Power Matching to Protect Smart Cards from Power Attacks

Craig L. Munsee
Department of Electrical and Computer Engineering,
Oregon State University, Corvallis, Oregon 97331 - USA.
E-mail: munsee@engr.orst.edu

Abstract— **Smart card security has been greatly compromised due to passive power attacks. These attacks are easy to implement and are virtually unseen to the card owner. Many countermeasures have been tried and some have succeeded in increasing the difficulty of obtaining the secret key that a smart card holds. Just as quickly as a countermeasure is introduced another way to analyze the data is presented. This is all due to the presence of a faulty algorithm that only focuses on performance rather than security and power consumption. Protecting smart cards from power attacks is a daunting task. The key to protecting smart cards lies in the algorithm and the consumption of power being the same for all smart cards. With security in mind a new algorithm will be introduced that will make the power signature the same for all smart cards. This new method is called power matching.**

Keywords— **Smart cards, Power attacks, Simple Power Analysis, SPA, Differential Power Analysis, DPA, Inferential Power Analysis, IPA, Power matching.**

I. Introduction

Security is a concern to everyone, especially the security of his or her money. This is the whole point behind encryption and the use of smart cards. Smart cards are intended to make transactions more secure and prevent an attacker from masquerading as someone else. Since Smart cards use encryption you would think that this ensures the protection needed, but it doesn't. The design of a cryptographic algorithm is designed with its security in mind, but the physical implementation of the circuit is designed with efficiency in mind. Security aspects, as compared to efficient, simplicity, or power consumption criteria, do still only play a marginal role in circuit design [3]. Because of this smart cards are susceptible to attacks.

There are two different types of attacks. We are most familiar with an active attack where the card is usually stolen and put through a series of tests that would reveal the secret key the card contains. Active attacks leave visible signs of an attack, allowing the owner of the card to report the attack. The second type of attack is a passive attack that goes unseen to the owner of the card, giving the attacker time to take advantage of the newly acquired information. The most powerful of the passive attacks is the power attack. The attacker monitors the power that is used by a smart card, and uses this information to deduce the key.

Since smart cards are susceptible to these power attacks countermeasures have been introduced, but it seems as soon as a countermeasure is introduced a new way to use the power signature is introduced. This paper will discuss these security attacks and some of the countermeasures that have been implemented. This paper will then address the security problem and will design a circuit with security in mind that will make the power signature the same for all smart cards. This method is known as the power matching method.

II. Active attacks

Active attacks are usually applied to stolen smart cards, because they leave visible signs of tampering and sometimes destroy the card. Active attacks include fault attacks, probing attacks, and chip microsurgery. These attacks require large amounts of time, sophisticated equipment, and knowledge of how the chip works. There is little defense against active attacks, because with enough time and resources any security system can be defeated. Since these attacks require large amounts of time and resources and also leave visible signs of tampering, these attacks are rarely used. Smart cards that show signs of tampering are immediately reported by the owner of the card and make it difficult for the attacker to use the newly acquired secret information. This is why attackers prefer to use passive attacks.

III. Passive attacks

Passive attacks are applied without the owner of the smart card knowing that they are being attacked. These attacks are not destructive, and do not leave visible signs of tampering. Passive attacks include timing attacks, glitch attacks, and power analysis [2]. They require little sophistication and minimal investment, and can be carried out against a large number of individual cards by a small number of rogue card readers [2]. Timing attacks and glitch attacks pose little risk to well designed smart card applications, since it is easy to protect the software and hardware elements of smart cards against them [2]. Power attacks are the most successful of the passive attacks, because they are easy to implement and very difficult to avoid. It is based on the observation that the detailed power consumption curve of a typical smart card (which describes how the externally supplied current changes over time) contains a huge amount of information about its operation [2]. These power attacks are in a group known as leakage attacks. These attacks exploit the fact that a hardware device can sometimes leak information when running a cryptographic algorithm

[5]. One source of leaked information is the time-varying power consumption of a device executing a cryptographic algorithm [5]. Power attacks are discussed in greater detail below.

IV. Power Attacks

Power attacks are executed by observing detailed power consumption of a typical smart card transaction. When gates switch on and off, a sequence of events are displayed in the power fluctuations. This information is much like a fingerprint. It will be distinct for a specific secret key. For example, the power consumption profiles of addition and multiplication operations are completely different, the power consumed by writing 0..0 and 1..1 to memory are noticeably different, and it is possible to visually extract the secret key of an RSA operation by determining which parts look like a modular squaring and which parts look like modular multiplication [2]. There are three types of power attacks. These types are Simple Power Analysis (SPA), Differential Power Analysis (DPA), and Inferential Power Analysis (IPA). Some implementations are resistant to SPA attacks, but not DPA attacks. Others are resistant to DPA, but not SPA attacks. Each attack has its strengths and weaknesses.

V. Simple Power Analysis (SPA)

In SPA the attacker studies a single power consumption curve to obtain statistical information about the identity of the instructions and the Hamming weight of the data words read from or written into memory at any given clock cycle [2]. The measured values must be sampled at adequate instants, whose timing must be known by the attacker. In SPA attacks, the aim is essentially to guess - from the values of the consumption - which particular instruction is being computed at a certain time and with which input or output, and then to use this information to deduce some part of the secret [9]. Figure 1 shows the electric consumption of a chip, measured during a DES computation on a real smart card [9].

The strength of SPA over DPA is that it only requires the power consumption characteristics of one execution of the algorithm, where as DPA requires multiple power consumption curves recorded from different executions with different inputs. The weakness of SPA is that random noise can sometimes prevent an attack.

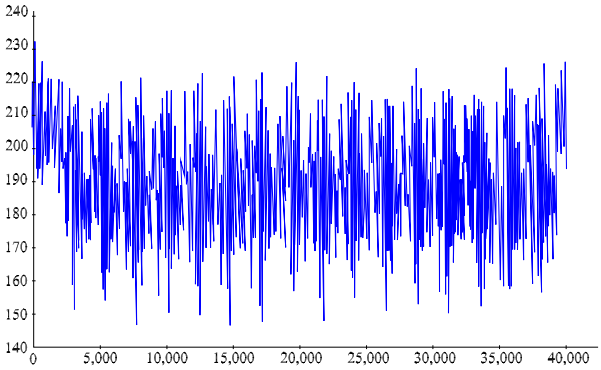

Figure 1: Electric consumption measured on the 16 rounds of a DES computation [9].

VI. Differential Power Analysis (DPA)

In DPA the attacker studies multiple power consumption curves recorded from different executions with different inputs, and uses statistical differences between particular subsets of executions to find in an automated way particular key bits [2].

There are two types of DPA attacks, first-order DPA attacks and second-order DPA attacks. During a first-order DPA attack, the attacker monitors power consumption signals and calculates the individual statistical properties of the signals at each sample time [5]. In a higher-order DPA attack, the attacker calculates joint statistical properties of the power consumption at multiple sample times within the power signals [5]. Second-order DPA attacks require more complex analysis, increased memory and processing requirements and an increased number of power consumption measurements [5].

The initial focus was on symmetric cryptosystems such as DES and AES candidates, but public-key cryptosystems have since been shown to be also vulnerable to the DPA attacks [4].

The strengths of DPA over SPA are that the attacker doesn't need to know the implementation to perform the attack, and that the averaging process reduces noise that is an obstacle to SPA. The weakness of DPA is that it requires time.

VII. Inferential Power Analysis (IPA)

An IPA attack is characterized by two stages, the first a lengthy profiling stage, and the second a simpler key extraction stage [10]. The goal of the profiling stage is to locate and identify the key bits as they are used during the computation [10]. The profiling stage only needs to be performed once for each software implementation. The key extraction stage can be done quickly. To extract the key from a new instance of the same implementation, we take a single power trace, chop it into rounds, and measure the power consumed at the locations specified in the key location table [10]. Using our knowledge of the key bit power distribution, which we obtained during the profiling stage, we can tell whether the key bit is a 0 or a 1 [10].

There are several strengths to IPA that would make it effective in situations that DPA would be difficult. This is the reason that IPA has been introduced.

The first is when attacking a DES algorithm the plaintext or the ciphertext would be needed for DPA. Neither the plaintext nor the ciphertext are required to mount an IPA attack. Also, a DPA attack is restricted to points in the algorithm where the plaintext (or ciphertext) interacts directly with the key; this is because the differential traces are based on a "selection function" that predicts a bit value based on a small number of plaintext bits and a small number of key bits [10]. This means that when attacking a DES algorithm, DPA would be restricted to the beginning or the

end of the cryptographic algorithm. IPA can look at any part of the algorithm for its attack.

The second is the ability of IPA to do fast key extraction after a single lengthy profiling stage. A protocol may disable a card after only a small number of operations. This could block DPA from acquiring the large number of traces needed to acquire the key. IPA could perform the profiling stages on another card and only a small number of traces for the card being attacked.

For more information on IPA see, IPA: A New Class of Power Attacks by Paul N. Fahn and Peter K. Pearson [10].

VIII. Countermeasures

Three countermeasures have been suggested to defend against power attacks. These countermeasures are:

1.) Introducing random timing shifts, so that the computed means do not correspond any longer to the consumption of the same instruction [4]. The crucial point consists here in performing those shifts so that they cannot be easily eliminated by statistical treatment of the consumption curves [4].

2.) Replace some of the critical instructions (in particular the basic assembler instructions involving writings in the carry, readings of data from an array, etc) by assembler instructions whose "consumption signature" is difficult to analyze [4].

3.) For a given algorithm, giving an explicit way of computing it, so that DPA is provably unefficient on the obtained implementation [4].

This paper doesn't intend to look at all countermeasures, but will look at some of the more recent countermeasures that have been introduced. These countermeasures are discussed in greater detail below.

IX. Duplication Method

Goubin et al. proposed a strategy, called the "duplication method", to protect the DES algorithm from first-order DPA attacks [5]. Their countermeasure works by splitting secret data into two random halves and operating on each half separately [5]. Such an approach causes the power consumption signals to be randomized, thus thwarting DPA [5]. As a generalization, Chari et al. suggested a countermeasure that splits the data into k shares [5]. They proved that the amount of analysis needed to attack such a scheme increases exponentially with respect to k [5].

X. Current Sensor

The current sensor method was introduced to equalize the current used by a smart card. A current sensor is used to actively equalize the current used by controlling an additional current sink. This method was found to be ineffective since the changes in the power supply curves were so rapid that any compensation curve technique is likely to lag behind and leave many power spikes clearly visible [2].

XI. Air Gap Method (Two Capacitors)

The air gap method was introduced to disassociate the correlation between the power supplied to the card from the power consumed by the card. This is implemented with two capacitors. The behavior of the capacitors is defined by a simple switch control unit and four power transistors, which are added to the smart card chip [2]. The preferred cyclic sequence of action is [2]:

1.) The first capacitor is disconnected from the external power.
2.) The first capacitor is connected to the chip.
3.) The second capacitor is disconnected from the chip.
4.) The second capacitor is connected to the external power.

With this behavior the smart card chip is always powered by at least one capacitor, but the external power supply is never connected directly to the internal chip [2]. These two capacitors are small and could easily fit inside the smart card casing.

The only disadvantage of the capacitor approach is that it can supply power to the chip only for several hundred clock cycles before its voltage becomes too low, and in each clock cycle the supplied voltage drops by about 0.01 volts [2].

For more information on the air gap method see, Protecting Smart Cards from Passive Power Analysis with Detached Power Supplies by Adi Shamir [2].

XII. Boolean and Arithmetic Masking

Thomas Messerges developed a general countermeasure, consisting in masking all the inputs and outputs of each elementary operations used by the microprocessor [4]. This masking strategy is possible if all the fundamental operations used in a given algorithm can be rewritten with masked input data, giving masked output data [4]. This is easily seen to be the case for the DES algorithm, because a single masking (using the XOR operation) can be used throughout the computation of the 16 rounds [4]. For RSA, a masking using the multiplication operation in the multiplicative group modulo n is also sufficient [4]. However, for algorithms that combine Boolean and arithmetic functions, two different kinds of masking have to be used [4]. For these algorithms a method to convert between Boolean and arithmetic masking was introduced. The masking method is shown below.

Boolean masking: $x' = x \oplus r$

Arithmetic masking: $x' = (x - r) \bmod 2^k$

Jean-Sebastien Coron and Louis Goubin [4] found a weakness in this countermeasure using DPA. Even though they didn't perform the experiments to validate the attack they laid the groundwork for future research.

For more information on Boolean and Arithmetic Masking see, On Boolean and Arithmetic Masking against Differential Power Analysis by Jean-Sebastien Coron and Louis Goubin [4].

XIII. Random Register Renaming

Random register renaming tries to take out the correlation between the physical measurements taken at different points during the computation and the internal state of the processing device, which is related to the secret key [1]. For example, when data is loaded from memory, the memory bus will have to carry the value of the data, which will take a certain amount of power depending on the data value [1]. Since the load instruction always happens at the same time one can produce correlations between various runs of the application, eventually giving away the secret of the smart card [1].

D. May, H. Muller and N.P. Smart propose a method for introducing highly aggressive randomized execution into a conventional processor [1]. They argue that this produces a great deal of temporal misalignment of traces, which can help defeat DPA [1]. The methodology is to take standard techniques from the design of super-scalar architectures and replace parallel execution with random execution [1]. They call this new processor architecture NDISC for Non-Deterministic Instruction Stream Computer [1]. This defense is trying to introduce timing shifts, and hard to analyze algorithms.

One could argue that the register renaming in a superscalar architecture is not random, and will produce the same sequence on the same set of instructions each time it is reset and executed. The NDISC uses a process they call "Instruction De-scheduling" to randomize the sequence. This is done by identifying instructions that can be executed in parallel, and instead of running these instructions in parallel, they use this information to run the instructions out of order. This out of order instruction scheduling makes a DPA attack much more difficult if not impractical.

For more information on random register renaming see, Random Register Renaming to Foil DPA by D. May, H.L. Muller, and N.P. Smart [1].

XIV. Random Process Interrupts

One of the most common countermeasures against DPA is the introduction of random process interrupts (RPIs) [6]. Instead of executing all the operations sequentially, the CPU interleaves the code's execution with that of dummy instructions so that corresponding operation cycles do not match because of time shifts [6]. This has the effect of smearing the peaks across the differential trace due to a de-synchronization effect, known in digital signal processing under the name of incoherent averaging [6]. The RPIs don't make DPA attacks impossible, since the time shift can be considered as added noise, but does make the attack considerably more difficult. Christophe Clavier, Jean-Sbastien Coron, and Nora Dabbous introduced a sliding window to DPA making this attack more feasible [6]. For more information on attacking RPIs see, Differential Power Analysis in the Presence of Hardware Countermeasures by Christophe Clavier, Jean-Sbastien Coron, and Nora Dabbous [6].

XV. Power Matching Method

The power matching method has been developed to protect the square-and-multiply algorithm for RSA. The square-and-multiply algorithm has been implemented two ways, and are shown below.

Modular exponentiation implementations [8]:

```
exp1(M, e, N)
{ R := M
  for (i := n − 2 downto 0)
  { R := R^2 mod N
    if (ith bit of e is a 1)
      R := R · M mod N }
return R }

exp2(M, e, N)
{ R := 1
  S := M
  for (i := 0 to n − 1)
  { if (ith bit of e is a 1)
      R := R · S mod N
    S := S^2 mod N}
return R }
```

The main problem with both algorithms is that the outcome of the "if statement" might be observable in the power signal [8]. This would directly enable the attacker to learn every bit of the secret exponent [8]. At first glance the simple solution would be to always multiply whether it is a 1 or a 0, and save the correct value based on the 1 or the 0. This increases the time and may not be enough to hide the save operation.

The power dissipated in CMOS cells such as logic gates, flip-flops, or latches mainly depends on changes of components' states rather than on the states themselves [3]. These are the things that make the power signature different for each smart card. The only way to completely protect a smart card is to match every bit with its opposite at every operation. This doubles the area needed for the computation because for every value its opposite needs to be calculated along side it at the same instance. Granted this isn't the most efficient way to compute the value, but we are putting security first and efficiency second. The design is discussed in greater detail below.

XVI. Power Matching Design

The algorithm that was used is essentially the same for exp1. We modified it to always perform the R = R*M mod N multiplication and made the squaring step a multiplication operation. The new algorithm is shown below.

```
exp1(M, e, N)
{ R := M
  for (i := n − 2 downto 0)
  { R := R · R mod N
  { S := R · M mod N
    if (ith bit of e is a 1)
      R := S }
return R }
```

The main difference is the design of the chip. For every set of bits its opposite is present, causing the Hamming weight to always be the same. The squaring procedure has been changed to a multiplication procedure, making it impossible to distinguish the difference between the squaring and the multiplication operations. Inside the multiplier, gates that would be an AND are matched with a NAND gate. Buffers are added to make the timing the same. A simple example is shown in Figure 2. Each value that is stored in memory is stored with each bit next to its opposite.

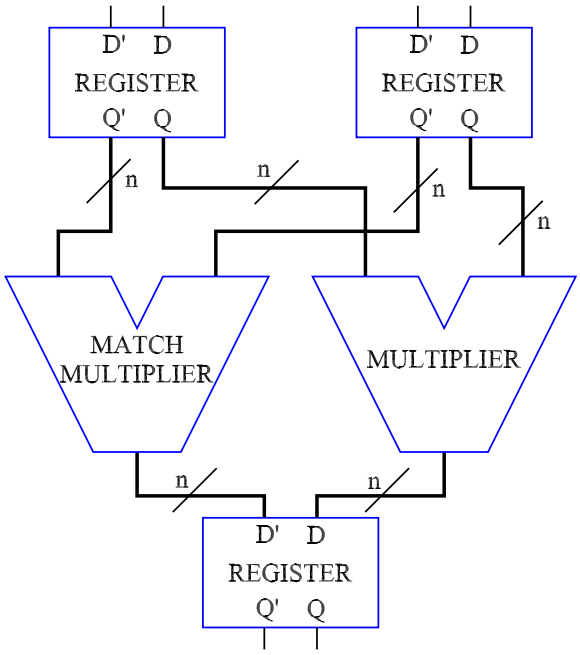

Figure 2: Simple multiplication scheme with opposite values imbedded.

XVII. Test Results

Using the power matching method we produced a prototype smart card and subjected it to SPA, DPA and IPA. With our test results we found that we were able to defend a smart card from all types of power attacks. We changed the secret key of the smart card several times and each time we were not able to distinguish the difference between cards with different secret keys. Table 3 shows different power attacks used against three smart cards. The first with no countermeasures, the second with noise introduced, and the third, using the Power Matching method.

	Normal	With Noise	Power Matching
SPA	Yes	No	No
DPA	Yes	Yes	No
IPA	Yes	Yes	No

Table 3: Results of Power Attacks Against the Power Matching Method

XVIII. Limitations

The limitation to this method is that the size of a chip on a smart card is limited by the International Organization for Standardization (ISO) standard 7816. This isn't sufficient room to make the computation and compute its opposite at the same time. Before this can be implemented in a commercial smart card a revision to this standard will need to be introduced.

XIX. Conclusion

The threat of power attacks to smart cards is a real concern. We have discussed the different types of power attacks and some of the countermeasures that have been implemented to protect smart cards. As you can see from above, there are some good countermeasures that have been introduced. Many of these countermeasures have still been defeated, because attackers have found new ways to process the power curves. The power matching method, introduced here, will make processing the power curves uninformative. Essentially the power matching method will make the power signature of every smart card the same. This will remove the power leakage venerability of smart cards.

References

[1] H.L. Muller D. May and N.P. Smart, "Random register renaming to foil dpa," in *Cryptographic Hardware and Embedded Systems — CHES 2001*, Ç. K. Koç and C. Paar, Eds. 2001, Lecture Notes in Computer Science, No. 2162, pp. 28–38, Springer, Berlin, Germany.

[2] Adi Shamir, "Protecting smart cards from passive power analysis with detached power supplies," in *Cryptographic Hardware and Embedded Systems - CHES 2000*, Ç. K. Koç and C. Paar, Eds. 2000, Lecture Notes in Computer Science, No. 1965, pp. 71–77, Springer, Berlin, Germany.

[3] Rita Mayer-Sommer, "Smartly analyzing the simplicity and the power of simple power analysis on smartcards," in *Cryptographic Hardware and Embedded Systems - CHES 2000*, Ç. K. Koç and C. Paar, Eds. 2000, Lecture Notes in Computer Science, No. 1965, pp. 78–92, Springer, Berlin, Germany.

[4] Jean-Sebastien Coron and Louis Goubin, "On boolean and arithmetic masking against differential power analysis," in *Cryptographic Hardware and Embedded Systems - CHES 2000*, Ç. K. Koç and C. Paar, Eds. 2000, Lecture Notes in Computer Science, No. 1965, pp. 231–237, Springer, Berlin, Germany.

[5] Thomas S. Messerges, "Using second-order power analysis to attack dpa resistant software," in *Cryptographic Hardware and Embedded Systems - CHES 2000*, Ç. K. Koç and C. Paar, Eds. 2000, Lecture Notes in Computer Science, No. 1965, pp. 238–251, Springer, Berlin, Germany.

[6] Jean-Sebastien Coron Christophe Clavier and Nora Dabbous, "Differential power analysis in the presence of hardware countermeasures," in *Cryptographic Hardware and Embedded Systems - CHES 2000*, Ç. K. Koç and C. Paar, Eds. 2000, Lecture Notes in Computer Science, No. 1965, pp. 252–263, Springer, Berlin, Germany.

[7] Pil Joong Lee Eun Jeong Lee and Yong Duk Kim, "How to Implement Cost-Effective and Secure Public Key Cryptosystems," in *Cryptographic Hardware and Embedded Systems - CHES 1999*, Ç. K. Koç and C. Paar, Eds. 1999, Lecture Notes in Computer Science No. 1717, pp. 73–79, Springer, Berlin, Germany.

[8] Ezzy A. Dabbish Thomas S. Messerges and Robert H. Sloan, "Power analysis attacks of modular exponentiation in smartcards," in *Cryptographic Hardware and Embedded Systems - CHES 1999*, Ç. K. Koç and C. Paar, Eds. 1999, Lecture Notes in Computer Science No. 1717, pp. 144–157, Springer, Berlin, Germany.

[9] Louis Goubin and Jacques Patarin, "Des and differential power analysis," in *Cryptographic Hardware and Embedded Systems -*

CHES 1999, Ç. K. Koç and C. Paar, Eds. 1999, Lecture Notes in Computer Science, No. 1717, pp. 158–172, Springer, Berlin, Germany.

[10] Paul N. Fahn and Peter K. Pearson, "Ipa: A new class of power attacks," in *Cryptographic Hardware and Embedded Systems - CHES 1999*, Ç. K. Koç and C. Paar, Eds. 1999, Lecture Notes in Computer Science No. 1717, pp. 173–186, Springer, Berlin, Germany.