Efficient and Provably Secure Methods for Switching from Arithmetic to Boolean Masking

Blandine Debraize

Leuven, September 10th, 2012

Differential Power Analysis

- In 1999, Paul Kocher introduced the concept of Differential Power Analysis (DPA) [KJJ99].
- His idea is to analyse the power consumption of the device during its execution to recover secret information.
- DPA was extended to some other techniques :
  - Correlation Power Analysis (CPA)
  - ElectroMagnetic Analysis (EMA)...

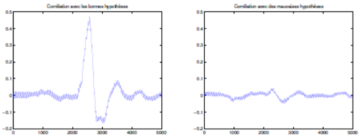

FIG.: Differential Power Analysis result when hypothesis are correct (left) or incorrect (right)

DPA principle

- Guess some key bits.
- Record several curves corresponding to different inputs.
- Average the curves in a way depending on the initial guess.
- The behavior of the averaged curves confirms or not the initial guess.

Algorithmic protections are frequently used to thwart these attacks.

Algorithmic Countermeasures

Principle

- Split all key-dependant intermediate variable processed during execution into several shares [CJRR99, GP99].
- The value of each share, considered independently from the other ones is:
  - randomly distributed,
  - independent of the value of the secret key.

  $\longrightarrow$ The power leakage of one share does not reveal any information.
- When only two shares are used, the method comes to masking all intermediate data with random.

  $\longrightarrow$ The implementation is said to be protected against first order DPA.

Protection of Boolean and arithmetic instructions

- *Boolean masking*: $x' = x \oplus r$
- *Arithmetic masking*: $x' = x - r \bmod 2^K$
- For algorithms that combine both instruction types, the conversion algorithms from one masking to another must also be secure against DPA.

  $\longrightarrow$ Software oriented finalists of the eSTREAM stream cipher competition

  $\longrightarrow$ Stream ciphers Snow 2.0, Snow 3G, block cipher IDEA

  $\longrightarrow$ Hash function designs of SHA family used for HMAC constructions.

Known Conversion Methods

Condition :

- All intermediate variables of the conversion algorithm must be independent of the secret data.

Boolean to arithmetic

1. Efficient method proposed by Louis Goubin [Gou01].
   $\longrightarrow$ Rely on the fact that $f_{x'}(r) = (x' \oplus r) - r$ is affine in r over GF(2).

Arithmetic to Boolean

1. Method also proposed by Goubin in [Gou01], based on the following recursion formula:

$$(A + r) \oplus r = u_{K-1}, \text{ where: } \begin{cases} u_0 = 0, \\ \forall k \geq 0, u_{k+1} = 2[u_k \wedge (A \oplus r) \oplus (A \wedge r)]. \end{cases}$$

   $\longrightarrow$ less efficient than from Boolean to arithmetic, as the number of operation is linear in the size of the intermediate data.

2. Method proposed by Jean-Sébastien Coron and Alexei Tchulkine in [CT03].
   $\longrightarrow$ Based on the use of precomputed tables.
   $\longrightarrow$ Faster than Goubin's method.

3. Method proposed by Olaf Neiße and Jürgen Pulkus in [NP04].
   $\longrightarrow$ Extension of Coron-Tchulkine method.
   $\longrightarrow$ Compared to Coron-Tchulkine, reduction of RAM consumption.

Principle of Coron-Tchulkine method

Principle :

- Two tables G and C are generated during precomputation phase.
- Both tables have size 2^k, where k is the size of the processed data
  $\longrightarrow$ For example if k = 4, a 32-bit variable is divided into 8, 4-bit nibbles: the algorithm works then in 8 steps.

The table G converts a nibble from arithmetic to Boolean masking:

Table G generation

1. Generate a random k-bit r
2. For $A = 0$ to $2^k - 1$ do
   $G[A] = (A + r) \oplus r$
3. Output G and r.

The table C manages carries coming from the modular addition.

Carry table C generation

Input : a random r of k bits.

1. Generate a random k-bit γ
2. For $A = 0$ to $2^k - 1$ do
   $$\text{C[A]} \leftarrow \begin{cases} \gamma, \text{ if } A + r < 2^k \\ \gamma + 1 \bmod 2^k, \text{ if } A + r \geq 2^k \end{cases}$$
3. Output C and γ.

Principle of Coron-Tchulkine method : carry management

Table G generation
1. Generate a random k-bit r
2. For $A = 0$ to $2^k - 1$ do
$G[A] = (A + r) \oplus r$
3. Output G and r.

Carry table C generation
Input : a random r of k bits.
1. Generate a random k-bit γ
2. For $A = 0$ to $2^k - 1$ do
$\mathrm{C[A]} \leftarrow \begin{cases} \gamma, \text{ if } A + r < 2^k \\ \gamma + 1 \bmod 2^k, \text{ if } A + r \geq 2^k \end{cases}$
3. Output C and γ.

- Let us consider x' splitted into n nibbles $x'_{n-1}||\ldots||x'_i||\ldots||x'_0$:
  $\longrightarrow$ each value $x_i = x'_i + r$ can be possibly more than 2^k.
  $\longrightarrow$ the carry must be added to the nibble x'_{i+1} before its conversion.
  $\longrightarrow$ As the carry value is not decorrelated from the secret data, it must be masked.
  $\longrightarrow$ The table C outputs the carry value c of x'_i masked by the addition of a random k-bit value γ.

Principle of Coron-Tchulkine method : conversion step

Conversion algorithm :

Conversion of a $(n \cdot k)$-bit variable
Input : (A,R) such that $x = A + R \bmod 2^{n \cdot k}$ and r, γ generated during precomputation phase
1. For $i = 0$ to $n-1$ do
2. Split A into $A_h \| A_l$ and R into $R_h \| R_l$ such that A_l and R_l have size k
3. $A \leftarrow A - r \bmod 2^{(n-i) \cdot k}$
4. $A \leftarrow A + R_l \bmod 2^{(n-i) \cdot k}$
5. if $i < n-1$ do
6. $A_h \leftarrow A_h + C[A_l] \bmod 2^{(n-i-1) \cdot k}$
7. $A_h \leftarrow A_h - \gamma \bmod 2^{(n-i-1) \cdot k}$
8. $x'_i \leftarrow G[A_l] \oplus R_l$
9. $x'_i \leftarrow x'_i \oplus r$
10. $A \leftarrow A_h$ and $R \leftarrow R_h$
11. Output $x' = x'_{n-1} \| \ldots \| x'_i \| \ldots \| x'_0$

Correctness of Coron-Tchulkine method

If $n > 2$, the Coron-Tchulkine method is not correct :

When:

- γ takes the value $2^k - 1$,
- The carry arising from the addition of the nibble A_l and r equals 1.

Then the output of the table $C[A_l]$ is not the expected value.

Immediate corrections are not first order DPA resistant

- When γ has size k, the output of Table C is not decorrelated from the value of the carry.
- γ must have size $n \times k$.

Neiße-Pulkus method

Extension of Coron-Tchulkine method

- Same 2^k-entry Table G as C.-T. method, used to convert nibble from arithmetic to Boolean masking.
- Contrary to C.-T. method, the carry is here stored unmasked in the 2^k-entry table.

The carry is masked during conversion step

- By the fact that sometimes the direct value of the intermediate variable is processed by conversion step and sometimes its complement is processed, depending on the value of a random bit z.

Security: possible vulnerability with combined SPA-DPA

- The value Z is manipulated several times during one conversion, this value is either `0` or `0xFF...FF`.
- It could be distinguished by the attacker in some context, using SPA techniques.
- With this information, the attacker could mount a DPA attack, using the fact that the carries are then unmasked.

$\longrightarrow$ The behavior of the component in terms of power and electromagnetic leakage must be studied very carefully before choosing this conversion method.

Using only one precomputed table

Both the information provided by Table G of Coron-Tchulkine method (update of the nibble in the new masking mode) and the information of Table C (additively masked carry) can be summarized in one unique table T:

Table T generation	
1.	Generate a random k-bit r and a random $(n \cdot k)$-bit γ
2.	For $A = 0$ to $2^k - 1$ do
	$T[A] = ((A + r) \oplus r) + \gamma \bmod 2^{n \cdot k}$
3.	Output T, r and γ

$\longrightarrow$ If the value $A + r$ is greater than 2^k during the precomputation of T, the $(k+1)^{\text{th}}$ least significant bit of $T[A]$ is automatically set to 1 before being masked by the addition of γ.

$\longrightarrow$ Here the random value γ has the same size as the processed data ($n \cdot k$ bits), thus T's outputs have no dependance on the value of the carries.

Using only one precomputed table

During the conversion algorithm, the carry is added to the current variable at the same time as the nibble A_l is updated (line 5):

Conversion of a $(n \cdot k)$-bit variable

Input : (A,R) such that $x = A + R \bmod 2^{n \cdot k}$
and r, γ generated during precomputation phase

1. For $i = 0$ to $n - 1$ do
2. Split A into $A_h||A_l$ and R into $R_h||R_l$, such that A_l and R_l have size k
3. $A \leftarrow A - r \bmod 2^{(n-i) \cdot k}$
4. $A \leftarrow A + R_l \bmod 2^{(n-i) \cdot k}$
5. $A \leftarrow A_h||0 + T[A_l] \bmod 2^{n \cdot k}$
6. $A \leftarrow A - \gamma \bmod 2^{n \cdot k}$
7. $x'_i \leftarrow A_l \oplus R_l$
8. $x'_i \leftarrow A_l \oplus r$
9. $A \leftarrow A_h$ and $R \leftarrow R_h$
10. Output $x' = x'_0||\ldots||x'_i||\ldots||x'_{n-1}$

This method allows both to correct and to improve time performance of Coron-Tchulkine method.

First idea

Idea:

- Blind the carry with a Boolean mask.
- Use a precomputed table to keep the carry masked during the algorithm execution.

Remark

To be first order DPA resistant, such lookup table must be such that:

- The input of the table is masked, and then treated during conversion step as a memory address information.
- The output of the table is masked.

New Algorithm

To obtain time performance:

- Combine the information about the update of the current nibble and of the masked carry bit with one unique table T:
  - $\longrightarrow$ In the input of the table
  - $\longrightarrow$ And in the output of the table.
- During conversion phase, the choice of the address in T not only depends on the value of the nibble but also on the value of the masked previous carry.
  - $\longrightarrow$ T has size 2^{k+1}.
- The output of T is directly the value $(A + r + c) \oplus r$, where c is the carry resulting from the previous addition.

New Algorithm

Table T generation

1. Generate a random k-bit r and a random bit ρ
2. For $A = 0$ to $2^k - 1$ do
   $T[\rho||A] = (A + r) \oplus (\rho||r)$
   $T[(\rho \oplus 1)||A] = (A + r + 1) \oplus (\rho||r)$
3. Output T, r and ρ

Conversion of a $n \cdot k$-bit variable

Input: (A,R) such that $x = A + R \bmod 2^{n \cdot k}$,
r, ρ generated during precomputation phase

1. $A \leftarrow A - (r||\ldots||r||\ldots||r) \bmod 2^{n \cdot k}$
2. $\beta \leftarrow \rho$
3. For $i = 0$ to $n - 1$ do
4. Split A into $A_h||A_l$ and R into $R_h||R_l$, such that A_l and R_l have size k.
5. $A \leftarrow A + R_l \bmod 2^{(n-i) \cdot k}$
6. $\beta||x'_i \leftarrow T[\beta||A_l]$
7. $x'_i \leftarrow x'_i \oplus R_l$
8. $A \leftarrow A_h$ and $R \leftarrow R_h$
9. Output $x' = (x'_0||\ldots||x'_i||\ldots||x'_{n-1}) \oplus (r||\ldots||r||\ldots||r)$

Implementation choices

Generic choices

- The versions chosen for the tests are the ones that are optimized in terms of time performance. A special optimized version of the Neiße-Pulkus method was implemented for the tests (Appendix C.1 and C.2 in the paper).
- The size of the data to be converted from arithmetic to Boolean is 32 bits (most common size for intermediate data of cryptographic algorithms).
- Two nibble size were tested: $k = 4$ and $k = 8$.
- Tested on 8-bit, 16-bit and 32-bit architectures.

For 8-bit and 16-bit architectures:

- We performed C implementations.
- The results are given in clock cycles number, computed with the help of a simulation tool.

8-bit and 16-bit architectures

TAB.: Smart card 8-bit microprocessor

	Goubin's method	Mod. N.-P.		Imp. C.-T.		New method	
		$k=4$	$k=8$	$k=4$	$k=8$	$k=4$	$k=8$
Precomputation time	10325	2562	40274	18589	109391	3166	93007
Conversion time	39213	15479	9208	13969	7060	11720	6111
Table size	0	16	512	64	1024	32	1024

TAB.: Smart card 16-bit microprocessor

	Goubin's method	Mod. N.-P.		Imp. C.-T.		New method	
		$k=4$	$k=8$	$k=4$	$k=8$	$k=4$	$k=8$
Precomputation time	86	377	3734	921	5933	439	5174
Conversion time	934	558	308	512	274	445	257
Table size	0	16	512	64	1024	32	1024

Implementation choices

- We performed performance comparison tests in ARM assembler on a 32-bit 26 MHz microprocessor.
- The time results are given in microseconds.

TAB.: Smart card 32-bit microprocessor

	Goubin's method	Mod. N.-P.		Mod. C.-T.		New method	
		$k=4$	$k=8$	$k=4$	$k=8$	$k=4$	$k=8$
Precomputation time	15.1	9.6	156.2	25.5	188.8	12.1	180.3
Conversion time	32.9	12.9	10.3	12.1	8	14.9	9.2
Table size	0	16	512	64	1024	32	1024

Conclusion

In this paper we investigated the fastest methods for switching from arithmetic to Boolean masking.

- First we analyzed two known methods [CT03, NP04] based on precomputed lookup tables:
    - We showed that the algorithm proposed in [CT03] is not correct and proposed an improved correction.
- We also proposed a new method that is:
    - Well adapted for 8-bit architecture
    - As the correction of [CT03], offers better security against side channel analysis in practice than the algorithm proposed in [NP04].

Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi.
Towards sound approaches to counteract power-analysis attacks.
In *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

Jean-Sébastien Coron and Alexei Tchulkine.
A new algorithm for switching from arithmetic to boolean masking.
In *CHES*, volume 2779 of *Lecture Notes in Computer Science*, pages 89–97. Springer, 2003.

Louis Goubin.
A sound method for switching between boolean and arithmetic masking.
In *CHES*, volume 2162 of *Lecture Notes in Computer Science*, pages 3–15. Springer, 2001.

Louis Goubin and Jacques Patarin.
Des and differential power analysis (the "duplication" method).
In *CHES*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.

Paul C. Kocher, Joshua Jaffe, and Benjamin Jun.
Differential power analysis.
In *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

Olaf Neiße and Jürgen Pulkus.
Switching blindings with a view towards idea.

In *CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 230–239. Springer, 2004.