correlation Power Analysis** (CPA), **Collision Attack** (CA), and **Mutual Information Analysis** (MIA) into the white-box setting.
  - Target function: $\mathbb{F}_2^m \rightarrow \mathbb{F}_2^n$ satisfying $m > n$ cause the key leakage.

- IEEE Access’20. Lee et al. proposed the **Modified Spectral Analysis** (MSA) to reduce the time complexity of trace computation.
  - The imbalance of linear encodings cause the key leakage.

- CHES’21. Carlet et al. proposed the **Improved Spectral Analysis** (ISA) to enhance SA as a structural attack.
  - The invertibility of linear encodings cause the key leakage (8-bit non-linear encoding).
### Countermeasures of the Published Computation Analyses

<table>
<thead>
<tr>
<th>Analysis</th>
<th>Countermeasure</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>8-bit non-linear</td>
</tr>
<tr>
<td>IDCA</td>
<td></td>
</tr>
<tr>
<td>MSA</td>
<td>non-invertible linear</td>
</tr>
<tr>
<td>CPA</td>
<td></td>
</tr>
<tr>
<td>MIA</td>
<td>non-invertible linear</td>
</tr>
<tr>
<td>CA</td>
<td>invertible linear</td>
</tr>
<tr>
<td>ISA (Structural Attack)</td>
<td>invertible linear + 8-bit non-linear</td>
</tr>
</tbody>
</table>

Encoding analysis mainly focuses on the properties of linear encodings.

- Lack of the analysis of properties of non-linear encodings.
- Lack of the comparison of the capabilities and time complexity among the published analyses.
Outline

1. White-Box Cryptography and Internal Encodings
2. Revisiting the Computation Analysis
3. Algebraic Degree Computation Analysis against Internal Encodings
4. Conclusion
A Refined 8-Bit Structure from DIBO

- A (32, 8)-bit DIBO can be refined as an (8, 8)-bit function.
- $L \leftarrow$ linear mapping, $N \leftarrow 4/8$-bit non-linear permutation.
- $F \leftarrow$ random function (without linear/non-linear part).
Key Guess Model

- $k^* \leftarrow$ secret key,
  Target function $O_{k^*} = F \circ S \circ \oplus k^*$.
- $k \leftarrow$ key guess,
  Chosen input $x' = \oplus k \circ S^{-1}(x)$.
- If $k = k^*$, $A_{k^*} = F$ which is the internal encoding.
- If $k \neq k^*$, $A_k = F \circ S \circ \oplus k^* \circ \oplus k \circ S^{-1}$ which a random function.
Distinguish $A_{k^*}$ and $A_k$

- The algebraic degree of the AES Sbox is 7.
- If $k \neq k^*$, high probability:
  \[ A_k = F \circ S \circ \oplus k^* \circ \oplus k \circ S^{-1} \Rightarrow d_{alg}(A_k) = 7. \]
- If $k = k^*$, $A_{k^*} = F \Rightarrow d_{alg}(A_{k^*}) = d_{alg}(F)$, different constructed $F$ have various degrees.
- If $d_{alg}(F) < 7$,
  \[ \#\{d_{alg}(A_{k^*}) < 7\} > \#\{d_{alg}(A_k) < 7\}, \]
can distinguish the correct key.
The combined encoding has degree $\leq 3$.

Each Boolean function has degree $< 7$.

$\#\{d_{alg}(A_k^*) < 7\} > \#\{d_{alg}(A_k) < 7\}$, can distinguish the correct key.
Non-Invertible Linear + 8-bit Non-Linear Encodings

- The combined encoding has degree $\leq 7$.
- The number of Boolean functions with degrees $< 7$ is from 0 to 8.

For most cases,
- $\#\{d_{\text{alg}}(A_k^*) < 7\} > \#\{d_{\text{alg}}(A_k) < 7\}$, can distinguish the correct key.

<table>
<thead>
<tr>
<th>Case</th>
<th>Constriction of $L$</th>
<th>Constriction of $N$</th>
<th>$d_{\text{alg}}(F_i) &lt; 7$</th>
<th>$d_{\text{alg}}((A_k)_i) &lt; 7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(non-)invertible</td>
<td>nibble</td>
<td>8 (100%)</td>
<td>2 (0.07%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 (3.15%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 (96.82%)</td>
</tr>
<tr>
<td>2</td>
<td>non-invertible</td>
<td>byte-level</td>
<td>8 (18.19%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7 (2.51%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6 (8.69%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5 (17.97%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4 (22.33%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 (18.38%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 (9.20%)</td>
<td>$&gt;$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 (2.42%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 (0.31%)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>invertible</td>
<td>byte-level</td>
<td>2 (0.05%)</td>
<td>2 (0.03%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 (3.12%)</td>
<td>1 (2.96%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 (96.83%)</td>
<td>0 (97.04%)</td>
</tr>
</tbody>
</table>
Non-Invertible Linear + 8-bit Non-Linear Encodings

• Computing the probability that cannot distinguish the correct key, satisfying

\[ \#\{d_{\text{alg}}(A_k^*) < 7\} \leq \#\{d_{\text{alg}}(A_k) < 7\} \]

⇒ The probability that can distinguish the correct key: \( 1 - 0.37902\% = 99.62098\% \).
Analysis of Linear/Non-Linear Encodings

Invertible Linear + 8-bit Non-Linear Encodings

- The combined encoding has degree $= 7$.
- Each Boolean function has degree $= 7$ with high probability.

For most cases,
- $\#\{d_{\text{alg}}(A_k^*) < 7\} = \#\{d_{\text{alg}}(A_k) < 7\}$, cannot distinguish the correct key.

<table>
<thead>
<tr>
<th>Case</th>
<th>Constriction of $L$</th>
<th>Constriction of $N$</th>
<th>$d_{\text{alg}}(F_i) &lt; 7$</th>
<th>$d_{\text{alg}}((A_{k+})_i) &lt; 7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(non-)invertible</td>
<td>nibble</td>
<td>8 (100%)</td>
<td>2 (0.07%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 (3.15%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 (96.82%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>8 (18.19%)</td>
<td>7 (2.51%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6 (8.69%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5 (17.97%)</td>
<td>2 (0.04%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4 (22.33%)</td>
<td>1 (2.66%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 (18.38%)</td>
<td>0 (97.30%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 (9.20%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 (2.42%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 (0.31%)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>non-invertible</td>
<td>byte-level</td>
<td>2 (0.05%)</td>
<td>2 (0.03%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 (3.12%)</td>
<td>1 (2.96%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 (96.83%)</td>
<td>0 (97.04%)</td>
</tr>
<tr>
<td>3</td>
<td>invertible</td>
<td>byte-level</td>
<td>2 (0.05%)</td>
<td>2 (0.03%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 (3.12%)</td>
<td>1 (2.96%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 (96.83%)</td>
<td>0 (97.04%)</td>
</tr>
</tbody>
</table>
Analysis of Linear/Non-Linear Encodings

Invertible Linear + 8-bit Non-Linear Encodings

• Computing the probability that can distinguish the correct key, satisfying

\[\#\{d_{\text{alg}}(A_{k^*}) < 7\} > \#\{d_{\text{alg}}(A_k) < 7\}\]

⇒ The probability that can distinguish the correct key: 3.077648%.

<table>
<thead>
<tr>
<th>The Number of Coordinates</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>(d_{\text{alg}} ((A_{k^*})_i) &lt; 7)</td>
<td>(d_{\text{alg}} ((A_k)_i) &lt; 7)</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td></td>
</tr>
</tbody>
</table>
A Generic Tool: Algebraic Degree Computation Analysis

ADCA

- To reveal the vulnerability of the non-linear encodings based on the algebraic degrees.
- A higher-degree attack to match the degrees of the mappings from the inputs to the intermediate variables.
- A flexible time complexity for breaking different constructed encodings.
Procedures of ADCA

Three Steps of ADCA

1. Collecting the computation traces by choosing inputs $x' = S^{-1}(x) \oplus k$ for a key guess $k \in \mathbb{F}_{2}^{8}$. The obtained traces $\mathbf{v} = (v_1, v_2, \cdots, v_T)$ contain the outputs of coordinate functions $(A_k(x))_i$ where $1 \leq i \leq 8$.

2. For each Boolean function $(f_k)_j : \mathbb{F}_2^8 \mapsto \mathbb{F}_2 : (f_k)_j(x) = v_j$ where $j \in [T]$ and $k \in \mathbb{F}_2^8$, computes the degree $d_{alg} ((f_k)_j)$.

3. For a given degree $d$, searching the correct key $k^*$ that has the maximal numbers of $d_{alg} ((f_k^*)_j) \leq d$ where $j \in [T]$. 
Computation of Algebraic Degree

- **Degree-\(d\) combination** of each bit in \(x = (x_1, x_2, \cdots x_8)\).
- For instance, \(d = 2\), \((1, x_1, x_2, \cdots, x_8, x_1x_2, \cdots, x_7x_8)\).
- For \(N\) inputs and the \(v_j\)-th sample of traces, constructing the following linear system:

\[
\begin{pmatrix}
1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_{9-d}^{(1)} & x_8^{(1)} \\
1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_{9-d}^{(2)} & x_8^{(2)} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
1 & x_1^{(N)} & x_2^{(N)} & \cdots & x_{9-d}^{(N)} & x_8^{(N)}
\end{pmatrix}
\begin{pmatrix}
a_1 \\
a_2 \\
\vdots \\
a_p
\end{pmatrix}
= \begin{pmatrix}
(f_k)_j(x^{(1)}) \\
(f_k)_j(x^{(2)}) \\
\vdots \\
(f_k)_j(x^{(N)})
\end{pmatrix}
= \begin{pmatrix}
v_j^{(1)} \\
v_j^{(2)} \\
\vdots \\
v_j^{(N)}
\end{pmatrix}
\]

- If the system is solvable, \(d_{\text{alg}}((f_k)_j) \leq d\).
- The resulting vector \((a_1, a_2, \cdots, a_p)\) represents the coefficients of ANF.
ADCA Distinguisher

• For degree-$d$ internal encodings, degree-$d$ ADCA computes the number of Boolean functions with degree $d$.

• ADCA Distinguisher:

$$\delta_{ADCA}^k = \arg \max \# \{ d_{alg} \left( (f_k)_j \right) \leq d \mid j \in [T] \}$$

• For defeating 4-bit non-linear encodings: $d = 3$.

• A general attack against the 8-bit DIBO: $d = 6$. 
For $D \cdot a = v$, if $r(D) \geq r(D | v)$, the linear system is solvable.

- The coefficient matrix $D$ is independent of key and traces.
- The Gauss Elimination of $D$ can be pre-computed with time complexity $r = O(N \cdot p)$.

- The time complexity of ADCA is $O(r \cdot T \cdot |k|)$, where $r$ can be computed precisely:

<table>
<thead>
<tr>
<th>Degree</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r$</td>
<td>$2^{10.32}$</td>
<td>$2^{11.63}$</td>
<td>$2^{12.42}$</td>
<td>$2^{12.88}$</td>
<td>$2^{13.04}$</td>
<td>$2^{13.07}$</td>
<td>$2^{13.10}$</td>
</tr>
<tr>
<td>Time Complexity</td>
<td>$2^{21.32}$</td>
<td>$2^{22.63}$</td>
<td>$2^{23.42}$</td>
<td>$2^{23.88}$</td>
<td>$2^{24.04}$</td>
<td>$2^{24.07}$</td>
<td>$2^{24.10}$</td>
</tr>
</tbody>
</table>
Comparison of Time Complexity

- To attack the 8-bit DIBO function with 256 inputs and traces.
- ADCA has the lowest time complexity.

<table>
<thead>
<tr>
<th>Analysis</th>
<th>DCA</th>
<th>IDCA</th>
<th>CPA</th>
<th>CA</th>
<th>MIA</th>
<th>MSA</th>
<th>SA</th>
<th>ISA</th>
<th>ADCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>$2^{22}$</td>
<td>$2^{27}$</td>
<td>$2^{22}$</td>
<td>$2^{22}$</td>
<td>$2^{22}$</td>
<td>$2^{22}$</td>
<td>$2^{27}$</td>
<td>$2^{32}$</td>
<td>$2^{21.32} \sim 2^{24.07}$</td>
</tr>
</tbody>
</table>
Random encodings with degrees from 1 to 7, without the linear/non-linear part.

- For degrees from 1 to 6, each coordinate function has degree < 7, can distinguish the correct key.
- For degree 7, each coordinate function has degree = 7 with high probability, cannot distinguish the correct key.
Experiment Setup

Combined Encodings (CE): 14 Cases

- Invertible ($HW > 1$) linear + 4-bit non-linear.
- Invertible ($HW = 1$) linear + 4-bit non-linear.
- Non-invertible ($HW > 1$) linear + 4-bit non-linear.
- Non-invertible ($HW = 1$) linear + 4-bit non-linear.
- Invertible ($HW > 1$) linear + 8-bit non-linear.
- Invertible ($HW = 1$) linear + 8-bit non-linear.
- Non-invertible ($HW > 1$) linear + 8-bit non-linear.
- Non-invertible ($HW = 1$) linear + 8-bit non-linear.
- 4-bit non-linear
- 8-bit non-linear.

Random Encodings (RE): 7 Cases

- Degrees from 1 to 7.
## Experimental Results

### The Failed Cases of SA, ISA, and ADCA

- **3 cases of CE:**
  - Invertible ($\text{HW} > 1$) linear + 8-bit non-linear.
  - Invertible ($\text{HW} = 1$) linear + 8-bit non-linear.
  - 8-bit non-linear
  \[ \Rightarrow \text{Invertible (Identity) + 8-bit non-linear} \]

- **1 case of RE:**
  - Degree 7.

### Table: Computation Analysis

<table>
<thead>
<tr>
<th>Computation Analysis</th>
<th>The Number of Broken Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CE</td>
</tr>
<tr>
<td>DCA</td>
<td>3</td>
</tr>
<tr>
<td>IDCA</td>
<td>9</td>
</tr>
<tr>
<td>CPA</td>
<td>3</td>
</tr>
<tr>
<td>CA</td>
<td>3</td>
</tr>
<tr>
<td>MIA</td>
<td>3</td>
</tr>
<tr>
<td>SA</td>
<td>11</td>
</tr>
<tr>
<td>MSA</td>
<td>1</td>
</tr>
<tr>
<td>ISA</td>
<td>11</td>
</tr>
<tr>
<td>ADCA</td>
<td>11</td>
</tr>
</tbody>
</table>
Comparison among ADCA and Other Analyses

Compares with DCA, IDCA, CPA, CA, MIA, and ISA

- ADCA can break the most cases.

Compares with SA

- SA lacks of encoding analysis.
- ADCA reveals the weakness of the algebraic degree.

Compares with ISA

- ISA is a structural attack, focuses on the invertibility of linear encoding.
- ADCA is an automatic gray-box attack, focuses on the degree of random encoding.
1. White-Box Cryptography and Internal Encodings

2. Revisiting the Computation Analysis

3. Algebraic Degree Computation Analysis against Internal Encodings

4. Conclusion
Conclusion

• The vulnerability against computation analysis is related to the degree of encodings.

• Each coordinate function of the encoding with degree 7 is a countermeasure.

• ADCA can be a general computation analysis tool with the lowest time complexity.

Code

https://github.com/scnucrypto/Revisit_Computation_Analysis
Future Work

Computation analysis defeat the external encodings?

• At SAC’19, Amadori et al. mounted the Differential Fault Analysis (DFA) on a white-box AES with 8-bit external encodings.

• The published computation analyses all require the absence of input or output external encoding.
Thanks for your attention!