An Improved Power Analysis Attack Against Camellia's Key Schedule

Lu Xiao⋆ and Howard M. Heys†

⋆ QUALCOMM Incorporated
lxiao@qualcomm.com

† Electrical and Computer Engineering
Memorial University of Newfoundland
howard@engr.mun.ca

September 22, 2005

Abstract

This paper presents an improved simple power analysis attack against the key schedule of Camellia. While the original attack required an exact determination of the Hamming weight of intermediate data values based on power measurements, in this paper, two variants of the simple power analysis attack are presented and shown to be tolerant of errors that might occur in the Hamming weight determinations. In practical applications of the attack such errors are likely to occur due to noise and distortion in the power measurements and their mapping to the Hamming weights of the data. Further, we propose a practical method to evaluate the susceptibility of other block ciphers to simple power analysis attacks. To resist these attacks, the required design rationale of key schedules and several practical countermeasures are suggested.

1 Introduction

Proposed by NTT and Mitsubishi in 2000, Camellia [1] is a 128-bit block cipher with a Feistel round structure and supports 128-, 192-, and 256-bit keys. Two logic functions (called FL- and FL^{-1}-functions) are inserted every 6 rounds for security enhancement. It has been shown in [2, 3, 4] that Camellia is well designed to resist differential, linear, and integral attacks. Camellia was included together with AES [5] into the NESSIE portfolio of 128-bit block ciphers in February 2003 [6].

Introduced in [7], power analysis exploits the fact that the power consumption of some cryptographic implementations, such as smart cards, is dependent on the intermediate data values. It is indicated in [8] that there is a roughly linear relation between the Hamming weight of the data and the power consumed at the associated clock cycle. The Hamming weight attack against the key schedules of DES and AES were discussed in [9, 10], and it is shown that the cipher key can be successfully deduced given accurate leakage information of Hamming weights. The susceptibility of NESSIE candidates to power attacks was theoretically evaluated in [11], which mainly focused on differential power analysis and gave Camellia a high rank among others.

In [12], it is shown that Camellia is susceptible to a simple power analysis attack under the assumption of an exact correlation between power measurements and the Hamming weights of intermediate data values. In this paper, we modify the attack to make the attack robust in the presence of errored Hamming weight values derived from noisy power measurements. Since real Hamming weight determinations are not likely to be error free, the modified attack represents a significant improvement in the attack and raises increased concerns over the susceptability of Camellia to power attacks. Also in this paper, a method is proposed to evaluate how vulnerable a block cipher is toward similar attacks and countermeasures in terms of both design rationale and implementation are suggested.

2 Description of Camellia's 128-Bit Key Schedule

The attack described in this paper is focused on Camellia's 128-bit key schedule [1]. The attacking technique to be discussed can be easily modified for 192- and 256-bit key schedules.

Camellia's 128-bit key schedule expands 26 subkeys of 64 bits from the original key K_L and another derived key K_A of 128 bits. Each subkey can be obtained as one half of K_L or K_A after they are left rotated for a specific number of bits. This number can be 0, 15, 30 (only for K_A), 45, 60, 77 (only for K_L), 94, or 111, depending on the round number. During encryption or decryption, 16 subkeys are used for the round function in the 16 rounds. The other 8 subkeys are used for pre-, post-whitening and the FL-, FL^{-1}-functions.

K_A is derived from the original key K_L through a Feistel network. As shown in Figure 1, K_L is the input of such a network. The left half is the input to the same round function as in encryption. The round function can be divided into 3 steps: (1) a 64-bit constant, denoted as Σ_i for round i, is eXclusive-ORed (XORed) with the input, (2) the S-function performs byte-wise bijective substitution, and (3) the P-function performs a linear transformation. The output of the round function is XORed with the right half of the round input. The two halves are then swapped. This Feistel structure is iterated 4 rounds for the 128-bit key schedule. Note that the intermediate result after 2 rounds is XORed with K_L to form the next round input. Each 64-bit block in Figure 1 is labelled as T_i, $0 \leq i \leq 17$.

3 Hamming Weight Attack

The Hamming weight attack exploits the relation between data and its Hamming weight. If the Hamming weight can be captured from a poorly designed cryptographic device, we can use it to eliminate those data candidates failing to meet this relation. The original simple power analysis attack, based on exact Hamming weight information is presented in [12]. In this section, we present some of the basic concepts of the attack, in order to provide the context to understand the attack variants, presented in the next section.

Given a Hamming weight of h for a particular byte, there are $\binom{8}{h}$ byte values consistent with this weight. Hence, as deduced in [13, 10], the number of byte values consistent with a Hamming weight is expected to be

$$\sum_{h=0}^{8} Prob\{H = h\} \binom{8}{h} = \sum_{h=0}^{8} \frac{1}{256} \binom{8}{h}^2 \approx 50.27 \; . \tag{1}$$

Thus, to attack a block cipher with 128-bit key running on an 8-bit processor, the leakage of Hamming weight information for each key byte straightforwardly enables attackers to reduce the possible key space from 2^{128} to 50.27^{16} ($\approx 2^{90.43}$). However, dependent on the nature of a block cipher, the outcome of a Hamming weight attack could be much better than this reduced workload if many intermediate values are derived from a small subset of key or

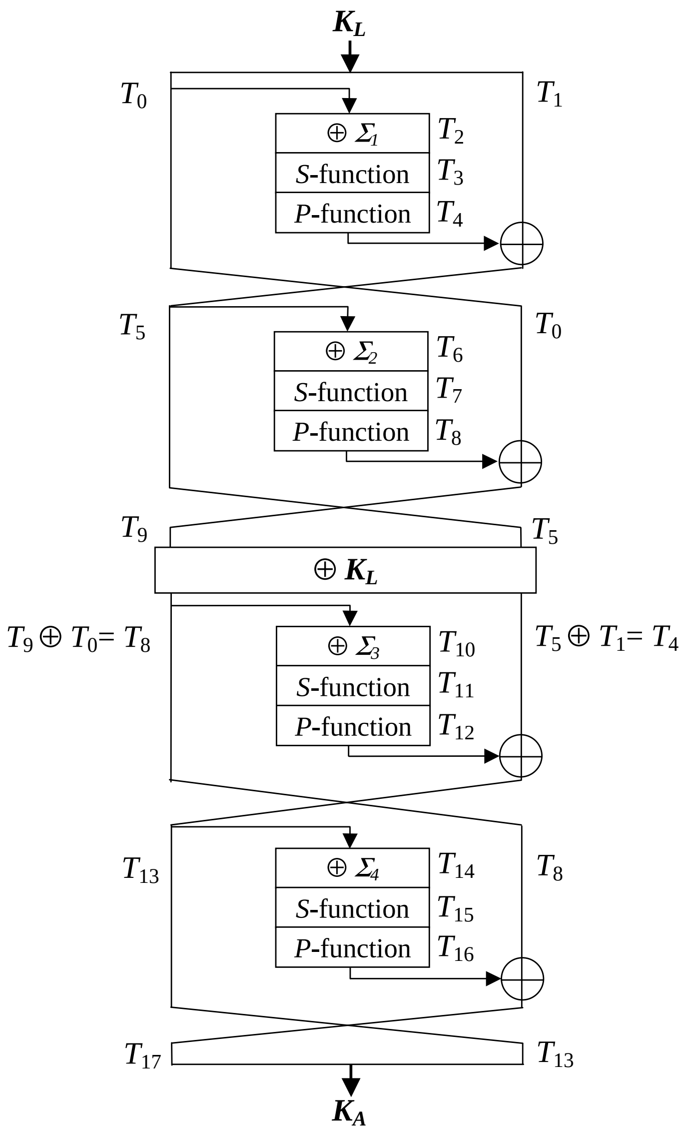

Figure 1: Camellia's 128-bit Key Schedule

subkey bits.

3.1 Basic Power Leakage Model

A popular power leakage model was proposed in [8] with two assumptions. One assumption is that the processor leaks the Hamming weights of data being processed. It is also assumed that the power consumed by the processor demonstrates a linear relation to the Hamming weight of the processed data. As defined in [8], the power consumption at a specific time j

is

$$P[j] = \varepsilon \cdot H[j] + L + n \tag{2}$$

where $H[j]$ is Hamming weight at time j, L is the additive constant portion in the power trace, ε is a power-related constant, and n is a random variable with zero mean representing noise. In the basic model, we assume that the power consumption monotonically varies in relation to the change of the Hamming weight of processed data. Hence, the power consumption is

$$P[j] = f(H[j]) + L + n \tag{3}$$

where $f(\cdot)$ is a monotonically increasing or decreasing function. In the basic model, we assume that the influence of L and n can both be ignored by averaging and offsetting the power traces. Therefore, the Hamming weight can be reliably quantized from $P[j]$. The attack discussed in [12] is based on this model.

3.2 Requirements for the Attack

In general, in order to launch a Hamming weight attack, the following prerequisites have to be satisfied.

- *Access to the power consumption.* The attacker needs to collect the power consumption traces from the cryptographic device when the same cipher key is used. A typical approach is to sample the power dissipated by a small resistor, which is inserted between external power or ground and its corresponding pin on the smart card.

- *Ability to identify the clock cycles for individual steps in the key schedule.* For example, if the attacker knows the implementation well (e.g., a former employee), the timing information can be easily determined. Alternatively, a general method is suggested in [9] to distinguish the periods used for the key schedule from periods associated with data processing. The basic idea is to execute the protocol many times on several smart cards, each with different user information. Then, statistical analysis is performed to identify those clock cycles in which the same card behaves similarly with various data to be encrypted but different cards behave differently even if the same data is encrypted. These clock cycles are assumed to be used for the key schedule. Within these periods, the attacker can identify the clock cycles for specific operations based on features of the key schedule (e.g., DES and IDEA [14] both have rotations for subkey generation).

- *Monotonic relation between power and Hamming weight.* The power consumed by the attacked device has to be at least a monotonic function of the Hamming weight of processed data. Although the basic power leakage model of (2) is linear, as well as monotonic, in fact, the monotonic nature of the function is sufficient.

- *One pair of plaintext and ciphertext.* The Hamming weight attack is expected to reduce the key space to a small subset. The cipher key is then distinguished by checking whether one hypothesized key can be used to encrypt the plaintext to the expected ciphertext. For Camellia, when enough Hamming weights can be collected, we can deduce all key bits with certainty without requiring a plaintext encryption and in this case, this requirement is not necessary.

3.3 Basic Attack Against the Key Schedule

In Section 3 of [12], the basic simple power analysis attack of Camellia's key schedule is presented. The attack is described as it is applicable to 8-bit smart card implementations of Camellia. The attack assumes that the Hamming weight determinations of intermediate data byte values are exact. That is, the power measurement noise is small enough that the power measurements can be perfectly translated into the corresponding Hamming weights, with no errors in the derived Hamming weights. The attack is divided into 2 steps: the first step exploits the rotational relations between K_L and the resultant subkeys and the second step exploits relations in the derivation of K_A from K_L. The basic principle of the attack is to use the Hamming weights of certain byte partitions determined from power measurements to verify the correctness of candidate partial keys. This is done by having the Hamming weights of intermediate data values during the key scheduling tested for consistency with candidate partial keys. If the candidate partial key is not consistent with the determined Hamming weight, the candidate partial key is discarded. This is continued until all but the one correct candidate is left or until there are few enough potential candidates left to easily determine the correct key through exhaustive search. The attack in [12] is presented using identical notation to this paper and we refer the reader to [12] for the details.

In [12], the attack is applied to Camellia's 128-bit key schedule with 10,000 randomly generated sample keys. The experimental results listed in Table 1 show that 2 rounds of Hamming weight checks in K_A's derivation is enough for unique key identification in most cases. It costs less than 5 ms to compute the possible key candidate(s) in a PIII 933MHz computer with 512 MB memory.

Table 1: Experimental Attack Results with 10^4 Samples of 128-Bit Camellia Cipher Keys [12]

Scope of HW checks in K_A's derivation	$T_0 \sim T_7$	$T_0 \sim T_8$	$T_0 \sim T_9$	$T_0 \sim T_{10}$
Percentage of case identification with unique key	14.04 %	97.49 %	99.98 %	100 %
Ave. # of spurious keys	5.3588	0.0264	0.0002	0

3.4 Extension to 192-Bit and 256-Bit Key Schedules

When the key size is 192 or 256 bits, K_L is the first 128 key bits. The remainder of the key is denoted as K_R, which is also rotated to generate subkeys. For 192-bit keys, K_R's right 64 bits are padded with the complement of its left 64 bits. The input of K_A's derivation is changed from K_L to $K_L \oplus K_R$. Another derived key K_B is obtained through two rounds of Camellia's encryption structure with $K_A \oplus K_R$ as input.

Similar to the attack against the 128-bit key schedule but in the reverse direction, the attack begins with the last round of the Feistel structures used to derive K_A and K_B. Combined with Hamming weight checks during the rotations used in the generation of subkeys, a small number of K_A and K_B candidates are expected to pass the test. Using a combination of these candidates, the number of valid candidates for K_L and K_R can be reduced dynamically. It is unlikely for wrong guesses of K_A and K_B to deduce K_L and K_R able to pass Hamming weight checks.

4 Two Variants of Attack with Robustness to Measurement Errors

A Hamming weight attack is normally fast and easy to implement when all required Hamming weights are measured accurately. However, in real circumstances, imperfect measurement cannot be always avoided. An attacker could attempt to mitigate the measurement noise using some statistical methods (e.g., averaging) in order to keep measurement accuracy at a satisfactory level. The attack presented in [12] is not error tolerant. As a result, a spurious key or no key could be recognized as the correct key when measurement noise is high enough to cause errors in the determination of Hamming weights. Two modified attacks are thus given to tolerate errors.

4.1 Noisy Power Leakage Model

Denote $h[j]$ as the Hamming weight quantized from the power trace at time j in this model. Since $f(\cdot)$ is not always linear, the error during quantization needs to be considered. A number of wrong captures of clock cycles may also occur. This is caused by imperfect understanding of timing information about the implementation. Thus, the Hamming weight processed by an unrelated instruction may be wrongly recognized. Let ΔH denote the offset due to possible small errors from measurement and quantization. Typically, $\Delta H = 1$. Then, the real Hamming weight obtained from measurement equipment is modelled as

$$h[j] = \begin{cases} H[j] \pm \Delta H & with\ P = P_\alpha \\ rand([0, \cdots, H_{max}]) & with\ P = P_\beta \\ H[j] & with\ P = 1 - P_\alpha - P_\beta \end{cases} \quad (4)$$

where P_α is the probability that $h[j]$ is wrongly quantized as its adjacent Hamming weight and P_β is the probability that the result is uniformly randomly taken due to wrong recognition of clock cycles or other severe noise. When $P_\alpha = P_\beta = 0$, the leakage model is equivalent to the basic model in Section 3.1 and assumed in [12].

4.2 Attack Variant 1 Robust Against Small Noise

The "small" noise mentioned here means that its effect is only able to cause an error no more than ΔH on the measured Hamming weight. Such type of noise suits the power leakage model given by (4) where $P_\beta = 0$. To tolerate these small errors, the only modification in the attack is to change the method of Hamming weight checks. Instead of considering whether the two Hamming weights from a candidate byte partition and measurement (denoted by h' and h, respectively) are the same in order to determine the viability of the candidate, a candidate byte remains viable if $|h' - h| \leq \Delta H$.

Since the current Hamming weight comparison is looser than equality checking, a wrong key guess is more likely to pass the test. This attack variant costs more time and memory because a wrong key guess may need more checks to be eliminated. However, Camellia's K_A derivation provides checks up to T_{17} and these can all be used to eliminate wrong keys. The processing times used to perform the attack[1] with different error rates P_α are listed in Table 2, where $\Delta H = 1$. Note that when P_α is high, the processing time is short. This is because when the small measurement errors occur more frequently, it is more likely for candidate Hamming weight h' passing the current Hamming weight comparison to be farther from the Hamming weight of the actual key, thus, more likely for its associated key guess to fail in next Hamming weight comparison.

Table 2: Processing Times of Attack Variant 1 on a PIII 933MHz Computer

Error rate P_α	1	0.8	0.6	0.4	0.2	0
Processing time	13 mins	45 mins	7.2 hours	2.2 days	$\approx$ 7 days	$\approx$ 70 days

4.3 Attack Variant 2 Robust Against Wide Range of Noise

Attack variant 1 overcomes the effects of small errors in Hamming weight measurement whether frequently happening or not. However, in some systems a wide range of noise may occur due to wrong recognition of clock cycles associated with Hamming weight measure-

[1]Based on randomly generated key $K_L = \{$D7, 13, E8, 80, 5F, FD, E3, 9E, 1B, C6, CF, 4D, F4, C7, 66, EF$\}$ in hexadecimal.

ments or severe external interference. When a clock cycle is wrongly recognized, h may be any integer among $[0, H_{max}]$ dependent on the data processed at that moment. The occurrence of this type of error is reflected in a nonzero value for P_β in (4). In this circumstance, attack variant 1 could lead to a correct byte failing a check and being eliminated and eventually to determination of an incorrect key. Attack variant 2, however, can be employed to attack the key schedule when wide range of noise is unavoidable, i.e., $P_\alpha, P_\beta > 0$.

Instead of dynamically pruning key guesses through a local Hamming weight comparison, a weighted comparison scheme is applied. Each Hamming weight check now returns a weight w which measures the difference between $h^{'}$ and h:

$$w = W[|h^{'} - h|] \ .$$

The entry value of array $W[\cdot]$ depends on the error distribution and drops to 0 as the index rises (e.g., in our experiment, $W[0] = 5, W[1] = 2, W[2] = \cdots = W[H_{max}] = 0$). Let S_w denote the sum of return values from n Hamming weight comparisons for the bytes of a partial key candidate. When a candidate partial key is true and $\Delta H = 1$ (see (4)), it is expected that

$$S_w = \sum_{i=1}^{n} W[h_i^{'} - h_i] \approx (1 - P_\alpha - P_\beta) \cdot n \cdot W[0] + P_\alpha \cdot n \cdot W[1] \tag{5}$$

when $W[2] = \cdots = W[H_{max}] = 0$. Thus, the probability of the following inequality being true is quite high

$$S_w \geq \eta \cdot (1 - P_\alpha - P_\beta) \cdot n \cdot W[0] \tag{6}$$

when n is large enough and $0 \leq \eta \leq 1$. A smaller η makes (6) more likely to be true, but allows more spurious partial keys to pass the test.

If all of the left half of K_L is hypothesized to calculate S_w, the processing time for an exhaustive search in 2^{64} candidates will be formidable. Therefore, we use a nested EDST approach illustrated in Figure 2. The left half of K_L is divided into 3 parts. The weight sum S_w is calculated for each candidate partial key. Given a specific η, the candidates will be discarded if inequity (6) cannot be satisfied. The remaining candidates are sorted according to S_w and only λ candidates with high S_w will be stored to form larger candidate partial keys.

Within affordable computation, the attack prefers a small value of η and a large value of λ so that the true guess will not be lost due to errors. In the experiment to attack Camellia's key schedule with 20 randomly generated keys as samples, the EDST approach has been run for 2, 3, and 8 byte candidate partial keys with $\eta = 0.5, 0.7$, and 0.8, respectively. The percentages of small noise and wide ranging noise are both 10% (i.e., $P_\alpha = P_\beta = 0.1$). When $\lambda = 256$, 30% of keys can be uniquely identified in a average time of 74 hours; 45% of keys will be uniquely identified with more processing time when $\lambda = 512$.

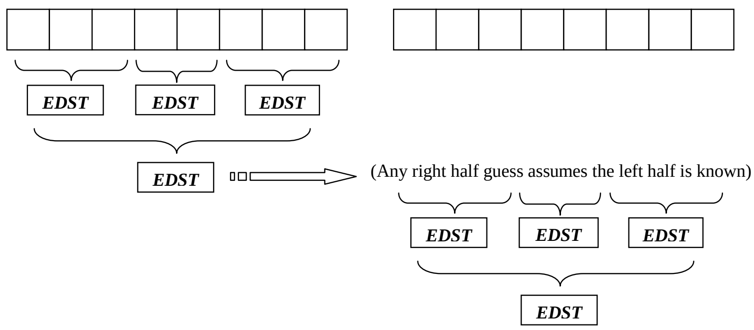

Figure 2: A Nested EDST Approach
(**E**: Evaluate S_w for each candidate; **D**: Discard if not satisfying (6);
S: Sort remaining candidates with S_w; **T**: Truncate and keep first λ candidates.)

5 General Susceptibility Evaluation

The Hamming weight attack and its variants described in this paper also work for the key schedule of some other ciphers. Two main measures are of interest for this attack: (1) the size of targeted partial key space (denoted by Ω) that the attack begins with; (2) the average number of Hamming weight checks per byte in the targeted partial key space, denoted as ξ. An attacker hopes to find a scenario to reduce the candidates in Ω. A small Ω implies a low workload for exhaustive search within Ω. A high ξ leads to a small number of valued candidates left after attacking. Assuming the operations in the key schedule to be independent of each other, the number of candidates left is expected to be $256(\frac{50.27}{256})^{\xi}$. This implies that when $\xi > 3.41$, it is possible to reduce the number of valid partial key candidates close to one. In a real attack, ξ has to be much larger than 3.41 because most

Table 3: Susceptibility Evaluation for Several Block Ciphers

Ciphers	$\|\Omega\|$	ξ	Comments
AES	2^{40}	4.4	mainly exploit $\oplus$
Camellia	2^{8m+4}	≈ 6.22	exploiting rotation only
DES	2^{8m+8}	≈ 8	exploit rotation
IDEA	2^{8m+6}	≈ 6.5	exploit rotation
SAFER++	2^{8m+8}	≈ 24	exploit rotation and byte-wise addition
SHACAL-0	2^{32}	4	$\oplus$ without rotation hypothesize 1 byte in each word
SHACAL-1	2^{64}	≈ 3.5	$\oplus$ with rotation hypothesize 2 bytes in each word

operations are correlated (such as the fixed rotations of Camellia). For the attack in [12], $|\Omega| = 2^{36}$, ξ=6.22.

Table 3 shows the susceptibility of DES, IDEA, SAFER++ [15], AES (deduced from [10]), SHACAL-0 and SHACAL-1 [16] toward similar attacks. The values of $|\Omega|$ and ξ listed in this table are based on our assessment of values that can lead to a real attack. It is possible that more efficient attacking scenarios exist with more desirable $|\Omega|$ and ξ. No evident vulnerability to the attack from the key schedules of MISTY1 [17], Khazad [18], SHACAL-2 [16], and RC6 [19] are observed.

6 Countermeasures

Hamming weight attacks, like other simple power attacks, work well only on poorly implemented cryptographic devices. Most countermeasures require additional operations and diminish performance. From the viewpoint of a cipher designer, a key schedule is resistant to a Hamming weight attack in nature if a good avalanche effect exists from the cipher key to subkeys as well as from one subkey to another. As a result, a very large Ω (ideally the whole key space) has to be hypothesized to get a number of ξ high enough for key identification. From the viewpoint of a system designer, a 16- or 32-bit smart card implementation is desirable because a larger word size decreases the number of possible Hamming weight checks and makes measurement harder and less accurate. To provide resistance to a cipher already designed on 8-bit smart cards, the following countermeasures can be selected during implementation:

- One popular approach is to mask operations with random content. For example, $Z = X \oplus Y$ can be implemented with $Z = ((X \oplus \Delta X) \oplus Y) \oplus \Delta X$. The random data ΔX enlarges $|\Omega|$ with an expected factor of 50.27.

- Some operations in key schedules are commutative and distributive, e.g., $(X \oplus Y) <<< 1 = (Y <<< 1) \oplus (X <<< 1)$. It is hard for attackers to recognize the proper clock cycles from power traces if the program switches these equivalent operations randomly or data-dependently (e.g., reverse order of $\oplus$ and $<<<$ when X is odd). Thus, the measurement Hamming weight h could be unrelated to candidate Hamming weight h' due to wrong clock cycle recognition, which makes P_β larger.

- A more resistant CMOS technology proposed in [20] can be used for smart cards if applicable. The power consumed by these types of circuits do not depend on the data being processed.

7 Conclusions

Camellia has a key schedule with high agility. K_A's derivation brings nonlinear properties into subkeys and gains more resistance to slide and related-key attacks. However, as shown in [12], based on the assumption of accurate power measurements resulting in perfect Hamming weight determinations, rotations used to generate subkeys provides enough information about K_L to compromise the key. Further, the fact that K_L is used as the input of K_A's derivation structure provides attackers enough information to launch a Hamming weight attack to uniquely identify the key. The Feistel structure of K_A's derivation gives many chances to verify the hypothesis. The two attack variants introduced in this paper exploit this redundancy to gain robustness in the presence of errors in the Hamming weight measurements. Consequently, when Camellia is implemented in the device with Hamming weight leakage, it is very important for implementors to consider appropriate countermeasures as discussed in Section 6. Moreover, we propose two measures to evaluate the susceptibility of a block cipher against similar attacks.

References

[1] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita, "Camellia: a 128-bit block cipher suitable for multiple platforms - design and analysis," in *Proceedings of Selected Areas in Cryptography - SAC 2000*, vol. 2012 of *Lecture Notes in Computer Science*, pp. 39–56, Springer-Verlag, 2001.

[2] M. Kanda, "Practical security evaluation against differential and linear attacks for Feistel ciphers with SPN round function," in *Proceedings of Selected Areas in Cryptography - SAC 2000*, vol. 2012 of *Lecture Notes in Computer Science*, pp. 324–338, Springer-Verlag, 2001.

[3] T. Shirai, S. Kanamaru, and G. Abe, "Improved upper bounds of differential and linear characteristic probability for Camellia," in *Proceedings of Fast Software Encryption - FSE 2002*, vol. 2365 of *Lecture Notes in Computer Science*, pp. 128–142, Springer-Verlag, 2002.

[4] Y. Yeom, S. Park, and I. Kim, "On the security of CAMELLIA against the square attack," in *Proceedings of Fast Software Encryption - FSE 2002*, vol. 2365 of *Lecture Notes in Computer Science*, pp. 89–99, Springer-Verlag, 2002.

[5] National Institute of Standards and Technology, "FIPS 197 Advanced Encryption Standard (AES)," Available at `csrc.nist.gov/publications/fips`.

[6] New European Schemes for Signatures Integrity and Encryption (NESSIE) website. Available at `www.cosic.esat.kuleuven.ac.be/nessie`.

[7] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proceedings of Advances in Cryptology - CRYPTO'99*, vol. 1666 of *Lecture Notes in Computer Science*, pp. 388–397, Springer-Verlag, 1999.

[8] T. S. Messerges, "Using second-order power analysis to attack DPA resistant software," in *Proceedings of Cryptographic Hardware and Embedded Systems - CHES 2000*, vol. 1965 of *Lecture Notes in Computer Science*, pp. 238–251, Springer-Verlag, 2000.

[9] E. Biham and A. Shamir, "Power analysis of the key scheduling of the AES candidates," in *Second Advanced Encryption Standard (AES) Candidate Conference*, Rome, Italy, 1999.

[10] S. Mangard, "A Simple Power-Analysis (SPA) attack on implementations of the AES key expansion," in *Proceedings of the 5th International Conference on Information Security and Cryptology - ICISC2002*, vol. 2587 of *Lecture Notes in Computer Science*, Springer-Verlag, 2002.

[11] E. Oswald and B. Preneel, "A theoretical evaluation of some NESSIE candidates regarding their susceptibility towards power analysis attacks," Technical report, Katholieke Universiteit Leuven, Dept. ESAT, October 2002.

[12] L. Xiao and H. Heys, "A simple power analysis attack against the key schedule of the Camellia block cipher," in *Information Processing Letters*, vol. 95, pp. 409–412, Elsevier, 2005.

[13] T. Messerges, E. Dabbish, and R. Sloan, "Examining smart-card security under the threat of power analysis attacks," in *IEEE on Transactions on Computers*, vol. 51, pp. 541–552, April 2002.

[14] NESSIE Archive, "The IDEA block cipher," 2000. Available at `www.cosic.esat.kuleuven.ac.be/nessie`.

[15] NESSIE Archive, "SAFER++ submission," 2000. Available at `www.cosic.esat.kuleuven.ac.be/nessie`.

[16] H. Handschuh and D. Naccache, "SHACAL," 2000. Available at `www.cosic.esat.kuleuven.ac.be/nessie`.

[17] M. Matsui, "New block encryption algorithm MISTY," in *Proceedings of Fast Software Encryption - FSE'97*, vol. 1267 of *Lecture Notes in Computer Science*, pp. 54–68, Springer-Verlag, 1997.

[18] P. Barreto and V. Rijmen, "The Khazad legacy-level block cipher," in *First Open NESSIE Workshop*, Leuven, November 2000. Available at `www.cosic.esat.kuleuven.ac.be/nessie`.

[19] R. L. Rivest, M. J. B. Robshaw, R. Sidney, and Y. L. Yin, "The RC6 block cipher," August 2002. Available at `www.rsasecurity.com/rsalabs/rc6`.

[20] K. Tiri, M. Akmal, and I. Verbauwhede, "A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards," in *Proceedings of the 28th European Solid-State Circuits Conference*, Florence, Italy, September 2002.