On the Simplicity of Converting Leakages from Multivariate to Univariate


Amir Moradi, Oliver Mischke
Embedded Security Group + Hardware Security Group
Ruhr University Bochum, Germany
Outline

- Definitions, Masking, etc.
- Target masking scheme
- The story behind our findings
- Practical issues
Masking

- Well-known SCA countermeasure
Masking

- Well-known SCA countermeasure
- to make the SC leakages independent of expected intermediate values
Masking

- Well-known SCA countermeasure
- to make the SC leakages independent of expected intermediate values
- Randomness is required
Masking

- Well-known SCA countermeasure
- to make the SC leakages independent of expected intermediate values
- Randomness is required
- Let’s consider the most common one, Boolean Masking
Masking

- Well-known SCA countermeasure
- to make the SC leakages independent of expected intermediate values
- Randomness is required
- Let’s consider the most common one, Boolean Masking

\[
P \rightarrow p \oplus k \rightarrow S(p \oplus k)
\]
Masking

- Well-known SCA countermeasure
- to make the SC leakages independent of expected intermediate values
- Randomness is required
- Let’s consider the most common one, Boolean Masking

\[ S(p \oplus k) \]

\[ S(p \oplus k) \oplus m' \]
Univariate vs. Multivariate Attacks

\[ S(p \oplus k) \]

masked

\[ S(p \oplus k) \oplus m' \]
Univariate vs. Multivariate Attacks

\[ S(p \oplus k) \]

masked

\[ S(p \oplus k) \oplus m' \]

\[ S(p \oplus k) \oplus m' \]
Univariate vs. Multivariate Attacks

\[ p \xrightarrow{k} \text{Sbox} \xrightarrow{S(p \oplus k)} p \xrightarrow{k} \text{masked Sbox} \xrightarrow{S(p \oplus k) \oplus m'} \]

Univariate vs. Multivariate Attacks

\[ S(p \oplus k) \]

Masked Sbox

\[ S(p \oplus k) \oplus m' \]
Univariate vs. Multivariate Attacks

$$\begin{align*}
Sbox & \rightarrow S(p \oplus k) \\
\rightarrow & \quad \text{masked}\ Sbox \quad \rightarrow \quad S(p \oplus k) \oplus m'
\end{align*}$$
Univariate vs. Multivariate Attacks

\[ S(p \oplus k) \]

DPA/CPA/MIA
Univariate vs. Multivariate Attacks

\[ S(p \oplus k) \]

DPA/CPA/MIA

\[ S(p \oplus k) \oplus m' \]

bivariate MIA

combining: DPA/CPA
Univariate vs. Multivariate Attacks

\[ S(p \oplus k) \]

- **DPA/CPA/MIA**

- bivariate MIA

combining: DPA/CPA

multiply: 2\textsuperscript{nd} order bivariate
Univariate vs. Multivariate Attacks

DPA/CPA/MIA

squaring: 2\textsuperscript{nd} order univariate

bivariate MIA

combining: DPA/CPA

multiply: 2\textsuperscript{nd} order bivariate
Univariate vs. Multivariate Attacks

**DPA/CPA/MIA**

- **squaring**: 2\(^{nd}\) order univariate

**bivariate MIA**

- **combining**: DPA/CPA
- **multiply**: 2\(^{nd}\) order bivariate
- **addition**: 1\(^{st}\) order bivariate
Masking in Hardware

\[ i \oplus m \xrightarrow{\text{masked Sbox}} S(i) \oplus m' \]
Masking in Hardware

- Pre-computing the masked tables in software

\[ i \oplus m \rightarrow \text{masked Sbox} \rightarrow S(i) \oplus m' \]
Masking in Hardware

- Pre-computing the masked tables in software
  - Sequential operations, Time consuming, Low efficiency

\[ i \oplus m \rightarrow \text{masked Sbox} \rightarrow S(i) \oplus m' \]
Masking in Hardware

- Pre-computing the masked tables in software
  - Sequential operations, Time consuming, Low efficiency
- High efficiency is desired in HARDWARE
  - amongst the main reasons

\[ i \oplus m \xrightarrow{\text{masked Sbox}} S(i) \oplus m' \]
Masking in Hardware

- Pre-computing the masked tables in software
  - Sequential operations, Time consuming, Low efficiency
- High efficiency is desired in HARDWARE
  - amongst the main reasons
- ad-hoc/heuristic schemes
Masking in Hardware

- Pre-computing the masked tables in software
  - Sequential operations, Time consuming, Low efficiency
- High efficiency is desired in HARDWARE
  - amongst the main reasons
- ad-hoc/heuristic schemes
- Processing the mask ($m$) and masked data ($i \oplus m$) simultaneously
Masking in Hardware

- Pre-computing the masked tables in software
  - Sequential operations, Time consuming, Low efficiency
- High efficiency is desired in HARDWARE
  - amongst the main reasons
- ad-hoc/heuristic schemes
- Processing the mask ($m$) and masked data ($i \oplus m$) simultaneously
  - joint distribution of SC leakages mainly because of GLITCHES
  - possible attacks
Masking in Hardware

- Pre-computing the masked tables in software
  - Sequential operations, Time consuming, Low efficiency
- High efficiency is desired in HARDWARE
  - amongst the main reasons
- ad-hoc/heuristic schemes
- Processing the mask \( m \) and masked data \( i \oplus m \) simultaneously
  - joint distribution of SC leakages mainly because of GLITCHES
  - possible attacks
- Systematic schemes
Masking in Hardware

- Pre-computing the masked tables in software
  - Sequential operations, Time consuming, Low efficiency
- High efficiency is desired in HARDWARE
  - amongst the main reasons
- ad-hoc/heuristic schemes
- Processing the mask ($m$) and masked data ($i \oplus m$) simultaneously
  - joint distribution of SC leakages mainly because of GLITCHES
  - possible attacks
- Systematic schemes
  - Threshold Implementation, Security against 1st order attacks
Masking in Hardware

- Pre-computing the masked tables in software
  - Sequential operations, Time consuming, Low efficiency
- High efficiency is desired in HARDWARE
  - amongst the main reasons
- ad-hoc/heuristic schemes
- Processing the mask \( m \) and masked data \( i \oplus m \) simultaneously
  - joint distribution of SC leakages mainly because of GLITCHES
  - possible attacks
- Systematic schemes
  - Threshold Implementation, Security against 1\(^{st}\) order attacks

Desired: security against univariate attacks of any order
Target Scheme

Target Scheme

- Multi-party computation + Shamir’s secret sharing
Target Scheme

- Multi-party computation + Shamir’s secret sharing
- Basic GF(2^8) operations, e.g., addition is easy
Target Scheme

- Multi-party computation + Shamir’s secret sharing
- Basic GF(2^8) operations, e.g., addition is easy
  - Multiplication needs more effort
Target Scheme

- Multi-party computation + Shamir’s secret sharing
- Basic GF(2^8) operations, e.g., addition is easy
  - Multiplication needs more effort
- An Sbox computation
Target Scheme

- Multi-party computation + Shamir’s secret sharing
- Basic GF($2^8$) operations, e.g., addition is easy
  - Multiplication needs more effort
- An Sbox computation

- Our goal
  - Hardware implementation using minimum settings
  - Using a Virtex-5 FPGA (SASEBO-GII)
Target Scheme - Design
Target Scheme - Design
Target Scheme - Design
Target Scheme - Design
Target Scheme - Design
Target Scheme - Design
Target Scheme - Performance
Target Scheme - Performance

- 66 clock cycles for Inversion, 66 clock cycles for Affine
  - One Sbox in 132 clock cycles
Target Scheme - Performance

- 66 clock cycles for Inversion, 66 clock cycles for Affine
  - One Sbox in 132 clock cycles
- One full SubBytes in $132 \times 16 = 2112$ clock cycles
Target Scheme - Performance

- 66 clock cycles for Inversion, 66 clock cycles for Affine
  - One Sbox in 132 clock cycles
- One full SubBytes in $132 \times 16 = 2112$ clock cycles
- One full MixColumns + AddRoundKey in $12 \times 16 = 192$ clock cycles
Target Scheme - Performance

- 66 clock cycles for Inversion, 66 clock cycles for Affine
  - One Sbox in 132 clock cycles
- One full SubBytes in $132 \times 16 = 2112$ clock cycles
- One full MixColumns + AddRoundKey in $12 \times 16 = 192$ clock cycles
Target Scheme - Performance

- 66 clock cycles for **Inversion**, 66 clock cycles for **Affine**
  - One Sbox in 132 clock cycles
- One full SubBytes in $132 \times 16 = 2112$ clock cycles
- One full MixColumns + AddRoundKey in $12 \times 16 = 192$ clock cycles

<table>
<thead>
<tr>
<th>Design</th>
<th>FF</th>
<th>LUT</th>
<th>Slice</th>
<th>SB CLK</th>
<th>MC + ARK CLK</th>
<th>Encryption CLK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 SB MC</td>
<td>315</td>
<td>1%</td>
<td>1387</td>
<td>5%</td>
<td>859</td>
<td>12%</td>
</tr>
<tr>
<td>16 SB MC</td>
<td>4275</td>
<td>15%</td>
<td>21328</td>
<td>74%</td>
<td>no fit</td>
<td>132</td>
</tr>
</tbody>
</table>

Target Scheme - Performance

- 66 clock cycles for **Inversion**, 66 clock cycles for **Affine**
  - One Sbox in 132 clock cycles
- One full SubBytes in $132 \times 16 = 2112$ clock cycles
- One full MixColumns + AddRoundKey in $12 \times 16 = 192$ clock cycles

- Hard to convince the industry sector?

<table>
<thead>
<tr>
<th>Design</th>
<th>FF #</th>
<th>FF %</th>
<th>LUT #</th>
<th>LUT %</th>
<th>Slice #</th>
<th>Slice %</th>
<th>SB CLK</th>
<th>MC+ARK CLK</th>
<th>Encryption CLK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 SB MC</td>
<td>315</td>
<td>1%</td>
<td>1387</td>
<td>5%</td>
<td>859</td>
<td>12%</td>
<td>2112</td>
<td>192</td>
<td>22 896</td>
</tr>
<tr>
<td>16 SB MC</td>
<td>4275</td>
<td>15%</td>
<td>21 328</td>
<td>74%</td>
<td>no fit</td>
<td></td>
<td>132</td>
<td>12</td>
<td>1431</td>
</tr>
</tbody>
</table>
Target Scheme - Performance

- 66 clock cycles for **Inversion**, 66 clock cycles for **Affine**
  - One Sbox in 132 clock cycles
- One full SubBytes in $132 \times 16 = 2112$ clock cycles
- One full MixColumns + AddRoundKey in $12 \times 16 = 192$ clock cycles

<table>
<thead>
<tr>
<th>Design</th>
<th>FF #</th>
<th>FF %</th>
<th>LUT #</th>
<th>LUT %</th>
<th>Slice #</th>
<th>Slice %</th>
<th>SB CLK</th>
<th>MC+ARK CLK</th>
<th>Encryption CLK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 SB MC</td>
<td>315</td>
<td>1%</td>
<td>1387</td>
<td>5%</td>
<td>859</td>
<td>12%</td>
<td>2112</td>
<td>192</td>
<td>22896</td>
</tr>
<tr>
<td>16 SB MC</td>
<td>4275</td>
<td>15%</td>
<td>21328</td>
<td>74%</td>
<td>no fit</td>
<td></td>
<td>132</td>
<td>12</td>
<td>1431</td>
</tr>
</tbody>
</table>

- Hard to convince the industry sector?
- Getting close to software?
Target Scheme - Performance

- 66 clock cycles for **Inversion**, 66 clock cycles for **Affine**
  - One Sbox in 132 clock cycles
- One full SubBytes in $132 \times 16 = 2112$ clock cycles
- One full MixColumns + AddRoundKey in $12 \times 16 = 192$ clock cycles

Hard to convince the industry sector?
- getting close to software?
- Gaining univariate resistance at what price?
Target Scheme - Evaluation
Target Scheme - Evaluation

- A variant by processing all three shares at the same time
Target Scheme - Evaluation
Target Scheme - Evaluation

- A variant by processing all three shares at the same time
Target Scheme - Evaluation

- A variant by processing all three shares at the same time
Target Scheme - Evaluation

- A variant by processing all three shares at the same time
Target Scheme - Evaluation

- A variant by processing all three shares at the same time
Target Scheme - Evaluation

- A variant by processing all three shares at the same time
Target Scheme - Evaluation

- Original Design, 3MHz
Target Scheme - Evaluation

- Original Design, 3MHz
Target Scheme - Evaluation

- Original Design, 3MHz

![Graphs showing voltage, ln(MI), and correlation over time for Target Scheme evaluation.](image)
Target Scheme - Evaluation

- Original Design, 3MHz
Measurement Setup
Measurement Setup
Measurement Setup
Measurement Setup
Measurement Setup
Measurement Setup
Measurement Setup

Standard Setup
Measurement Setup
Target Scheme – Evaluation (Standard Setup)

- Original Design, 3MHz
Standard vs. Amplified Setup

- Exemplary Design
Standard vs. Amplified Setup

- Exemplary Design
Standard vs. Amplified Setup

- Exemplary Design
Standard vs. Amplified Setup

- Exemplary Design

![Graphs showing voltage and correlation over time for standard and amplified setup.]
Standard vs. Amplified Setup

- Exemplary Design
Efficiency as a Factor
Efficiency as a Factor

Summary

Figure 3.12: The power consumption of the AES ASIC during four clock cycles. A different clock frequency has been used for each of the four traces.

Power Analysis Attacks
Revealing the Secrets of Smart Cards

Stefan Mangard
Elisabeth Oswald
Thomas Popp
Efficiency as a Factor

- Original Design, Standard Setup, 24MHz
Efficiency as a Factor

- Original Design, Standard Setup, 24MHz
Efficiency as a Factor

- Original Design, Standard Setup, 24MHz
Efficiency as a Factor

- Original Design, Standard Setup, 24MHz
Summing Up / Future Issues
Summing Up / Future Issues

- Cost of univariate resistance
  - security-performance tradeoff
  - processing the shares consecutively
Summing Up / Future Issues

- Cost of univariate resistance
  - security-performance tradeoff
  - processing the shares consecutively

- a light at the end of the tunnel by [pure] masking in hardware?
Summing Up / Future Issues

- Cost of univariate resistance
  - security-performance tradeoff
  - processing the shares consecutively

- a light at the end of the tunnel by [pure] masking in hardware?
  - slowly reaching the software performance?
    - making a processor by giant hardware?
Summing Up / Future Issues

- Cost of univariate resistance
  - security-performance tradeoff
  - processing the shares consecutively
- a light at the end of the tunnel by [pure] masking in hardware?
  - slowly reaching the software performance?
    - making a processor by giant hardware?
  - relatively easy ways to combine the leakages
    - measurement setup & high clock freq.
Summing Up / Future Issues

- Cost of univariate resistance
  - security-performance tradeoff
  - processing the shares consecutively

- a light at the end of the tunnel by [pure] masking in hardware?
  - slowly reaching the software performance?
    - making a processor by giant hardware?
  - relatively easy ways to combine the leakages
    - measurement setup & high clock freq.

- What to do when evaluating a countermeasure / product?
Summing Up / Future Issues

- Cost of univariate resistance
  - security-performance tradeoff
  - processing the shares consecutively

- a light at the end of the tunnel by [pure] masking in hardware?
  - slowly reaching the software performance?
    - making a processor by giant hardware?
  - relatively easy ways to combine the leakages
    - measurement setup & high clock freq.

- What to do when evaluating a countermeasure / product?
  - without any addition/modification on measurement setup?
    - not fair, the attacker may do it
Summing Up / Future Issues

- Cost of univariate resistance
  - security-performance tradeoff
  - processing the shares consecutively

- a light at the end of the tunnel by [pure] masking in hardware?
  - slowly reaching the software performance?
    - making a processor by giant hardware?
  - relatively easy ways to combine the leakages
    - measurement setup & high clock freq.

- What to do when evaluating a countermeasure / product?
  - without any addition/modification on measurement setup?
    - not fair, the attacker may do it
  - with any sophisticated measurement setup?
    - not fair, its security relies on a univariate leak-free scheme
Thanks!

Any questions?

amir.moradi@rub.de

Embedded Security Group, Ruhr University Bochum, Germany