ISSC 2007, Derry. September 13–14

Correlation Power Analysis of Large Word Sizes

Michael Tunstall[†], Neil Hanley[†], Robert P. McEvoy[†], Claire Whelan[†], Colin C. Murphy[†] and William P. Marnane[‡]

Department of Electrical Engineering
University College Cork
Cork, IRELAND

[†]{miket,neilh,robertmce,clairew,cmurphy}@eleceng.ucc.ie [‡]l.marnane@ucc.ie

Abstract — **Side channel analysis can be used to derive secret/private keys used in naïve implementations of cryptographic algorithms. Such attacks typically involve demonstrating the relationship between the data being manipulated by a microprocessor or FPGA, and the instantaneous power consumption of the device. When statistical power attacks are applied to software implementations of block ciphers, the secret key can be derived in stages dictated by the size of the substitution tables (S-boxes) used. When statistical power attacks are applied to other algorithms that make use of 32-bit, or larger, variables, the complexity of a statistical power attack is dictated by the size of the machine word used by a microprocessor. This paper provides a method of reducing the complexity of statistical power analysis when attacking algorithms implemented on 32-bit microprocessors. It is also demonstrated that this method can be applied to hardware implementations, as found in coprocessors used in smart cards.**

Keywords — **Side Channel Analysis, Hardware DES Implementations, Smart Cards.**

I Introduction

Side Channel Analysis (SCA) is a type of cryptanalytic attack, which focuses on exploiting the implementation or some measurable non-mathematical property of a cryptographic system, such as the power consumption [1] or electromagnetic emanations [2]. SCA has advanced immeasurably since its breakthrough into the security community almost a decade ago. SCA attacks, such as Differential Power Analysis (DPA) [1], which exploits the instantaneous power consumption of a device, have proved to be very powerful and very difficult to defend against.

In particular, SCA has been associated with smart cards, where implementation characteristics are easy to monitor, measure and exploit. Two models have been proposed to describe the power consumption of such devices, namely the Hamming weight and Hamming distance models [3]. The information derived from these models is subsequently used in combination with statistical techniques to attack a target cryptographic device. These models are ultimately based on the word size of the underlying microprocessor, where the word size of the microprocessor is one of the main contributory factors to the efficiency and feasibility of the attack. In older technologies, where 8 or 16-bit processors were prevalent, the Hamming weight/distance and associated attacks could be executed relatively efficiently. However, the current trend in smart card technology is towards the use of 32-bit processors, making such attacks problematic due to a high computational intensity and the amount of acquisitions required.

The idea of statistically treating power analysis traces was first presented to the cryptographic community in [1], and is referred to as Differential Power Analysis (DPA). This involved predicting one bit of the output of a substitution table (S-box) used in block ciphers, in order to determine the bits of the secret key that were used to generate the predicted bit. Power consumption traces are divided into two sets based on the value of this bit and two average power consumption traces are created. A DPA trace is then formed by computing the pointwise difference between the two traces. Where the analysed bit is guessed correctly there will be a significant difference between the two average traces, and these differences are often referred to as DPA peaks.

The typical example used to illustrate an attack based on DPA is that of forming hypotheses on one bit of the output of an S-box in the first round of DES [4], this is then used to derive six bits of the secret key. Attacking a 32-bit word using DPA is problematic because of the amount of noise induced by the other 31 ignored bits in the register at the time of attack. This can mean a great deal of acquisitions are required in order to produce a significant peak. Also, small desynchronisation effects are, typically, amplified by taking numerous acquisitions, which can mean that acquiring more data does not always provide an efficient means of overcoming this noise. DPA is also subject to the "ghost peak" problem [3], which can lead to false positives on the key hypotheses.

An alternative to DPA was published in [3], called Correlation Power Analysis (CPA), where hypotheses on key bits are confirmed in a similar manner to DPA. However, an attacker verifies hypotheses by calculating the correlation between the instantaneous power consumption and the predicted output of an S-box. This provides a method of deriving information on a certain number of bits of a secret key used in a block cipher, that is less prone to validate false key hypotheses. The best results are achieved when the entire contents of the machine word being manipulated by a microprocessor are predicted. In this scenario the predicted power consumption matches the power consumption of the entire register. For example, if CPA is applied to an implementation of DES on an 8-bit chip, an attacker will be required to predict all eight bits of the register used to contain the output of each S-box. This is relevant as, in embedded software, S-boxes are typically stored in a compressed format (i.e. two S-box entries per byte in memory). However, for large word sizes, such as 32 bits, the amount of CPA traces required is likely to be computationally infeasible.

This paper proposes an optimisation for the computation of CPA traces for 32-bit platforms to derive information on the secret/private key used. It follows an extend and prune process somewhat similar to that explained in [5]. This optimisation allows the rapid calculation of CPA traces compared to conventional methods and also allows verification of hypotheses after linear functions, or non-linear functions calculated in parallel.

This paper is organised as follows. In Section II the power consumption models that have been used to derive statistical attacks are introduced. CPA is described in Section III, and Section IV describes how our proposed method makes use of the partial correlation in a similar manner. In Section V a case study is presented were the proposed method is applied to a DES coprocessor on a smart card. Finally conclusions are drawn in Section VI.

II Power Consumption Models

There are two main models for relating the instantaneous power consumption and the data being manipulated. These are the Hamming weight model and the Hamming distance model. These models are described below.

a) The Hamming Weight Model

The simplest model, initially proposed in [1, 6], involves an affine relationship between the power consumption and the Hamming weight of the data being manipulated at a given point in time. This is largely caused by the bus on the microprocessor, that consumes a large amount of power compared to any other single feature on the chip. This can be expressed as,

$$Y = aH(X) + b \tag{1}$$

where Y is the power consumption, X is the value of the data being manipulated, and H is a function that calculates the Hamming weight. The variable b includes the acquisition noise and the variation from one clock cycle to another, as the commands being executed by the CPU change.

This can be justified by observing that the tracks of the bus will either be zero (and consume no power) or one (and will therefore consume power). As each track will consume the same amount of power, the power is therefore proportional to the number of tracks with current flowing in them. As all the information being processed by the chip is carried across the bus, this relationship can be exploited when secret information is sent across the bus.

b) The Hamming Distance Model

Another model was proposed in [3] that generalises the Hamming weight model. This model is a similar model to the Hamming weight model but with an addition of another variable, P. This model is:

$$Y = aH(P \oplus X) + b \tag{2}$$

The notation is the same as described in the previous model, and the value P represents some previous state. The justification for this model is that the Hamming weight model is only valid if the bus is set to zero between each value sent. If this value is not set to zero then there is a transition from state P to state X, so the change in value of the bus can be modelled by an XOR between the two values.

In embedded software a command fetched by the CPU will often consist of several opcodes: usually an instruction followed by the data being manipulated. In this case, X is the data being manipulated and P is the instruction opcode. If the data is being sent over a separate data bus, P will either be previous data or zero, depending on how the chip manages values on the data bus, i.e. if the bus is set to zero when it is not in use or if it maintains its previous state.

When attacking a hardware implementation P needs to be predicted taking into account the design under attack. The previous state may be variable or constant depending on the design. This can be problematic if the design is not known, but can also provide a means of verifying hypotheses concerning the design of a given implementation.

III Correlation Power Analysis

An attacker using CPA will acquire a set of N power consumption traces while a given algorithm is being computed (w_i for $1 \leq i \leq N$), and attempt to predict the Hamming weight of the computer word being manipulated at a chosen point in time for each acquired trace (h_i for $1 \leq i \leq N$). The correlation between these predictions H and the instantaneous power consumption of the set of acquired traces W, i.e.

$$\rho_{WH} = \frac{\operatorname{cov}(W, H)}{\sigma_W \sigma_H}, \tag{3}$$

can be calculated to deduce where in the traces the chosen point in time appears. This involves generating a CPA trace that represents the correlation between H and W at each point in the acquired power traces.

The series of data h_i for $1 \leq i \leq N$ is predicted using one of the models proposed in Section II. When the Hamming distance model is used, it is also necessary to calculate the correlation coefficient for every possible value for the unknown previous state P, and the largest correlation coefficient should give the value of P. This will involve generating a CPA trace for each of the possible values for P. On an architecture with a small word size, e.g. 8-bit, this is straightforward. However, on an architecture that uses a relatively large word size, e.g. 32-bit, this is computationally prohibitive.

When CPA is used to attack a block cipher, an attacker forms a hypothesis about the result of a given function, e.g. an S-box, for each of N acquisitions where the following structure exists:

$$I = S(M \oplus K) \tag{4}$$

i.e. a part of the message block M is XORed with part of the key K, and then passed through an S-box (the function S) to produce an intermediate state I. An attacker will try and predict the series h_i for $1 \leq i \leq N$ for each possible value of K (or the fraction being manipulated at the chosen point in time). An attacker is therefore obliged to generate a CPA trace for each value of K. This subject is given a more rigorous treatment in [3].

In the case of secret key cryptographic algorithms that are designed to run quickly on 32-bit platforms this could pose a problem due to the amount of computation required. For example, RC6 [7] uses 32-bit modular multiplication in its round function. An attacker would be obliged to generate 2^{32} CPA traces to derive a portion of the secret key. If the Hamming distance model is used a further 2^{32} CPA traces are required per hypothesis, for a total of 2^{64} CPA traces. The same problem presents itself if an attacker tries to attack a HMAC based around SHA or MD5 [8] as these also use 32-bit functions. This problem is amplified on FPGA and ASIC implementations of block ciphers, as designers will attempt parallelise as much as possible. An attacker is obliged to model the design to be able to predict everything that is occurring at a given point in time in order to produce an efficient attack. The method proposed in this paper seeks to reduce the amount of computation required to make attacking hardware designs and 32-bit chips feasible.

IV Using the Partial Correlation

In [3] it is pointed out that if the correlation coefficient of l independent bits amongst m is calculated, a partial correlation still exists and can be predicted as a function of the coefficient that would be generated if all the bits of m were included. This is given as:

$$\rho_{WH_{l/m}} = \rho_{WH} \sqrt{\frac{l}{m}} \tag{5}$$

where l bits from an m-bit word are known.

This can be used to provide a method to attack implementations on processors with large word sizes. An attacker can correlate with a certain number of bits in the word being manipulated by a processor; the largest correlation coefficient should correspond to the correct hypothesis for these bits. A further selection of bits can then be analysed, taking into account the bits that have already been derived. The largest correlation coefficient will be greater than the previous stage, as more bits are known. This allows the second selection of bits to be deduced, and also validates the bits derived during the first stage of the attack.

For example, if we consider the multiplication of two 32-bit values (A, B) in a 32-bit ALU, i.e. the multiplication takes place modulo 2^{32}, where A is known and B is unknown and it is known that the Hamming weight model applies. Let the 32-bit word output be written as four bytes `MSB`, `MMSB`, `MLSB`, and `LSB`. An attacker can attempt to derive the value of each byte of B in succession.

CPA traces can be generated using just the values for `LSB`, as `LSB` can be predicted for the 2^8 possibilities for the 8 least significant bits of B. This will produce a set of CPA traces where the trace corresponding to the correct hypothesis of B will contain a correlation coefficient of $\sqrt{\frac{8}{32}} = 0.5$ the correlation produced if all the bits were predicted.

An attacker can then generate CPA traces to determine the 16 least significant bits of B with the same amount of computation, as the 8 least significant bits of B will have been determined. This will produce a correlation coefficient of $\sqrt{\frac{16}{32}} = 0.707$ the correlation produced if all the bits were predicted. This provides a further 8 bits of B, and validates the 8 bits of B derived in the first stage of the attack.

This can be continued for `MMSB` and `MSB` to derive the value of B, and requires the generation of $4 \times 2^8 = 2^{10}$ CPA traces. This is considerably more efficient than forming hypotheses directly on B, which would require the generation of 2^{32} CPA traces.

In practice, it may be advantageous to take a certain number of hypotheses from each stage of the analysis rather than just the most likely hypothesis. A partial correlation does not have the same distinction between a correct hypothesis and an incorrect hypotheses that one would expect when calculating the correlation using all m bits. This makes the optimisation more robust as it allows for errors in interpreting results (e.g. possibly caused by noise during the acquisition process). In the above example, if four hypotheses from each stage of the attack are retained, rather than one, then this would raise the number of CPA traces that would need to be generated, in the above example, to $2^8 + 3(4 \times 2^8) \approx 2^{12}$.

In the case of the Hamming distance model the same optimisation applies, although it is more practical to consider the word in sections of 4 bits. Otherwise, generating the number of CPA traces that are required to validate hypotheses at each stage of the attack can become prohibitively time consuming. In the example given above, there would be 64 unknown bits (B and some previous state) that would normally require the generation of 2^{64} CPA traces.

This optimisation will work for functions where the operation being attacked can be broken down into smaller units, as in the above example. This will not be possible in functions where each bit of the output is dependent on every bit of the input.

V Case Study: DES Coprocessor

Hardware implementations of block ciphers, such as DES, are often included in smart card chips to improve performance. Hardware implementations of block ciphers are able to compute an algorithm significantly faster than embedded software. Such hardware implementations of block ciphers have typically been viewed as difficult to attack using side channel analysis. This section describes how the method described above can be applied to power consumption traces acquired during the computation of a hardware implementation of DES [4].

The computation of DES can be considered as a transformation of two 32-bit variables (L_0, R_0), i.e. the message block, through sixteen iterations of a round function to produce a ciphertext block (L_{16}, R_{16}). The round function can be written as:

$$\begin{aligned} R_n &= S(R_{n-1} \oplus K_n) \oplus L_{n-1} \\ L_n &= R_{n-1} \end{aligned} \tag{6}$$

where S is the S-box function. There are some bitwise permutations but these are not relevant to this description, i.e. they render an implementation more complex but principle of the attack is unaffected. The eight different S-boxes are applied in each round to sets of six bits, where each S-box reduces a 6-bit input to a 4-bit output. The subkeys K_n, for $1 \leq n \leq 16$, are each 48 bits generated from the 56-bit key, by permuting the bits of the initial key.

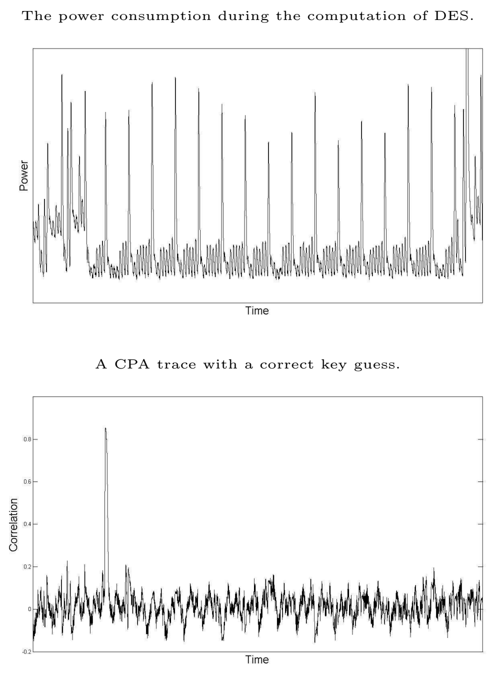

Fig. 1: Power consumption during the computation of a DES in hardware (upper), and a CPA trace computed using the correct hypothesis for the first subkey (lower).

When designing hardware implementations of block ciphers it is natural to try and do as much as possible in parallel, i.e. to calculate all eight S-boxes in parallel each round. This is especially true in smart cards since the standard clock applied is typically quite slow (usually 3.57 MHz). Some smart cards do include internal clocks, but these rarely exceed 10 MHz. This has traditionally been viewed as a means of preventing side channel

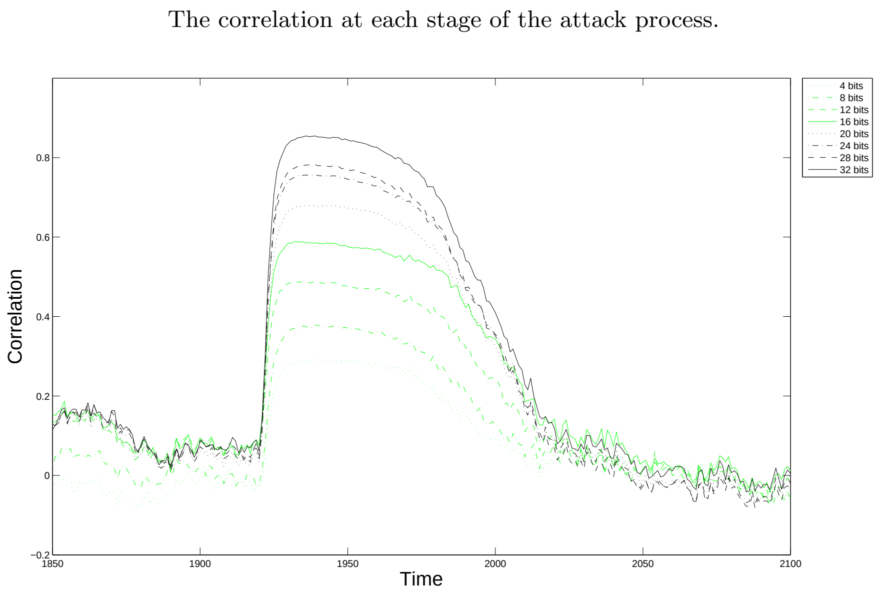

Fig. 2: The evolution of the partial correlation as bits of the key are determined. Note that this is a magnification of the peak shown in Figure 1.

analysis as an attacker needs to be able to predict the 32-bit output of all the S-boxes. An application of the attack method proposed in Section IV to a DES coprocessor is described below.

To conduct this case study a series of 200 traces, with the corresponding message and ciphertext blocks, were provided by Gemalto. These traces were average traces produced from an initial set of 2000 traces, i.e. the same message block was applied ten times and an average trace was formed from the ten separate acquisitions. An example trace is shown in Figure 1, where the 16 rounds of DES are clearly visible in the power consumption. The secret key and the details of the design were not provided.

The output T of a given S-box was modelled using the formula,

$$\begin{aligned} T &= S(K_1, R_0) \oplus L_0 \oplus R_0 \\ &= R_1 \oplus R_0 \ . \end{aligned} \tag{7}$$

This corresponds to the Hamming distance model described in Section II but with P being the previous state of the register used to store R_i for $0 \leq i \leq 16$.

The partial correlation for the 4-bit output of the first S-box was calculated using CPA for each of the possible 6 key bits that affect these bits. The four key hypotheses with the largest correlation were then included in the next calculation, which attempted to correlate the 8-bit output of the first two S-boxes. This would normally involve generating 2^{12} CPA traces, but, as four hypotheses were included from the first partial correlation, this involved the generation of $2^6 \times 4 = 2^8$ CPA traces. The four hypotheses corresponding to the largest partial correlations were then included in an attempt to correlate with the first three S-boxes. This process was continued until CPA traces could be generated on all 32 bits, which gave information of the 48-bit subkey used in the first round of DES. The whole process required the generation of $2^6 + 7(2^6 \times 4) = 2^{11}$ CPA traces, which is significantly fewer than the 2^{48} traces that would be required to test every possible hypothesis for the first subkey. The correlation coefficient at each stage in this process is shown in Figure 2, which only shows the peak in the correlation. It is interesting to note that the increase in correlation at each stage is not constant. We are currently unable to explain why this should occur, and this is left as an open question. The correlation with all 32 bits is shown in Figure 1 and shows the CPA trace for the whole computation of DES. The peak in the correlation corresponds to the peak in the power consumption at the end of the computation of the first round of DES, the CPA trace demonstrates that this is where the result of the first round is written back to the register containing R_0.

An attempt was made to generate DPA traces based on Equation 7 for comparative purposes, but it was not possible to derive any bits of the secret key used in the DES implementation. We assume that this is because of the noise introduced by the ignored bits in the hardware DES.

VI Conclusion

This paper proposes a method of applying CPA to implementations of cryptographic algorithms on architectures that use large word sizes. The example given in Section IV discusses applying the proposed method to a 32-bit architecture, but the same method should apply to architectures that use a larger word size.

As can be seen in Figure 1, CPA can be applied to a 32-bit register to retrieve a DES key from power traces whilst only calculating $2^6+7(2^6 \times 4) = 2^{11}$ CPA traces. This represents a significant computational saving when compared to the classical method of applying CPA. Generating CPA curves to produce an equivalent result without using the method proposed in this paper would require generating 2^{48} CPA traces, which is computationally infeasible. An alternative attack strategy would be to use the partial correlation to generate hypotheses independently on groups of bits. This reduces the number of CPA traces required to attack a hardware implementation of a secret key algorithm. However, the distinction between the correct hypothesis and the incorrect hypotheses is very small. The advantage of the proposed method is that each stage validates all the information that has already been acquired.

The method proposed in this paper should also apply to multiplication algorithms, such as Montgomery or Quisquater multiplication [9, 10]. This would allow CPA to be applied to public key cryptographic algorithms by calculating hypotheses on the intermediate states of the multiplication algorithms. This provides an attack method that allows CPA to be applied to a wider range of algorithms than the attack methods suggested for public key cryptographic algorithms, such as RSA [11]. CPA can be applied to a naïve implementation of RSA, as the intermediate states can take a limited number of values [12] and can therefore be exhaustively tested. The method proposed in this paper is also likely to be more efficient than previously published experiments applying DPA to RSA [13].

Acknowledgements

The authors would like to thank Francis Olivier from Gemalto for providing the power consumption traces used in the case study described in Section V.

This work was supported in part by the Embark Initiative, operated by the Irish Research Council for Science, Engineering and Technology (IRCSET). Also the support of the Informatics Commercialisation initiative of Enterprise Ireland is gratefully acknowledged.

References

[1] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. J. Wiener, editor, *Advances in Cryptology — CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer-Verlag, 1999.

[2] K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic analysis: Concrete results. In C. K. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer-Verlag, 2001.

[3] E. Brier, C. Clavier, and F. Olivier. Correlation power analysis with a leakage model. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems — CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer-Verlag, 2004.

[4] NIST. *Data Encryption Standard (DES) (FIPS–46-3).* National Institute of Standards and Technology, 1999.

[5] S. Chari, J. Rao, and P. Rohatgi. Template attacks. In B. S. Kaliski, Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer-Verlag, 2002.

[6] T. S. Messerges, E. A. Dabbish, and R. H. Sloan. Investigations of power analysis attacks on smartcards. In *USENIX Workshop on Smartcard Technology*, pages 151–161, 1998.

[7] R. L. Rivest, M. J. B. Robshaw, R. Sidney, and Y. L. Yin. The RC6 block cipher. algorithm specification version 1.1, 1998. available at `ftp://ftp.rsasecurity.com/pub/rsalabs/rc6/rc6v11.pdf`.

[8] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.

[9] P. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44:519–521, 1985.

[10] J.-J. Quisquater. Encoding system according to the so-called RSA method, by means of a microcontroller and arrangement implementing this system. U.S. Patent Number 5,166,978, 1992. Also presented at the rump session of EUROCRYPT '90.

[11] R. Rivest, A. Shamir, and L. M. Adleman. Method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[12] J.-S. Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In C. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 99*, volume 1717 of *Lecture Notes in Computer Science*, pages 292–302. Springer-Verlag, 1999.

[13] T. S. Messerges, E. A. Dabbish, and R. H. Sloan. Power analysis attacks of modular exponentiation in smartcards. In Ç. K. Koç and C. Paar, editors, *Cryptogaphic Hardware and Embedded Systems — CHES '99*, volume 1717 of *Lecture Notes in Computer Science*, pages 144–157. Springer-Verlag, 1999.