Side-Channel Attacks in the Presence of Countermeasures

Stefan Mangard
Chip Card, Security Innovation Group
Infineon Technologies, Munich, Germany
Email: Stefan.Mangard@infineon.com

Workshop on Provable Security against Physical Attacks
February 17th, Leiden, The Netherlands
Outline

- Types of side-channel countermeasures
- Implementation countermeasures that hide sensitive data
- Implementation countermeasures that mask sensitive data
- Conclusions
Many publications about countermeasures and many more about attacks (including many attacks on countermeasures)
Countermeasures Against Side Channel Attacks

There are essentially two options to counteract side-channel attacks:

- **Protocol countermeasures**
  - Keep the implementation
  - Change the protocol such that the attacker cannot obtain enough leakage in practice
    - Frequent key updates
    - Leakage resilient cryptography

- **Implementation countermeasures**
  - Keep the protocol
  - Change the implementation of the protocol such that there is no practically exploitable leakage any more
    - Masking
    - Dual-rail
    - ...
Questions

- Can we solve the problem at the protocol level without thinking about implementation countermeasures?
  
  (Consider: With templates a completely unprotected AES implementation can potentially be broken with a single trace)

- Can we solve the problem completely at the implementation level without thinking about the protocol?
  
  (Consider: An attacker with unlimited access to the device can detect even the smallest side-channel information)
Opinion

- We need to find the right mix of protocol and implementation countermeasures that leads to the best security/cost ratio for each application
  - There are applications where it is easier to change the protocol
  - There are applications where higher implementation costs for SCA-resistance are acceptable

- Protocols, standards, and infrastructure change slowly (there is still a significant part of the security market that uses Triple-DES)

→ In summary, practical countermeasures for implementations are needed and will be needed
Constraints and Requirements for Implementations

- Security is important, but security is not the only important property of a product
  - Performance
    - Do you want to wait for seconds when doing an authentication?
  - Area
    - Are you willing to pay something extra on your transport ticket?
  - Power
    - Would you accept that your contactless card only works at distance zero?

- Consequence
  - It is necessary to squeeze the maximum out of the system and to balance ALL requirements
The Challenge of Building a Secure Implementation

- The challenge is **NOT** to build a secure system!

- The challenge is to build a secure system with acceptable costs!

Comparison with AES contest

- The challenge was **NOT** to build a secure block cipher

- The challenge was to find a block cipher that had the best trade-off between security, performance, area requirements, ...
Addressing the Challenge

- During the last years numerous publications of countermeasures appeared

- These countermeasures are based on certain assumptions requirements on the implementation

- Often these assumptions are not stated explicitly
  - If an attack is published it is often not immediately clear whether the attack was possible because an assumption was not fulfilled or because there was a general problem with the countermeasure

- Sometimes assumptions are stated explicitly
  - Yet the costs for fulfilling the assumptions in practice is not clear
Types of Implementation Countermeasures

- Intermediate Values
- Power Consumption

**Hiding**

- Intermediate Values
- Power Consumption

**Masking**

- Intermediate Values
- Power Consumption
  - Randomized Intermediate Values
Countermeasures – Part I

“Hiding”
Ad-Hoc Countermeasures: Shuffling and Noise

- Basic idea
  - Hide the information by noise
  - Hide the information by processing the sensitive data at random moments of time

- Properties
  - Makes the life of the attacker more difficult, but does not prevent an attack
  - Based on statistics it can be shown how much the complexity increases by noise and by non-deterministic processing of the sensitive data

- Implementation challenges
  - Prevent attacker from removing non-determinism by signal processing
  - Prevent attacker from bypassing noise (e.g. EM instead of power)
Dual-Rail Pre-Charged (DRP) Circuits

- **Basic Idea**
  - Represent data signals by two complementary wires
  - Each pair of wires is balanced
  - Each pair of pull-down/pull-up networks is balanced
  - Hence the same amount of power is necessary to switch the two wires

- **Examples of Dual-rail pre-charged logic styles**
  - SABL
  - WDDL
  - ... 

- **Implementation challenges**
  - Balancing, early propagation
Building two things that have exactly the same physical properties is a challenging task (not only in semiconductors).

The most important challenges of balancing wires:

- Process variations
- Coupling
  - There is a capacitive coupling between the wires in a chip
  - So not only the two wires should be identical, but also their neighborhood (shielding)
It is not only important that the power consumption of complementary paths is equal. It is also critical when they switch.

The standard way to build logical gates is to build them such that they switch their output as soon as possible. This leads to data-dependent moments of switching!

Assume that the pair a always arrives before the pair b:

- If a is 0 the gate switches early
- If a is 1 the gate switches always later

The better the balancing is and the more fixed the moment of the switching of a gate is, the more costly the circuit becomes!
Countermeasures – Part II
“Masking”
**Masking**

- Basic idea
  
  Make the power consumption independent of sensitive data values by randomizing the sensitive data values

- Example: masking of a data value $d$ with a random mask $m$
  
  $$d_m = d \oplus m_d$$

  $d$ and $m_d$ are pairwise independent of $d_m$; Hence, only power consumption that depend on $d$ and $m_d$ can leak information about $d$
Effectiveness of Masking

- The information leakage per trace of a masked implementation is not fundamentally different than the one of a trace without masking
  - Each trace contains information about the masked value and the mask(s)

- In case of masking there are multiple shares for each data value and the fundamental assumption is that it is hard in practice to exploit the joint leakage of these shares

- This fundamental assumption is challenged in the following scenarios:
  - Glitches
  - Early propagation
  - Architectural leakage
  - Zero-value DPA
  - Biasing attacks
  - Higher-order DPA
**Glitches**

- Standard logical gates switch update their output whenever an input changes → they switch several times per clock cycle
- A circuit of logic gates leaks about all kinds of combinations of its inputs
- This is critical for masked circuits!
Glitches - Example

- Multiplication of additively masked operands
  - Input: $a_m, b_m, m_a, m_b, m_q$
  - Output: $q_m = a*b \oplus m_q$

- Straightforward implementation
  - $i_1 = a_m * b_m = (a \oplus m_a) * (b \oplus m_b) = a*b \oplus a*m_b \oplus b*m_a \oplus m_a*m_b$
  - $i_2 = a_m * m_b = (a \oplus m_a) * m_b = a*m_b \oplus m_a*m_b$
  - $i_3 = m_a * b_m = m_a * (b \oplus m_b) = m_a*b \oplus m_a*m_b$
  - $i_4 = m_a * m_b$
  - $q_m = i_1 \oplus i_2 \oplus i_3 \oplus i_4 \oplus m_q = a*b \oplus m_q$

- In a standard CMOS circuit, this implementation is not secure!
- Many more intermediate values than $i_1 ... i_4$ will occur in the circuit
Early Propagation

- Assume that there is no leakage due to glitches.

- Just like in the case of DRP circuits, the circuit can still be leakage due to early propagation.

Assume that the pair $a$ always arrives before the pair $b$:

- If $a$ is 0 the gate switches early.
- If $a$ is 1 the gate switches always later.
Architectural Leakage

- The power consumption of a circuit depends on its current state and on its former state.

- The same register (or wire) should never store the mask followed by the masked value (or vice versa).
  - Example with HW leakage: sequential storage of m, d_m
    - the power consumption is proportional to HD(m, d_m) = HW(m ⊕ d_m)

- Pitfalls
  - There are many registers in a design that are not visible
  - What happens, if there is a high priority interrupt?
  - ...


Biasing Attacks

- For masking it is necessary that the masks are uniformly distributed

- An attacker who finds a location in the power trace where the mask is processed can:
  - generate subsets of traces according to the power consumption
  - hence the distribution of the mask in these subsets is not uniform any more
  - hence a standard DPA attack can be performed

- Implementation challenge:
  - The attacker must not be able to identify locations in the power traces that depend on the processing of the mask
Zero-Offset DPA

\[ d = d_m \oplus m_d \]

\[ P(d_m, m_d) = P(d_m) + P(m_d) \]

- The average power consumption is equal for all possible values of \( d \) for all power consumption functions \( P() \).

- The statistical distribution of the power consumption is not equal for all values of \( a \), i.e. it is not independent of \( d \)!

- Dependency can be exploited by using other distinguishers or non-linear preprocessing (e.g. squaring) of the power consumption. \( \text{pre}(P(a_m, m_a)) \) can be used to make the average of \( P(a_m, m_a) \) depend on \( d \).
Zero-Offset DPA - Example

- Assume the leakage function is the identity function $P(d) = d$

- Assume we process a masked value $d$ and the mask $d_m$ in parallel

- Assume the power consumption is additive

→ In this scenario the variance of the power consumption leaks information about $d$

<table>
<thead>
<tr>
<th></th>
<th>$d$</th>
<th>$d_m$</th>
<th>$m$</th>
<th>$d_m+m_d$</th>
<th>$(d_m+m_d)^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Higher-Order DPA

- The attacker combines two points in the power trace that correspond to \( P(d_m) \) and \( P(m) \) or to two points that correspond to data values that use the same mask.

- Pre-processing functions are product combining, absolute difference, ...

- Alternatively the leakage of \( P(d_m) \) and \( P(m) \) is exploited directly by estimating the mutual information of \( P(d_m) \), \( P(m) \) and \( d \).

- Implementation challenge
  - Prevent that the attacker can easily find two points in time that correspond to \( P(d_m) \) and \( P(m) \).
Distinguishers and Distributions

- If two points of leakage are combined, the resulting distribution is not Gaussian any more.

- Different combination functions and distinguishers can be used.

- Generic calculation of mutual information does not make any assumption on distribution, but the estimation of mutual information typically converges more slowly than the estimation of other distinguishers.
Example: Template attack on a masked AES software implementation

- Masked AES software implementation on an 8 bit processor

- The attacker has the opportunity to build templates

- The key was revealed with approx. 10 measurements

- Is it possible to reveal the entire key of a masked AES software implementation on an 8 bit processor with a single trace?
Summary on Masking

The goal of masking is not to remove the leakage from a power trace → all we can achieve is to increase the complexity of an attack.

There are many pitfalls

- Special care needs to be taken that the implementation does not have a first-order leakage
  - Glitches, early propagation, architectural leakage, ...

- Special care needs to be taken that the implementation does not have an easy exploitable second-order leakage
  - In particular, the attacker must not be able to easily identify the time location when masks and masked data are processed
In order to proof security, we need to rely on certain assumptions.

Assumptions about implementations are very expensive to fulfill 100%.

To fulfill side-channel related-assumptions of an implementation, significant effort needs to be spent during implementation and during verification.

This holds for all the assumptions that are commonly made:
- only computation leaks
- the device only leaks HW or HD of some data
- all pairs of wires are balanced
- there is no early-propagation
- it is not possible to mount efficient second-order DPA attacks
- ...

Assumptions in Cryptography vs. Assumptions on Implementations

- In cryptography we rely on widely accepted assumptions that are based on fundamental mathematical problems.

- For implementations it is not so clear which assumptions are the best to rely on.
  - Important: every implementation is different and it is necessary to fulfill the assumption for each implementation.
  - We need implementation rules and techniques that lead to the fulfillment of the assumptions we need to prove security.
  - Currently, heuristics are used.
Conclusions

- Security against side-channel attacks is not something black or white; we currently do not have the assumptions that we can rely on 100%.

- Security against side-channel attacks is a question of complexity.

- Currently, heuristics are used to push the complexity of attacks beyond practical limits.

- Provability is of course desirable – however, only at reasonable costs ;-)!

Always the product will be the winner that addresses ALL the requirements (security, area, power, performance, ...) best.
Stefan Mangard
Chip Card, Security Innovation Group
Infineon Technologies, Munich, Germany
Email: Stefan.Mangard@infineon.com