Side-Channel Based Reverse Engineering of Secret Algorithms

Roman Novak
Jozef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
roman.novak@ijs.si

Abstract

[1] *Two techniques are introduced that enable side-channel based reverse engineering of secret algorithms. The first is sign-extended differential power analysis (SDPA) while the second technique targets table lookups. The SDPA reveals values that collide with the DPA target value within the circuitry. The interpretation of those values can provide significant amounts of the information about the algorithm. The attack on substitution blocks may reveal contents of lookup tables. It is based on identifying equal intermediate results from power measurements. The techniques have been successfully tested in a demonstration attack on a secret authentication and session key generation algorithm implemented on SIM cards in GSM networks.*

1 Introduction

Any real cryptographic device provides more information to a determined adversary than just the input plaintext and output ciphertext. This side-channel information is available as the timing of operations [1], power consumption of the devices [2], electromagnetic emanations [3], etc. Very little side-channel information is required to break many common ciphers. Non-invasive attacks and their accompanying countermeasures have been studied extensively over the past few years. Systems that rely on smartcards are of particular concern.

Most of the side-channel based methods deal with the extraction of cryptographic material, such as keys, from the implementations of well-known algorithms. However, many cryptographic algorithms are still kept secret. For instance, GSM network operators use an updated version of COMP128-1, designated as COMP128-2, as an authentication algorithm, but the algorithm remains unpublished. Some network operators even develop a proprietary algorithm in secrecy. The purpose of this paper is to show that secret algorithms offer very little protection.

[1]ERK 2003, pp. 445-448, 2003.

Section 2 gives a short introduction to power analysis techniques. In Sect. 3 a sign extension of Differential Power Analysis (SDPA) is described. The leaked information that can be captured by the SDPA is detailed and an example on an unknown GSM authentication algorithm is given. The algorithm is implemented on SIM smart cards. Usually, supplementary methods have to be employed to completely restore the algorithm. A technique that may reveal substitution blocks is introduced in Sect. 4.

2 Power Analysis

Smart cards consist of logic gates, which are basically interconnected transistors. During operation, charges are applied to or removed from transistor gates. The sum of all charges can be measured through power consumption, on which power analysis techniques are based.

Several variations of power analysis have been developed [2, 5]. The power consumption measurements of smart card operations are interpreted directly in Simple Power Analysis (SPA). SPA can reveal hidden data in algorithms in which the execution path depends on the data being processed. More advanced techniques, like Differential Power Analysis (DPA) and Inferential Power Analysis (IPA), allow observation of the effects correlated to the data values being manipulated.

Power analysis attacks have been known for a while and effective countermeasures exist that pose difficulties, even to a well-funded and knowledgeable adversary [6, 7, 4]. However, many solution providers are still convinced that implementation of a secret algorithm provides sufficient level of protection against low-cost side-channel attacks. The results presented here speak against attempts to establish secrecy by keeping cryptographic algorithms undisclosed.

3 Sign-Extended DPA

A DPA attack begins by running the encryption algorithm for N random values of plaintext input. For each of the inputs, a discrete time power signal is collected. The power signal is a sampled version of the power being consumed during the portion of the algorithm that is under attack. The power signals are split into two sets using a partitioning function, which is selected in a way to differentiate signals based on predicted target value of a chosen variable. Usually one bit of the target value is considered. The next step is to compute the average power signal for each set. By subtracting the two averages, a discrete time DPA bias signal is obtained. Selecting an appropriate partitioning function results in a DPA bias signal that can be used to verify guessed portions of the secret key. If enough samples are used and a correct guess as to the value of the intermediate variable has been made, the DPA bias signal will show power biases at some points in time and will converge to 0 all other times.

Basically DPA is used as a tool for testing various hypotheses about secret values in cryptographic algorithms [2]. On the other hand, Sign-Extended DPA (SDPA) is performed on known variables within an algorithm in order to deduce further knowledge about the algorithm. The Sign-Extended DPA method take into account signs of power biases [8]. It was shown that the sign of power bias carries information about the initial or final state of the circuitry that stores the DPA target value. In case of SDPA the target value is usually one bit of known variable within the algorithm. By performing sign analysis of power biases for all bits of known variable, values can be revealed that collide with the observed value in the same circuitry. An SDPA vector is defined that combine signs for all bits of a target value. Note that some vector components may not be defined as power biases for some bits may be missing.

3.1 Leaked Information

It is difficult to deduce the meaning of SDPA vectors, as there can be many explanations. For instance, a constant may be a real constant used by the algorithm, but it can also be the memory address of a variable, value of a variable, opcode, or just the effect of a precharged bus. Other more complex explanations are possible.

Moreover, the collision of the observed value with its transformed counterpart produce very distinct SDPA vectors, which can be considered as a signature of the transformation. First results based on simulation suggest that various arithmetic and logic operations may be identified.

For example, suppose that input value i collides with its shifted counterpart, $i \gg 3$. Then the following SDPA vector $\mathbf{s}_{j'}$

$$\mathbf{s}_{j'} = [0, 0, 0, \sim, \sim, \sim, \sim, \sim] \tag{1}$$

is expected, where the number of consecutive zeros equals the number of places shifted to the right and $\sim$ designates undefined sign.

Note that the same signature would be obtained if zero collided with masked input. Signatures for other typical logical and arithmetic operations may be derived. They are very dependent on the value with which the transformed value collides. Similar signatures may result from different transformations, while some transformations cannot be identified in this way. For instance, when the rotated value collides with its original, the DPA power bias limits to zero. Furthermore, many signatures, like those for addition and multiplication, have an effect on the size of a bias and not only on its sign. When the transition of some bits leak more information than others, i.e. power bias is dependent on bit position, the interpretation of magnitudes in addition to signs may appear impossible. However, the experimental results in [4] show that, on busses, the magnitude of the voltage pulse is directly proportional to the number of bits that changed. Further research is required in order to include the information about magnitudes into signatures.

Many ciphers implement complex compositions of simple operations through different levels of iterated code. In order to correctly interpret SDPA vectors, the investigation of similar vectors from multiple iterations can be very helpful. A SDPA matrix captures sign information for cross-iteration comparisons. The matrix is a collection of similar SDPA vectors that are observed within each iteration, i.e. vectors evenly spaced in time.

3.2 Example

We have applied SDPA to an unknown GSM authentication algorithm that can be found on SIM cards. The algorithm is a keyed hash function. It takes a 16-byte key (128 bits) and 16-byte of data (128 bits) to output a 12-byte (96 bits) hash. The key $k_0 - k_{15}$, as used in the GSM protocol, is unique to each subscriber and is stored in the SIM card. The input data $i_0 - i_{15}$ is a random challenge supplied by the base station. The first 32 bits of the hash are used as a response to the challenge and sent back to the base station. The remaining 64 bits are used as a session key for voice encryption using the A5 algorithm.

The following SDPA matrices are identified at the beginning of the algorithm:

$$\begin{gathered}
\mathbf{C}_0 = [0, 0, \ldots, 0, 0] \\
\mathbf{CNT}_{0-15} = [0, 1, \ldots, 14, 15] \\
\mathbf{KEY} = [76, 157, \ldots, 31, 224] \\
\mathbf{CNT}_{0-15} \oplus \mathbf{KEY} = [76, 156, \ldots, 17, 239] \\
\mathbf{CNT}_{69-84} \oplus \mathbf{KEY} = [9, 219, \ldots, 76, 180] \ ,
\end{gathered} \tag{2}$$

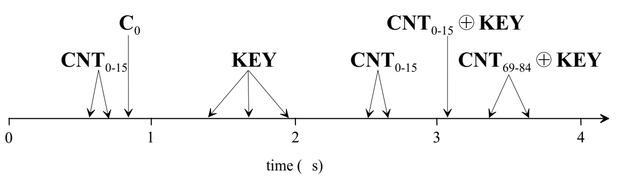
$\mathbf{C}_0$
$\mathbf{CNT}_{0\text{-}15}$
$\mathbf{KEY}$
$\mathbf{CNT}_{0\text{-}15}$
$\mathbf{CNT}_{0\text{-}15} \oplus \mathbf{KEY}$
$\mathbf{CNT}_{69\text{-}84} \oplus \mathbf{KEY}$
0
1
2
3
4
time (s)

Figure 1: Temporal ordering of the SDPA matrices within the first iterations

where the matrix columns are written as bit values of SDPA vectors. 8-bit blocks of plaintext input $i_0 - i_{15}$ were selected as known intermediate variables, one per iteration. In Fig. 1 a temporal ordering of the matrices within the algorithm's first iterations is shown.

When only one iteration is observed, the SDPA values are just different constants that collided with the input. The matrix representation gives more information. For instance, $\mathbf{CNT}_{0-15}$ can be interpreted as a loop counter in collision with the plaintext input. This explanation is very likely as both input and counter are handled within a typical loop. In the middle of the iteration a SDPA matrix that appears to contain random SDPA values is identified three times. Our prediction that a key $k_0 - k_{15}$ collided with the input has proven correct, hence the label **KEY**. At the end of the iteration two different matrices were identified. The first can be calculated as a bitwise exclusive or between $\mathbf{CNT}_{0-15}$ and **KEY**, $\mathbf{CNT}_{0-15} \oplus \mathbf{KEY}$, and the second as $\mathbf{CNT}_{69-84} \oplus \mathbf{KEY}$. It is highly unlikely that these values actually collided with the input; however, the same matrices are obtained if $\mathbf{KEY} \oplus i_0 - i_{15}$ collides with $\mathbf{CNT}_{0-15}$ and $\mathbf{CNT}_{69-84}$, respectively. The $\mathbf{CNT}_{69-84}$ may show the memory range where the results of the computation are stored. Other explanations are also possible. From the above conclusions, one can write the Alg. 1 that would cause a similar response to SDPA.

Algorithm 1. Initial computation in unknown algorithm

```
FOR $j$ from 0 to 15
    $x[j] = i[j] \oplus k[j]$
END FOR
```

In the above example, SDPA reveals information about keys, counters, memory ranges, operations and their temporal ordering.

4 Breaking a Substitution Block

As shown by a simple example, SDPA can be used in the reverse engineering attempts of an adversary to reveal secret code. After new intermediate values have been discovered, SDPA is performed on those values instead of on blocks of plaintext input.

Supplementary methods have to be employed to completely restore the algorithm under attack. For instance, simple power analysis (SPA) and the use of correlation techniques may help in identifying the algorithm's loops.

The next major difficulty in restoration attempts are substitution blocks, which are usually implemented as lookup tables. Modern cryptography uses substitution as a building block in complex compositions of strong ciphers. A substitution block is a very effective countermeasure against SDPA as it prevents intermediates from being tracked through the algorithm. However, many implementations of lookup operation are insecure. An attack on substitution blocks is proposed in [9]. The attack is based on identifying equal intermediate results from power measurements while the actual values of these intermediates remain unknown.

The side-channel attack on substitution blocks described below has been validated on the same SIM card as the above SDPA method. The contents of the lookup tables of the implemented authentication and session-key generation algorithm have been completely restored. Additional examples are given in [9].

Let $f(p)$ be a function that incorporates a lookup table T and some further transformations of the value read from the table. The parameter p represents plaintext input and may be extended to a sequence of parameters without significant change of the method.

$$r = f(p) \tag{3}$$

The problem that has to be solved by an adversary is to find the content of the unknown or modified lookup table T just by observing the side-channel information that is present in power variations. The value of parameter p is known to the attacker, while the result r is unknown, since it is further modified during algorithm execution.

The basic idea behind the attack on a substitution block is based on the fact that the same value of the parameter p gives the same intermediate result r, while different values of parameter p do not necessarily give different intermediate results r as soon as f is an injective function. By identifying equal intermediate results one can partially or fully restore the content of the lookup table and thus break an unknown substitution block.

First, identification of the relevant measurements is needed to enable comparison of the results within algorithm execution by side-channel information alone. Various methods are discussed in [9]. An equation can be written for each pair of parameters p_1 and p_2 for which the equality of the results have been detected:

$$f(p_1) = f(p_2),\ p_1 \neq p_2\ . \tag{4}$$

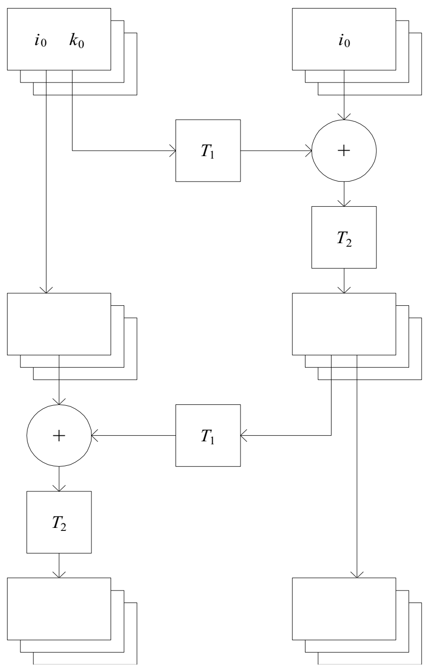

Figure 2: Computation in the GSM authentication algorithm

It is not necessary to find all similarities and, hence, all equations. The set of equations is then solved for the table values with some degree-of-freedom, DF. It is important for DF to be very small, in practice not larger than 2. A large DF makes exhaustive search of the solution space infeasible. The right table can be identified using DPA on the computation results after the lookup operation.

The requirement for small DF may prevent breaking a substitution block when only a single lookup operation is observed. However, the same table is frequently used several times within an algorithm because the same substitution block is used several times to form the product cipher.

We managed to completely restore the unknown GSM authentication algorithm using the above methods. In Fig. 2 a detail of the algorithm is shown that includes four lookup operations. Only a small fraction of the sixteen similar iterations is shown. The code is executed at the beginning of the algorithm and can be restored only by combining SDPA, SPA and substitution block attack.

5 Conclusion

An example of side-channel reverse engineering of secret algorithm has been described. In addition to SPA, a sign extension to DPA and side-channel attack on substitution block have been used. Many authors provide guidance for designing smart card solutions against power analysis attacks; however, they are not always present in real implementations. The designers must not rely on secrecy of the algorithm, as the algorithm may be reverse-engineered in the presence of side-channel information leakage.

References

[1] P. Kocher, Timing Attacks on Implementation of Diffie-Hellman, RSA, DSS and Other Systems. In: Koblitz, N. (ed.): Advances in Cryptology - Crypto'96. *Lecture Notes in Computer Science*, Vol. 1109. Springer-Verlag, Berlin Heidelberg New York, pp. 104–113, 1996.

[2] P. Kocher, J. Jaffe and B. Jun, Differential Power Analysis. In: Wiener, M. (ed.): Advances in Cryptology - Crypto'99. *Lecture Notes in Computer Science*, Vol. 1666. Springer-Verlag, Berlin Heidelberg New York, pp. 388–397, 1999.

[3] D. Agrawal, B. Archambeault, J.R. Rao and P. Rohatgi, The EM Side-Channel(s): Attacks and Assessment Methodologies. In: *Cryptographic Hardware and Embedded Systems - CHES'2002.*

[4] T.S. Messerges, E.A. Dabbish and R.H. Sloan, Examining Smart-Card Security under the Threat of Power Analysis Attacks. *IEEE Transactions on Computers*, 51(5), pp. 541–552, 2002.

[5] P.N. Fahn and P.K. Pearson, IPA: A New Class of Power Attacks. In: Koc, C.K., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems - CHES'99. *Lecture Notes in Computer Science*, Vol. 1717. Springer-Verlag, Berlin Heidelberg New York, pp. 173–186, 1999.

[6] O. Kömmerling and M.G. Kuhn, Design Principles for Tamper-Resistant Smartcard Processors. *Proceedings of the USENIX Workshop on Smartcard Technology - Smartcard'99*, Chicago, Illinois, May 10–11, USENIX Association, pp. 9–20, 1999.

[7] S. Chari, C.S. Jutla, J.R. Rao and P. Rohatgi, Towards Sound Countermeasures to Counteract Power-Analysis Attacks. In: Wiener, M. (ed.): Advances in Cryptology - Crypto'99. *Lecture Notes in Computer Science*, Vol. 1666. Springer-Verlag, Berlin Heidelberg New York, pp. 398–412, 1999.

[8] R. Novak, Sign-Based Differential Power Analysis. submitted for publication, 2003.

[9] R. Novak, Side-Channel Attack on Substitution Blocks. In: 1st MiAn International Conference on Applied Cryptography and Network Security, ACNS2003, Kunming, China, *Lecture Notes in Computer Science*, Springer-Verlag, Berlin Heidelberg New York, in print, 2003.