A Simple Power Analysis Attack Against the Key Schedule of the Camellia Block Cipher

Lu Xiao[1] and Howard M. Heys[2]

[1]QUALCOMM Incorporated, lxiao@qualcomm.com
[2]Electrical and Computer Engineering, Faculty of Engineering and Applied Science,
Memorial University of Newfoundland, howard@engr.mun.ca

Abstract

This paper presents a simple power analysis attack against the key schedule of Camellia. The attack works for the smart card environment which leaks the Hamming weight of data being processed, making use of the Hamming weight to deduce all key bits. It is shown that determining the cipher key given accurate power analysis data is very fast and does not require any pair of plaintext and ciphertext.

1 Introduction

Camellia [1] is a 128-bit block cipher with a Feistel round structure and supports 128-, 192-, and 256-bit keys. In February 2003, Camellia was included together with Advanced Encryption Standard (AES) into the NESSIE portfolio of 128-bit block ciphers [2]. Both the encryption and key schedule of Camellia can be easily implemented on an 8-bit platform [1].

This paper explores Camellia's potential vulnerability to a simple power analysis attack when implemented in an 8-bit smart card environment. A general description of power analysis attacks is available in references [3] and [4]. Using the Hamming weight information leakage model of [4], our attack on Camellia runs faster than the attack on AES and does not require any pairs of plaintext and ciphertext. The attack presented only considers a theoretical, idealized simple power analysis. Further details on the effects of real measurement noise on the applicability of the attack are discussed in [5].

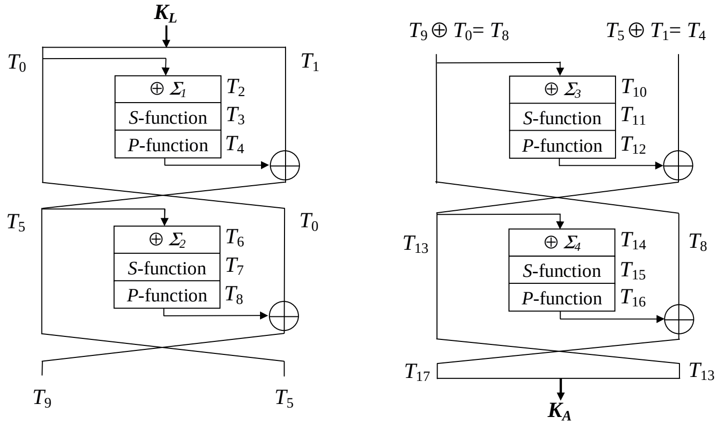

Figure 1: Camellia's 128-bit Key Schedule

2 Description of Camellia's 128-Bit Key Schedule

Camellia's 128-bit key schedule generates 26 subkeys of 64 bits from the original key K_L and another derived key K_A of 128 bits. K_A is derived from K_L as shown in Figure 1, where Σ_i is unique for each round i, S-function performs byte-wise substitution, and P-function performs a linear transformation. 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.

3 Hamming Weight Attack

To attack a block cipher with a 128-bit key running on an 8-bit processor, the leakage of Hamming weight information for each key byte as determined by the measurement of power consumption straightforwardly enables attackers to reduce the possible key space from 2^{128} to $2^{90.43}$, as discussed in [6]. However, depending on the nature of a block cipher, the implementation of a Hamming weight attack could be much simpler than this reduced workload. For example, our attack exploits the redundancy in the key schedule of Camellia and determines all key bits without knowledge of any plaintext and ciphertext pair.

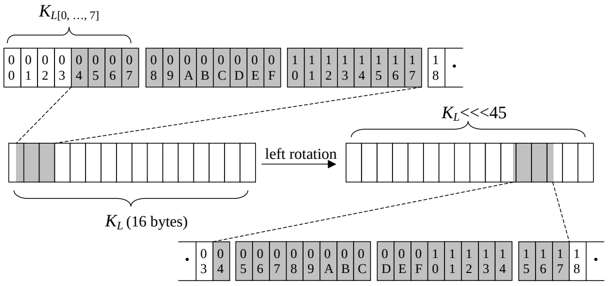

Figure 2: An Example of Subkey Generation
(gray bits: assumed $(8m{+}4)$-bit chunk where $m{=}2$)

3.1 Requirements for the Attack

The attack works with three prerequisites: 1) access to the power consumption information, 2) the ability to identify the clock cycles for individual steps in the key schedule (e.g., using the method suggested in [3]), and 3) a monotonic relation between power and Hamming weight.

3.2 Attack Against Subkey Generation

Our attack is implemented through two steps. The first step exploits the rotational relations between K_L and the resultant subkeys; the second step will exploit relations in the derivation of K_A from K_L. Several 64-bit subkeys are derived from K_L through left rotation for a certain number of bit positions (denoted by "<<<"). Since attackers can only check the Hamming weight of each byte, the rotation offsets (15, 45, 60, 77, 94, 111) provide information determined by the equivalent shift in bit positions as given by the remainder when the rotation offset is divided by 8.

As shown in Figure 2, each rotation of K_L gives a chance to consider bits with a different byte partition due to the shift of bit-positions with bytes. Assuming $8m{+}4$ adjacent bits of K_L are unknown, up to $5m$ Hamming weights collected through power measurement can be used to validate candidates for these key bits. Based on these checks, a dynamic pruning method can be used to reduce the search space over all $8m{+}4$ bits.

The key K_L may be divided into 4 overlapped parts $K_{L[124\sim127,0\sim31]}$, $K_{L[28\sim63]}$, $K_{L[60\sim95]}$, and $K_{L[92\sim127]}$ so that they can be processed quickly and independently. Each part produces a number of 36-bit candidates (i.e., $m = 4$). Any four candidates from these four parts can be joined into one K_L guess when their overlapped bits are consistent. When applied to Camellia's key schedule with 20 randomly generated cipher keys, an average of about 2^{38} candidates of the full K_L pass this step.

3.3 Attack Against the Derivation of K_A

In this section, we examine the second step in the attack, which gains more key information from the steps involved in the derivation of key K_A. In the first round illustrated in Figure 1, each byte of K_L's left half (denoted as T_0), is XORed with constant Σ_1. The result is denoted as $T_2 = T_0 \oplus \Sigma_1$. The following S-function is byte-wise substitution, denoted as $T_3 = S(T_2)$. If we still use $K_{L[124\sim127,0\sim31]}$ and $K_{L[28\sim63]}$ separately to prune partial key space, two more Hamming weight checks for each hypothetical byte can be performed by comparing the Hamming weights in T_2 and T_3 with the corresponding values from measurement. Each output byte of the P-function (denoted as $T_4 = P(T_3)$) depends on at least 5 input bytes. To continue the candidate pruning, we combine any two candidates of $K_{L[124\sim127,0\sim31]}$ and $K_{L[28\sim63]}$ with consistent overlapped bit values to form 8 byte guesses of K_L's left half $K_{L[0\sim63]}$, denoted as $T_0^{'}$. The output of round function with input $T_0^{'}$ is denoted as $T_4^{'}$. If $T_0^{'} = T_0$, then $T_4^{'} = T_4$. Because the Hamming weight per byte in T_4 is known, another 8 Hamming weight checks can be performed to examine each $T_0^{'}$. In most cases, only 1 or 2 possible candidates of the left half of K_L can pass this step.

For each $T_0^{'}$ remaining, the right half of K_L (denoted as T_1) is guessed. The second Feistel round in Figure 1 is expressed as: $T_5 = T_4 \oplus T_1, \quad T_6 = T_5 \oplus \Sigma_2, \quad T_7 = S(T_6), \quad T_8 = P(T_7)$. Similarly to the left half guess, $K_{L[60\sim95]}$ and $K_{L[92\sim127]}$ can be considered separately to prune the partial key space by using three more Hamming weight checks for each byte in $T_5 \sim T_7$. Then, any two candidates of $K_{L[64\sim99]}$ and $K_{L[96\sim128,0\sim3]}$ with consistent overlapped bit values are combined to form a candidate of K_L's right half $K_{L[64\sim127]}$, denoted as $T_1^{'}$. The output of the round function with input $T_0^{'} \oplus T_4^{'}$ is denoted as $T_8^{'}$. If $T_0^{'} = T_0$ and $T_1^{'} = T_1$, then $T_8^{'} = T_8$. Thus, another 8 Hamming weight checks can be performed to validate each $T_1^{'}$ candidate. Similarly, Hamming weight checks can be applied from T_9 to T_{17}.

We applied this attack 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. The attack can be easily extended to 192-bit and 256-bit key schedules [5].

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

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

4 Conclusion

When Camellia is implemented in a device with Hamming weight leakage due to power measurements, it is very important for implementors to consider appropriate countermeasures. The vulnerability of many ciphers to similar attacks and practical countermeasures have been discussed in [5].

References

[1] K. Aoki et al., "Camellia: a 128-bit block cipher suitable for multiple platforms - design and analysis", SAC 2000, LNCS 2012, pp. 39–56, Springer-Verlag, 2001.

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

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

[4] S. Mangard, "A Simple Power-Analysis (SPA) attack on implementations of the AES key expansion", ICISC 2002, LNCS 2587, pp. 343–358, Springer-Verlag, 2002.

[5] L. Xiao, *Implementation Analysis of Block Cipher Components and Structures.* PhD thesis, Memorial University of Newfoundland, 2003.

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