An AES Smart Card Implementation Resistant to Power Analysis Attacks*

Christoph Herbst, Elisabeth Oswald, and Stefan Mangard

Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology, Inffeldgasse 16a, A–8010 Graz, Austria
{christoph.herbst, elisabeth.oswald, stefan.mangard}@iaik.tugraz.at

Abstract. In this article we describe an efficient AES software implementation that is well suited for 8-bit smart cards and resistant against power analysis attacks. Our implementation masks the intermediate results and randomizes the sequence of operations at the beginning and the end of the AES execution. Because of the masking, it is secure against simple power analysis attacks, template attacks and first-order DPA attacks. Due to the combination of masking and randomization, it is resistant against higher-order DPA attacks. Resistant means that a large number of measurements is required for a successful attack. This expected number of measurements is tunable. The designer can choose the amount of randomization and thereby increase the number of measurements. This article also includes a practical evaluation of the countermeasures. The results prove the theoretical assessment of the countermeasures to be correct.

Keywords: AES, smart card, DPA resistance.

1 Introduction

Embedded processors have a large share in the processor market. Especially 8-bit processors are used in many smart cards. Smart cards play a crucial role in a lot of security systems. Due to the lack of secure PCs, smart cards are often used in order to store secret keys. In addition, smart cards are frequently used as authentication devices. For instance, in many ATM systems, users are authenticated not only via their PIN. In addition, the ATM card (the smart card) of the user authenticates itself to the ATM machine. In both scenarios it is imperative that the secret key never leaves the smart card. Consequently, the smart card not only stores the secret key, it is also capable of doing cryptographic operations with that key.

During the last six years, side-channel attacks in general, and power analysis attacks in particular, have shaken the believe in the security of smart cards.

* The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507270 SCARD and through the Austrian Science Fund (FWF) under grant number P16952.

J. Zhou, M. Yung, and F. Bao (Eds.): ACNS 2006, LNCS 3989, pp. 239–252, 2006.

Kocher showed in his pioneering article [KJJ99] that a smart card that is unprotected against power analysis attacks, can be broken easily. In a power analysis attack, the attacker records the power consumption of a smart card while it performs cryptographic operations with a fixed secret key. This secret key can subsequently be revealed based on the recorded data (the so called traces) and the corresponding plaintexts or ciphertexts. In the best case, such an attack requires no knowledge about the implementation details of the algorithm and no more than 100 traces [KJJ99].

The Advanced Encryption Standard (AES) [Nat01] is the most popular primitive for encryption today. It is a symmetric cipher and can be implemented efficiently on all kinds of platforms. It can also be used for authentication. Hence, it is an attractive algorithm for many security relevant applications. As we have pointed out already, the secure implementation of cryptographic primitives on smart cards is challenging. Nevertheless, implementations of the AES algorithm on smart cards that are resistance against power analysis attacks, are a primary interest of the industry. In addition, they are a challenging task for the research community: a smart card is a rather constraint device. It runs on a low clock frequency and is supposed to have a low power consumption. Furthermore, only a very limited amount of memory (program memory, RAM, ROM, etc.) is available that needs to be shared with the operating system.

In this article, we present an AES implementation that is highly resistant against power analysis attacks and that performs well on 8-bit processors (smart cards). We use a combination of countermeasures (masking and randomization of operations) to achieve resistance against power analysis attacks. A security analysis that includes a theoretical assessment and a practical evaluation accompanies this paper. The innovation in this work is the efficient combination of countermeasures, which is specifically tailored for AES implementations on 8-bit smart cards. This is the first work presenting an efficient implementation that offers resistance against power analysis attacks.

This article is organized as follows. In Sect. 2, we explain how masking and randomization work, how each of them can be attacked and how combining them increases the resistance against power analysis attacks. In Sect. 3, we explain how our masked and randomized AES implementation works. In Sect. 4, we analyze the security of our implementation and provide evidence on the soundness of our analysis by showing results of practical power analysis attacks. We conclude this article in Sect. 5. Throughout this article, we assume that the reader has a basic understanding of the working principle of differential power analysis (DPA) attacks.

2 Countermeasures Against DPA Attacks

In order to secure implementations of symmetric cryptographic algorithms against power analysis attacks, there are two approaches that are suitable for software implementations on smart cards. On the one hand, the intermediate values of the algorithm can be masked. On the other hand, the sequence of operations in the algorithm can be randomized. In this section, we briefly discuss these two methods.

2.1 Masking

In a masked implementation all intermediate values a are concealed by a random value m which is called mask. For every execution of the algorithm, new masks are generated on the smart card. Hence, the attacker does not know the masks. Most masking schemes use additive masking, *i.e.* the mask is exclusive-ored with the intermediate value. Thus, the masked intermediate value is $a_m = a \oplus m$. For AES, also multiplicative masking as been suggested [AG01]. Multiplicative masking means multiplying a mask value with an intermediate value: $b_m = b * m$. This multiplication is a modular multiplication. Hence, it is not suitable for most smart card implementations because a modular multiplier is not available on all smart cards. Consequently, we focus on additive masking schemes.

Masking prevents DPA attacks because the randomly masked intermediate values cause a power consumption that is not predictable by the attacker. The masks are added at the very beginning of the algorithm to the plaintext. During the execution of the algorithm, one needs to take care that every intermediate value stays masked. In addition, one needs to keep track how the masks are modified by the operations in the algorithm. For AES operations like ShiftRows and AddRoundKey this can be done with almost no effort. MixColumns requires some effort because it mixes bytes of different columns of the AES state. For the non-linear SubBytes operation, a more elaborated approach is required. In a typical software implementation the SubBytes operation is implemented as table look-up: $out = S(in)$ (S denotes the SubBytes table). The AES state consists of 16 bytes. Thus, we have to perform 16 table look-up operations. When we mask the SubBytes operation, we have to compute a masked SubBytes table S' such that $S'(a_m) = S'(a \oplus m) = S(a) \oplus m'$. At the very end of the algorithm, the masks are removed from the intermediate values.

Provably secure masking schemes for AES have recently been published in [BGK05] and [OMPR05]. Yet, these schemes have been mainly designed for hardware implementations. Nevertheless, also a first proposal for a software implementation of the scheme proposed in [OMPR05] has recently been published in [OS06]. This proposal is faster than the usual look-up table based scheme, if just one AES block needs to be encrypted using a fresh mask of 16 bytes. If several blocks are encrypted, the classical masking approach for AES (i.e. precomputing and storing masked S-Boxes in RAM) is more efficient. However, in an ideal masking scheme, where each intermediate value is masked with a different random value, one needs to keep track of 16 different masks. This leads to a serious decrease in performance and is unacceptable for most applications. In order to get a masked AES implementation with acceptable performance, tradeoffs between security and speed have to be made. Using fewer masks improves the performance but decreases the security against higher-order DPA attacks. Using only one mask leads to problems with MixColumns. If MixColumns needs to be computed efficiently, different masks for each row of the AES state have to be used. In most practical implementations, a small set of masks is used for all AES rounds. It is imperative for the security of a masked implementation that all intermediate values remain masked at all times.

Attacks on Masking Schemes. Masking schemes protect against fist-order DPA attacks. It is well known that, depending on the implementations, higher-order DPA attacks may succeed. In a higher-order DPA attack, several points of a power trace that correspond to several intermediate results, are combined in the statistical analysis. In particular, in a second-order DPA attack, one uses two intermediate points p_1 and p_2 of a trace that correspond to the processing of two values a_m and b_m. Typically the points are chosen such that they are concealed with the same mask m. Then, it holds that $|p_1 - p_2| \sim HW(a_m \oplus b_m)$. Because $a_m \oplus b_m = a \oplus b$ it is possible to predict the Hamming weight $HW(a_m \oplus b_m)$.

Only recently, the research community has picked up the topic of higher-order DPA attacks again, see [WW04], [SPQ05] and [JPS05]. The paper [OMHT06], that has been published only recently, provides theoretical discussions and practical results for second-order attacks on masked smart card implementation of AES. A conclusion from this paper is that second-order DPA attacks can be performed efficiently in practice with a low number of measurements. This means that masking alone does not lead to practically secure implementations if the masking scheme is supposed to be efficient. However, simply using more masks might not be the solution to the problem. This is because second-order DPA attacks work whenever two intermediate values are concealed by the same masks, or whenever the mask and the masked value occur at two moments in time. At some point in time, the masks have to be created, and at some point later, they are applied to some intermediate value. Hence, there are always two points in time that allow a second-order DPA attack.

As a consequence, it is better to combine a simple and efficient masking scheme with another countermeasure to achieve resistance against higher-order DPA attacks. For instance, the execution of the algorithm can be randomized.

2.2 Randomizing the Execution of the Algorithm

Randomizing the execution of the sequence of operations in an algorithm provides additional resistance against power analysis attacks. The goal of the randomization is to distribute the intermediate cipher operations (and thereby the intermediate values) over a given period of time. The distribution should neither be predictable nor be observable by the attacker.

Due to this distribution, the intermediate value that is used in the attack occurs only with a certain probability at a particular moment in time. Therefore, the correlation between this intermediate value and the power consumption is significantly reduced.

For this randomization approach, the insertion of random dummy operations or wait states has been proposed in the literature. The problem with wait states is that they can be easily identified and removed by analyzing a single power trace. When using random dummy operations, the programmer has to take care that dummy operations can not be distinguished from real operations of the algorithm.

We think that there are two efficient ways to introduce randomness in the execution of an algorithm. Either, one adds additional rounds (or parts of a round) to the encryption algorithm at the beginning and the end, or one randomly chooses the sequence of operations within the algorithm. The first method makes it impossible for the attacker to know when the real first round and the real last round takes place. The latter method provides an additional randomization within each round.

The statistical effects of randomization have been studied in [CCD00] and [Man04] in detail. Both papers come to the same conclusion. If the probability that the intermediate value occurs at a certain time is p, then the correlation coefficient decreases by a factor of p and the number of measurements needed for a successful attack increases by a factor of p^2.

3 A Power Analysis Resistant AES Smart Card Implementation

In our AES software implementation, we apply a combination of the countermeasures that we discussed in Sect. 2. The implementation is optimized for simple 8-bit smart cards. We make the common assumption that a random number generator is available.

All rounds of our implementation are masked. The first round and the last round are embedded in so-called randomization zones. Within a randomization zone, the sequence of masked AES operations is randomized and repeated a certain number of times. The number of repetitions in the first randomization zone defines the number of repetitions in the second randomization zone. The total number of repetitions is specified by the designer and is constant over multiple runs of the algorithm. The overall execution time stays therefore constant.

In principle, the masking scheme and the randomization scheme are designed independently from each other. However, we have changed the sequence of MixColumns and Shiftrows in order to facilitate the randomization. In the following subsections we first describe our masking scheme and afterwards the randomization of this scheme.

3.1 Efficiently Masking AES

In our masking scheme we use six different mask bytes. The first two bytes, M and M' are the input and output masks for the masked SubBytes operation. The remaining four bytes $M1$, $M2$, $M3$, and $M4$ are the input masks of the MixColumns operation. We take care that all intermediate values stay masked at all times.

Masking an AES round. At the start of each AES encryption, two precomputations take place. First we compute a masked SubBytes table S' such that $S'(x \oplus M) = S(x) \oplus M'$. Then we pre-compute the output masks for the MixColumns operation $(M1', M2', M3', M4') = \text{MixColumns}(M1, M2, M3, M4)$.

At the beginning of each round, the plaintext is masked with $M1'$, $M2'$, $M3'$, and $M4'$. Then, the AddRoundKey operation is performed. The round key is also masked (a detailed description is given in the subsequent section). Therefore, the masks change from $M1'$, $M2'$, $M3'$ and $M4'$ to the input mask M of the masked Subbytes table S'. Then, the table look-up with the table S' is performed. This changes the mask to M'. Before MixColumns, we change the mask from M' to $M1$ in the first row, to $M2$ in the second row, to $M3$ in the third row and to $M4$ in the fourth row. At the end of the round, MixColumns is performed which changes the masks Mi to Mi'. ShiftRows has no influence on the masks. At the end of the last encryption round, the masks are removed by the final AddRoundKey operation.

Masking the Key Schedule in Practice. Due to security reasons [Man03] the key schedule is also masked. In order to reuse the masked SubBytes table S', we decided to use the mask bytes M and M' also during calculation of the round keys. Furthermore, by applying the mask values Mi' to the round key bytes, we can save some remasking operations during the encryption round.

In the first step of the key schedule, the original cipherkey is masked. A byte of a word of the round key is masked with a value $Mi' \oplus M$. Figure 1 shows the masking scheme for all AES round keys, except for the one of the last round key. The masking scheme for the last round key is shown in Fig. 2. It differs because we want the last round key to remove the masks in order to obtain the ciphertext.

3.2 Randomizing the Masked AES

As explained in Sect. 2.2, there are two efficient possibilities to randomize the sequence of operations. Either, one adds additional rounds (or parts of a round) to the encryption algorithm at the beginning and the end, or one randomly chooses the sequence of operations within the algorithm.

In AES, several operations can be randomized. For instance, the AddRoundKey operation allows randomization. AddRoundKey adds each byte of the (masked) plaintext to the corresponding byte of the (masked) round key. The sequence of the processing can be randomized, because the 16 bytes of the state are processed independently. The same argument holds for the SubBytes operation. During MixColumns, the sequence of the processing of the columns can be randomized. Within each column, the processing of the rows can be randomized.

We also add parts of a round at the beginning and the end of each AES execution. The so-called dummy rounds work on a dummy state that lies in a different memory area in the smart card. In order to minimize information leakage about which state is used, we use base addresses for the dummy state and the real state that have the same Hamming weight. In Fig. 3, we depict the program flow of a randomized and masked AES encryption. The two randomization areas are called Randomization Zone 1 and Randomization Zone 2. Only in these zones,

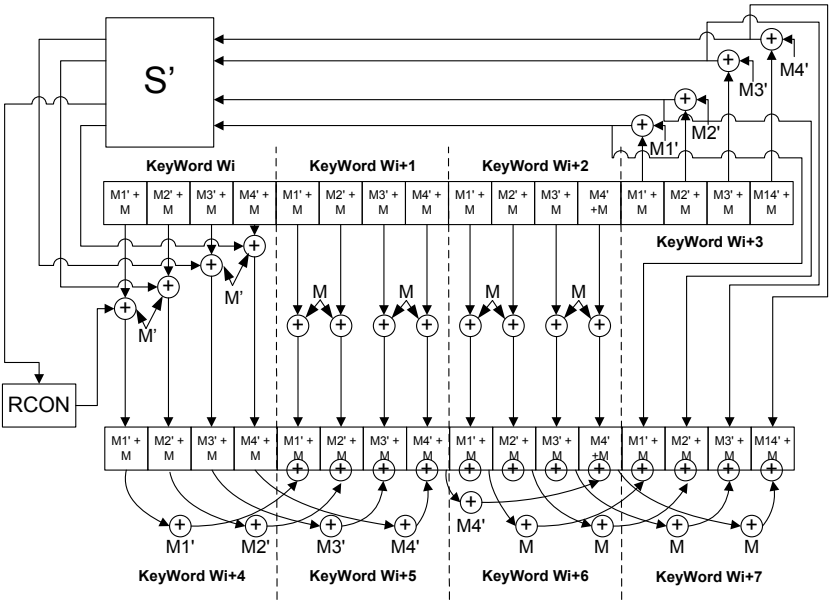

Fig. 1. Masking scheme for all but the last AES round keys

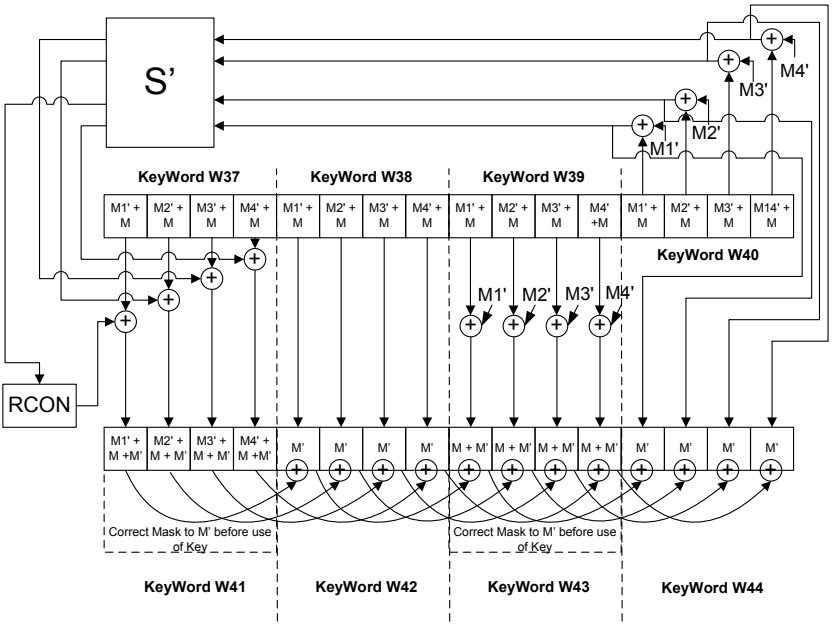

Fig. 2. Masking scheme for the last AES round key

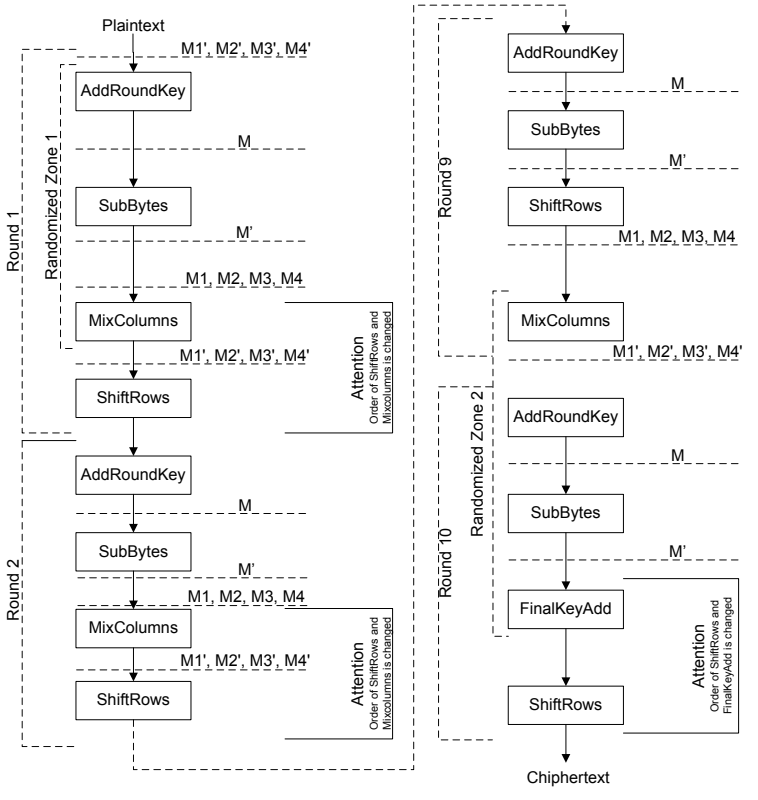

Fig. 3. Programm flow of a randomized and masked AES encryption

the two randomization approaches are applied. In between the two zones, the implementation of AES is protected by masking only.

Randomization Zone 1. Randomization Zone 1 includes the three transformations AddRoundKey, SubBytes and MixColumns. Note that the sequence of ShiftRows and MixColumns is changed. Therefore, we have to change the definition of one column of the state, see Fig. 4.

As discussed before, every operation that is included in Randomization Zone 1 allows some randomization. The idea of the randomization that we use is simple. We choose a block of operations that processes a single column of the AES state, see Fig. 5. This block of operations needs to be executed four times to process the complete AES state. We can choose the sequence of the columns randomly. Within each column, we can also choose the sequence of rows. Hence, in total there are 4×4 different ways of processing one AES state. In addition to this inner randomization we can add a certain number of dummy blocks of instructions, see Fig. 5. A variable called Max_Ops defines the amount of additional blocks added. If n blocks are added, then there are $16+4\times n$ different ways of computing Randomization Zone 1.

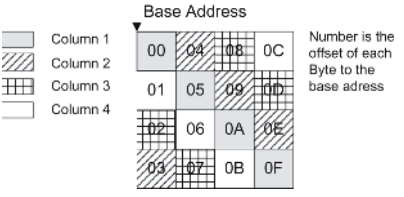

Fig. 4. The definition of a column of the state takes into account that MixColumns is performed before ShiftRows

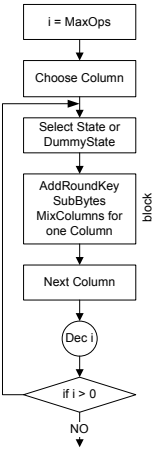

Fig. 5. The Definition of Randomization Zone 1

Randomization Zone 2. Randomization Zone 2 includes the MixColumns operation of round nine, two AddRoundKey transformations, and a SubBytes transformation. In this randomized zone, the order of the final key addition and ShiftRows are changed. To compensate for this change, an InverseShiftRows transformation is applied to the last round key.

3.3 Performance Analysis

The implementation of countermeasures against power analysis attacks does not come for free. Additional memory and additional operations are necessary for masking and randomization. In Tab. 1, we compare the execution time in clock cycles (*cc*) of our implementation against several other protected and unprotected AES smart card implementations. We focus on implementations for AVR and 8051-based 8-bit microcontrollers. Compared are clock cycles for full 128-bit AES encryptions that include the key schedule. The first part of Tab. 1 compares different unprotected AES implementations and serves as a reference. There is a notable difference between the amount of clock cycles between the AVR-based and the 8051-based implementations. Implementations that use masking only

Table 1. Comparison of AES implementations for 8-bit smart card processors

Implementation Type	AVR	8051
AES	7498*cc* [RÖ3] 4427*cc* [Ins06]	90500*cc* [AG01] 46860*cc* [Ins06] 38016*cc* [DR98]
masked AES	8420*cc*	293500*cc* [AG01]
masked & randomized AES	$11845 + n \times 240cc$	

are compared in the second part of the table. Our implementation takes around 8420 clock cycles, which is roughly two times slower than the best unmasked implementation. In contrast, the multiplicative masking scheme [AG01], which was implemented for 8051-based smart cards, requires roughly 7 times more clock cycles than the best unmasked 8051-based implementation. The third part of the table shows the performance figure for our masked and randomized implementation. It takes 11845 clock cycles when no additional blocks are added. This increases the running time by a factor of 3 compared to the unmasked AVR-based implementation. When n blocks are added $11845 + n \times 240$ clock cycles are needed.

4 Security Analysis

The countermeasures that we have implemented are both well known and several papers on their effectiveness have been published. In this section we provide arguments why a combination of them provides resistance against power analysis attacks. First, we provide a theoretical assessment. Then, we report on the practical results that we have obtained.

4.1 Theoretical Analysis

We use a combination of masking and randomization to counteract various types of power analysis attacks. Our implementation is secure against simple power analysis attacks and template attacks because all intermediate values are masked. For the same reason, our implementation is secure against (first-order) DPA attacks. We are also resistant against second-order DPA attacks for the following reasons. Remember that in our implementation, the execution of AES starts and ends with a randomization zone. Within that zone, an operation occurs at a certain position only with probability $p = 1/(16 + 4 \times n)$, where n denotes the number of blocks and is defined by the designer. Consequently, a second-order DPA attack on operations within the randomization zone will produce a peak with height reduced by a factor of $p = 1/(16 + 4 \times n)$ and require $(16 + 4 \times n)^2$ more measurements than a standard second-order DPA attack. Consequently, n can be chosen such that an attack gets impractical. A second-order DPA attack outside the randomization zone requires either to predict two intermediate value that occurs after the MixColumns operation, or to predict one value that occurs after MixColumns and one that is in the randomization zone. Any intermediate value that occurs after MixColumns depends on 32 bits of the round key. Consequently, in order to make a second-order DPA attack on two bytes after MixColumns, the attacker has to guess at least 32 bits of the round key. This leads to a huge number guesses that need to be tested; we consider this to be impractical. For an attack on one value after MixColumns and one value in the randomization zone, the attacker needs to guess 32 bits of the key and needs $(16 + 4 \times n)^2$ times more traces than in a standard second-order DPA attack. We consider this to be impractical as well.

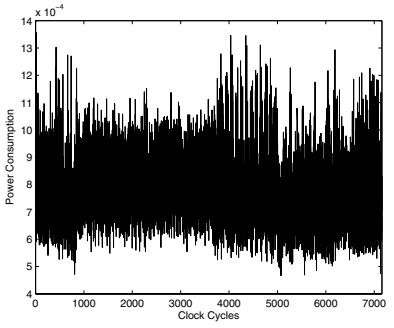

Fig. 6. Power trace of a masked AES encryption

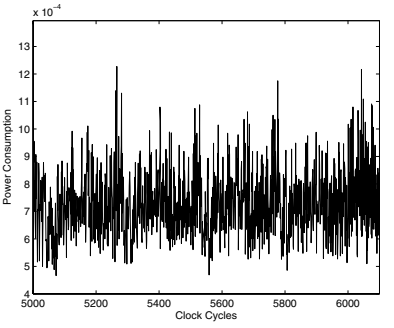

Fig. 7. Power trace showing the execution of the four columns only

4.2 Practical Analysis

We have applied first-order and second-order DPA attacks to a practical implementation of our secured AES. No first-order DPA attack has succeeded. In this section we report on one of the second-order DPA attacks that we have used to verify the theoretical estimates for the increase of the number of samples for a second-order DPA attack. Therefore, we have limited the amount of randomness that we introduce in the randomization zones to a factor of 4: no additional rounds are executed and there is no randomization of columns, only of rows.

Figure 6 shows the power consumption that we have acquired during the calculation of such an AES encryption. Each point in the trace represents one clock cycle. In the trace, several steps of the computation can be located. Between clock cycle 1000 and 3800 the pre-processing of the masked SubBytes table takes place. This calculation is followed by the masked key scheduling part of the algorithm which lasts approximately until clock cycle number 4900. Thereafter, until clock cycle 6100, Randomization Zone 1 is processed. We zoom into this part of the trace in Fig. 7. One can locate the four inner loops that correspond to the processing of the four columns. The first column is processed between clock cycle 5000 and 5200. Therefore, we have attacked this part of the trace with a second-order DPA attack.

Our attacked followed the scenario that has been described in Sect. 3.3 of [OMHT06]. In this scenario, one attacks two SubByte outputs. In [OMHT06], a theoretical estimate for the height of the correlation coefficient was given. The reported correlation coefficient was 0.24. This value can only be achieved under the assumption that the device leaks the Hamming weight of the processed data. Our smart card does not leak the Hamming weight. It leaks the Hamming distance of the data and the value that was manipulated before. Typically, the attacker does not know that value. Hence, the maximum correlation coefficient for our device is lower. We have assessed this height based on another unprotected AES implementation on the same device. It turned out, that the height is 0.7. We use this factor to scale the correlation coefficient

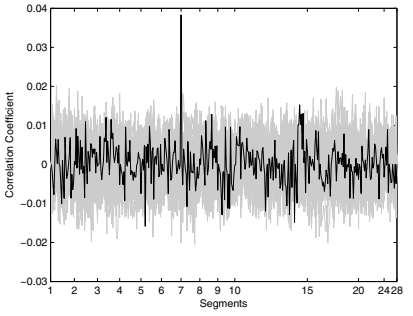

Fig. 8. The result of all key guesses in an attack

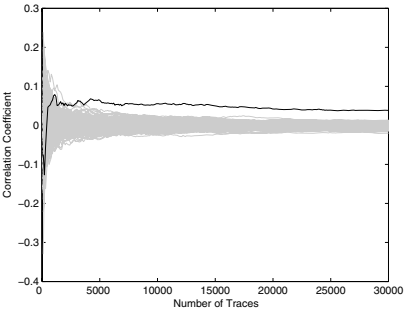

Fig. 9. Correlation coefficients for all keys depending on the number of power traces that are used in the attack

that was reported in [OMHT06]; the expected height of a second-order DPA on our implementation is therefore $0.24 * 0.7 = 0.168$. In the experiment that we performed, where only one column is randomized and no additional blocks are added, we expect a further decrease of the height by a factor of 4. Consequently, we expected to produce a peak of height 0.04 in a second-order DPA attack on the randomized AES. Figure 8 shows the result of the attack. It can be seen that for one of the segments (see [OMHT06] for a detailed explanation of the attack and the notation) we indeed produce a peak with a height that is roughly 0.04 for the correct key guess. Figure 9, shows the run of the correlation coefficient for an increasing number of samples. In both figures, the graphs for the incorrect key guesses are plotted in gray color and the graph for the correct key guess is plotted in black color. The results of this experiments confirm the theoretical estimates that we took from [CCD00] and [Man04].

5 Conclusion

In this article we have described an AES software implementation that is suited for 8-bit smart cards and that is resistant against power analysis attacks. Our implementation masks the intermediate results and introduces randomization at the beginning and the end of the execution. It is secure against simple power analysis attacks, template attacks and first-order DPA attacks because of masking. Due to the combination of masking and randomization, it is resistant against higher-order DPA attacks. Resistance means that a large amount of measurements has to be acquired for a successful attack. Our implementation compares well with other protected and unprotected AES software implementations for smart cards. The practical attacks that we have performed support our theoretical estimates about the security of the countermeasures.

References

[AG01] Mehdi-Laurent Akkar and Christophe Giraud. An Implementation of DES and AES, Secure against Some Attacks. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 309–318. Springer, 2001.

[BGK05] Johannes Blömer, Jorge Guajardo, and Volker Krummel. Provably Secure Masking of AES. In Helena Handschuh and M. Anwar Hasan, editors, *Selected Areas in Cryptography, 11th International Workshop, SAC 2004, Waterloo, Canada, August 9-10, 2004, Revised Selected Papers*, volume 3357 of *Lecture Notes in Computer Science*, pages 69–83. Springer, 2005.

[CCD00] Christophe Clavier, Jean-Sébastien Coron, and Nora Dabbous. Differential Power Analysis in the Presence of Hardware Countermeasures. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, volume 1965 of *Lecture Notes in Computer Science*, pages 252–263. Springer, 2000.

[DR98] Joan Daemen and Vincent Rijmen. AES proposal: Rijndael. First AES Conference, August 1998.

[Ins06] Institute for Applied Information Processing and Communication, Graz University of Technology. VLSI Products–Software Modules. http://www.iaik.tugraz.at/research/vlsi/02_products/index.php, January 2006.

[JPS05] Marc Joye, Pascal Paillier, and Berry Schoenmakers. On Second-Order Differential Power Analysis. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 293–308. Springer, 2005.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In Michael Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[Man03] Stefan Mangard. A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion. In Pil Joong Lee and Chae Hoon Lim, editors, *Information Security and Cryptology - ICISC 2002, 5th International Conference Seoul, Korea, November 28-29, 2002, Revised Papers*, volume 2587 of *Lecture Notes in Computer Science*, pages 343–358. Springer, 2003.

[Man04] Stefan Mangard. Hardware Countermeasures against DPA – A Statistical Analysis of Their Effectiveness. In Tatsuaki Okamoto, editor, *Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings*, volume 2964 of *Lecture Notes in Computer Science*, pages 222–235. Springer, 2004.

[Nat01] National Institute of Standards and Technology (NIST). FIPS-197: Advanced Encryption Standard, November 2001. Available online at `http://www.itl.nist.gov/fipspubs/`.

[OMHT06] Elisabeth Oswald, Stefan Mangard, Christoph Herbst, and Stefan Tillich. Practical Second-Order DPA Attacks for Masked Smart Card Implementations of Block Ciphers. In David Pointcheval, editor, *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings*, volume 3860 of *Lecture Notes in Computer Science*, pages 192–207. Springer, 2006.

[OMPR05] Elisabeth Oswald, Stefan Mangard, Norbert Pramstaller, and Vincent Rijmen. A Side-Channel Analysis Resistant Description of the AES S-box. In Henri Gilbert and Helena Handschuh, editors, *Fast Software Encryption, 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Proceedings*, volume 3557 of *Lecture Notes in Computer Science*, pages 413–423. Springer, 2005.

[OS06] Elisabeth Oswald and Kai Schramm. An Efficient Masking Scheme for AES Software Implementations. In J. Song, T. Kwon, and M. Yung, editors, *WISA 2005*, volume 3786 of *Lecture Notes in Computer Science*, pages 292–305. Springer, 2006.

[RÖ3] Christian Röpke. Praktikum B:Embedded Smartcard Microcontrollers. http://www.christianroepke.de/studium_praktikumB.html, 2003.

[SPQ05] Francois-Xavier Standaert, Eric Peeters, and Jean-Jacques Quisquater. On the Masking Countermeasure and Higher-Order Power Analysis Attacks. In *ITCC 2005*, 2005.

[WW04] Jason Waddle and David Wagner. Towards Efficient Second-Order Power Analysis. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004, 6th International Workshop, Cambridge, MA, USA, August 11-13, 2004, Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2004.