A Theoretical Evaluation of some NESSIE Candidates regarding their Susceptibility towards Power Analysis Attacks

Elisabeth Oswald and Bart Preneel
Katholieke Universiteit Leuven, Dept. ESAT,
Kasteelpark Arenberg 10,
B-3001 Leuven-Heverlee, Belgium

October 4, 2002

Abstract

In this paper, we discuss the vulnerability of dedicated hardware implementations and software implementations on standard microprocessors of some NESSIE candidates. We assess their vulnerability against power attacks based on theoretical assumptions on their particular implementation. Furthermore, we investigate the possibilities to protect them against power attacks. Based on these investigations, we try to give a fair comparison regarding their suitability to be securely implemented with respect to power attacks.

1 Introduction and Motivation

For ciphers that become part of a security related standard a widespread range of use can be expected. This implies that those ciphers are implemented in many different applications which in turn implies that they are implemented on many different platforms.

Recent findings (see [KJJ99]) have shown that certain unskilled implementations of cryptosystems allow an attacker to derive the secret key with very low effort. Nowadays, as it has been proven that such side-channel attacks are a serious threat, the evaluation procedure for cryptographic algorithms needs to be re-thought. An algorithm which is strong with respect to conventional cryptanalytic attacks is useless if it cannot be implemented securely on a broad range of platforms.

Already during the AES process, the cryptographic community came to this conclusion. The paper of Daemen et al. [DR99] tries to compare the possibilities to implement the AES candidates resistant to power-analysis attacks. This paper tries to extend this first approach in the sense that we do not only compare some of the NESSIE candidates by the operations that need to be implemented, but also by the statistical and structural properties that are relevant for power analysis attacks of the ciphers themselves. This article only deals with some of the symmetric algorithms which were not discarded in the second phase of the NESSIE project.

The structure of this article is as follows. In section 1, we present the concepts of power-analysis attacks and motivate the kind of power-analysis attacks that we will base our analysis on. In section 2, we motivate the choice of the criteria that we will use to evaluate the algorithms. In section 3, we motivate the choice of the algorithms themselves and in section 4, we present the results of our evaluation. Section 5 summarizes the results and presents the comparison of the algorithms. We conclude this paper in section 6.

2 Power Analysis Attacks

All power-analysis attacks are based on the assumption that the instantaneous power consumption of an integrated circuit is dependent on the executed instructions and the processed data. Of course, the power consumption also contains noise components and is influenced by various physical effects, but in (unprotected) hardware, there is always a small data-dependence present. The dependence on instructions is typically used in simple power-analysis attacks, while the data dependence is typically exploited in differential power-analysis attacks. The principles of both attacks will be explained in the following sections.

2.1 Simple Power-Analysis Attacks

Simple power-analysis attacks in general exploit the relationship between the instantaneous power consumption of a device and the executed instructions. For simple power-analysis attacks it is assumed that every instruction has its unique power-consumption trace. An attacker simply monitors the device's power consumption while it performs a cryptographic operation. Then, he carefully studies the obtained power-consumption trace to determine the sequence of instructions performed by the device. If this sequence is directly related to the secret key which was involved in the cryptographic operation, the attacker can deduce this secret key from the power-consumption trace. Such an attack typically targets implementations which use key dependent branching in the implementation.

A special kind of simple power-analysis attacks exploit a strong relationship between for example, the Hamming weight (i.e. the number of non-zero bits in the processed data) and the power-consumption trace (see [BS99] for a description of such an attack on an implementation of the DES). In such an attack, the leakage of the Hamming weight information is used to determine the secret key. For these types of attacks it is vital that the implementation is based on relatively small data-words such as for example, in an 8-bit implementation. This type of attack is usually applied to the implementation of the key schedule for a cipher. Yet, in implementations that try to achieve a protection against first-order differential power-analysis attacks, this method can be used to determine the used mask. Once the masking values are known, a standard differential power-analysis attack can be conducted.

2.2 Differential Power-Analysis Attacks

Differential power-analysis attacks exploit the correlation between the processed data and the instantaneous power consumption. Due to the fact that this correlation is usually very small, statistical methods must be used to exploit it. Figure 1 shows the concept of differential power- analysis attacks.

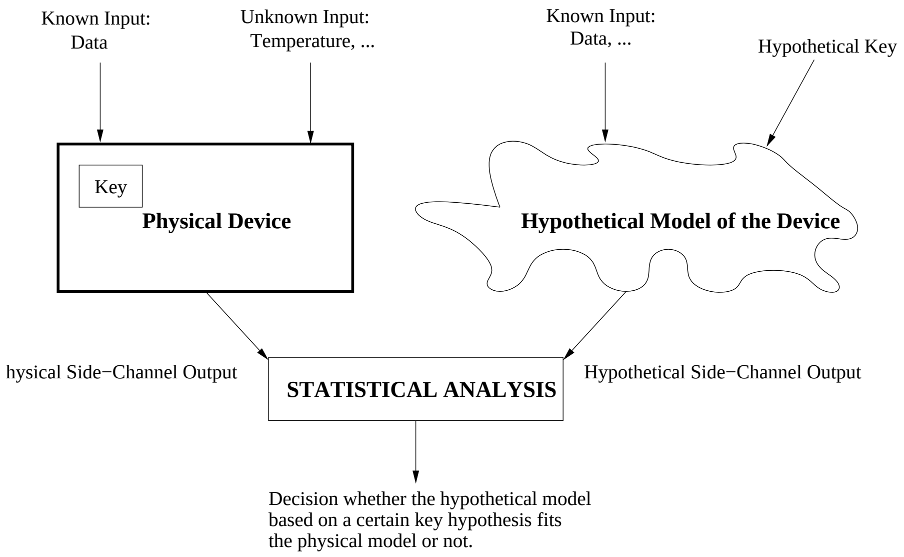

Known Input: Data
Unknown Input: Temperature, ...
Known Input: Data, ...
Hypothetical Key
Key
Physical Device
Hypothetical Model of the Device
hysical Side−Channel Output
STATISTICAL ANALYSIS
Hypothetical Side−Channel Output
Decision whether the hypothetical model based on a certain key hypothesis fits the physical model or not.

Figure 1: Concept of Differential Power-Analysis Attacks

In a differential power-analysis attack the output(s) of the real physical device and the output of a hypothetical model (based on a hypothetical key) of the device are compared. Only if the hypothetical key equals the real key, the output of the hypothetical model is correlated to the output of the real device. By comparing the two outputs, the secret key can be determined. If the hypothetical model only outputs a single value (i.e. it predicts the power consumption of the real device for only one moment in time) which is used in an attack, then the attack is called *first-order* differential power-analysis attack. If a model can output more values which are used in an attack then such an attack is called *higher-order* differential power-analysis attack. For example, if the model outputs two values, then one usually refers to the attack as a second-order differential power-analysis attack. If only the term differential power-analysis attack is used, then it refers to a first-order differential power-analysis attack.

Statistical methods are used to compare the outputs of the hypothetical model and the real device. The outputs are considered as the realizations of a statistical variable. Thus, comparisons can be done by comparing statistical values such as mean values or correlation coefficients.

An important aspect in differential power-analysis attacks is the quality of the hypothetical model that is used to predict the outputs of the real device. The term quality refers here to the number of bits it can predict correctly under the assumption of a specific power-consumption model. The more bits it can predict, the more accurate the model becomes and the more powerful it is. An attacker must therefore try to incorporate as much information about the physical device as possible into his hypothetical model. Typically, two types of attackers need to be distinguished. Firstly, attackers without information about the actual implementation. Such attackers basically only have outsider information, which typically consists of the knowledge which algorithm is executed and probably the data sheets for the hardware module which is used. Such attacks often result in a rather weak hypothetical model and are from now on referred to as outsider-attacks. Secondly, attackers who have insider information (e.g. a former chip designer). Such attackers have in a worst case scenario, very accurate information about the actual implementation, such as power consumption characteristics, the Spice-level model of the implementation, timing information, etc. Based on this information, an almost perfect hypothetical model can be built. Such attacks will be referred to as insider-attacks from now on. Of course, these two scenarios are the best-case and the worst-case attack scenarios and there are many levels in between them. An attacker without insider information who is experienced in the design of cryptographic hardware for example, can make reasonable assumptions about the actual implementation. Of course, knowledge about statistical properties of the attacked algorithm, can also improve the hypothetical model.

2.3 Comparing the Outputs by the Distance-of-Mean Test

The distance-of-mean test is a statistical method which compares two sets (or in statistical terms, distributions) by calculating the difference of their mean values. If the mean values are different (i.e. if they differ by a certain amount), then the two sets are considered to be different, otherwise they are considered to be equal. Paul Kocher et al. use this idea for their attack in [KJJ99]. In the attack described in this paper, the output of the hypothetical model is used to classify the output of the physical device into two sets. One set is supposed to consist only of measurements for which a low power consumption is predicted, the other set is supposed to consist of the measurements for which a high power consumption is predicted. If the hypothetical model (which is based on the hypothetical key) is correct, then one should be able to measure a difference between these two sets.

The hypothetical model used in this paper is rather poor. It assumes that the attacker has no information about the physical device (logic-style, etc.). Furthermore, the model only predicts one single bit for a complete DES-round. Of course, this model can be extended by making more assumptions about the implementation. Such extensions lead to hypothetical models that can predict more bits and therefore lead to a better classification into the two sets (see [MDS99]). We will refer to such attacks as the *mean-method* in the subsequent sections.

The main drawback of this statistical test is that it can only distinguish two different sets. Thus, the model's output can only be binary to be useful for this test. If the model can output more (accurate) predictions for the power consumption, other methods have to be used. The most common method (suggested in the scientific literature) is called ANOVA (analysis of variance). Our own experiments showed that the results when using the test described in the next section are better than the results obtained when using ANOVA. Therefore, we will not discuss the usage of ANOVA in more detail.

2.4 Comparing the Outputs by Calculating their Correlation Coefficient

Another possibility to compare the output of the hypothetical model and the output of the physical device is to calculate the correlation coefficient between the two outputs. Therefore, this method allows a direct comparison between the hypothetical model and the device itself. We will refer to this method as the *correlation-method* in the subsequent sections. The correlation method can be used to model insider as well as outsider attacks. This is the statistical method that we are have in mind for the differential power-analysis attacks considered in this paper.

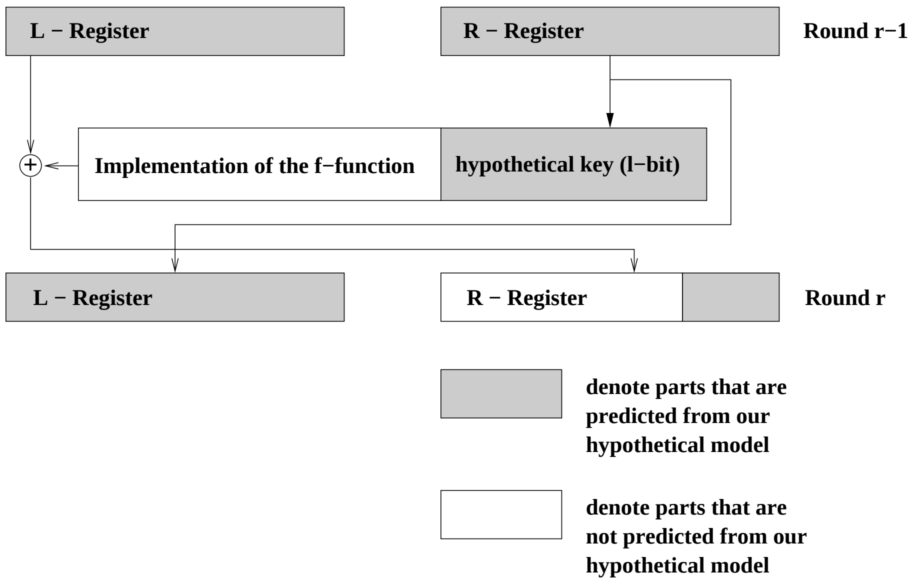
2.5 Differential Power-Analysis Attacks on Symmetric-Key Cryptosystems

In this section, we investigate the consequences of our approach described in the previous section towards the structures of block ciphers. Current designs are typically based on one of the following two structures. The structure which was used for example for the DES algorithm is called *Feistel-structure* and differs strongly from the structure that was for example used in the AES algorithm which is called *substitution-permutation network*.

The question is, what is the impact of the these structures on the quality of the hypothetical model for a differential power-analysis attack? This depends heavily on the device which we have in mind. On 8-bit processors, there is no difference at all. If we consider 32-bit processors it depends on the length of the data words that are used in the cipher. For modern ciphers this is usually 128 bits. So there are again no consequences at all. But, for certain implementations as for instance dedicated hardware implementations that work exactly with the size of the data words, it makes a large difference, as it will become clear in the next two sections.

2.5.1 The hypothetical model for Feistel ciphers

In this section, we describe the structure of our hypothetical model used for Feistel-ciphers. Figure 2 visualizes this model. This figure shows, which parts of one round of the cipher can be predicted by our model.



L − Register
R − Register
Round r−1
Implementation of the f−function
hypothetical key (l−bit)
L − Register
R − Register
Round r
denote parts that are predicted from our hypothetical model
denote parts that are not predicted from our hypothetical model

Figure 2: The hypothetical model that is used to attack ciphers with a Feistel structure where L_0 is not mixed with the key. The figure shows clearly that more than half of the bits of the first round can be predicted.

Of course, for (for example) 8-bit architectures, the contents of the L and the R register are computed sequentially, so the Feistel structure cannot be exploited in an attack. However, in a dedicated hardware-implementation, the implementation itself would very likely follow the Feistel-structure. This means, in a dedicated hardware implementation, the attacker can predict more than half of all bits if L_0 is not mixed with the key.

2.5.2 The hypothetical model for S-P networks

In this section we describe the structure of our hypothetical model used for substitution-permutation networks. Figure 3 visualizes this model. In this figure it is depicted, which parts of one round of the cipher can be predicted by our model.

Of course, for small architectures, such as 8-bit architectures, the content of the register is computed sequentially. A dedicated hardware implementation would again follow the structure of the cipher and thus, there would be a single register which would be updated after each round. Due to the structure of the

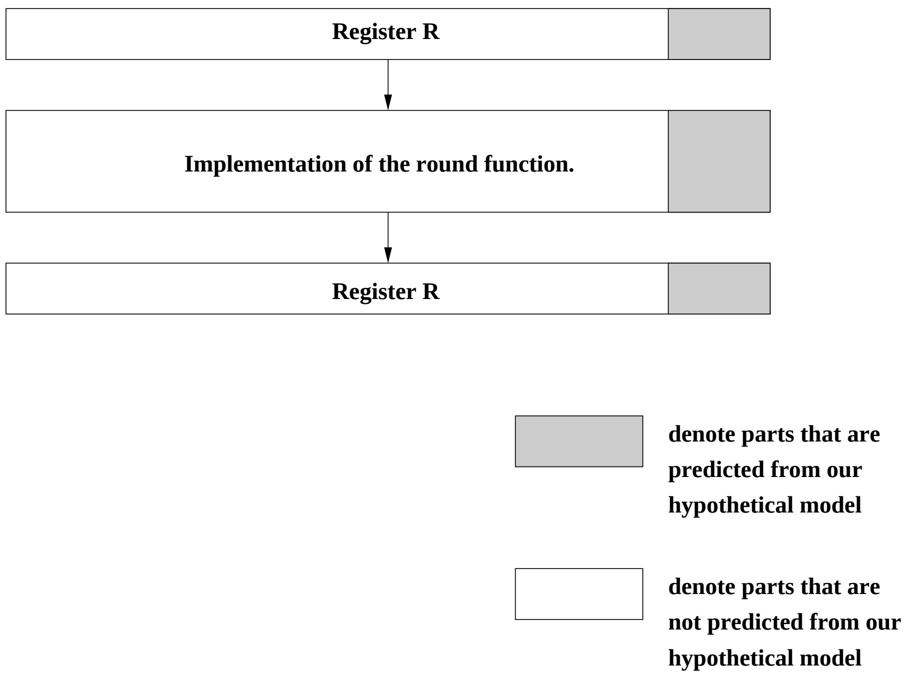

Register R
Implementation of the round function.
Register R
denote parts that are predicted from our hypothetical model
denote parts that are not predicted from our hypothetical model

Figure 3: The hypothetical model that is used to attack cipher with an S-P network structure.

cipher, only the bits that are directly influenced by the hypothetical key bits, can be predicted by our model.

3 Evaluation Criteria

Based on the attacks and on our models which we introduced in the previous section, we motivate the choice of the criteria for our comparison. There are two major issues that have to be taken into account. On the one hand it is important how easy it is to attack an unprotected implementation of an algorithm, and on the other hand, how easy it is to protect the implementation of an algorithm. We address those two issues in more detail in the two following subsequent sections.

3.1 Attacking an implementation

As stated in section 2 we can distinguish between simple power-analysis attacks and differential power-analysis attacks.

- Standard SPA: The type of simple power-analysis attacks which exploit the direct relationship between a sequence of operations and some secret data are really a matter of implementation and not related to a special property of an algorithm. Thus, we decided not to include such issues in our evaluation criteria.
- Hamming weight SPA: The type of simple power-analysis attacks that exploit Hamming weight information to derive the secret key depend on some properties of the algorithm. Usually, if the key schedule is simple, then only very few parts of some sub-keys have to be determined to get the secret key. This is also related to the countermeasure which is based on inserting randomness into the timing of the operations. Such a countermeasure can be defeated by attacking at a very early stage in the algorithm, where the timing variances are still very small. It is certainly an advantage for the attacker if only the first sub-key has to be attacked to get large parts of the secret key. Thus, we decided one evaluation criterion to be whether one or more sub-keys have to be attacked as one of our evaluation criteria.

Of course, simple power-analysis attacks are usually only an issue for software implementations of secret key cryptosystems.

- DPA on software implementations: Here the structure of the cipher is not important, since one round of the cipher is computed in several steps anyway. Considering an attack on the first round, the statistical properties of the S-box(es) are an issue. This is because in general, it cannot be assumed that the input data is random, so the power consumption cannot be assumed to be normally distributed, which in turn implies that the used statistical tests do not work optimal. Thus, we decided to take the statistical properties of the S-box(es) (or the round function) to be a criterion. Of course, this argument is not true for attacks which work on the last round, since then the input data has undergone many round transformations and key mixings, so the input data for the last round is sufficiently random. On the other hand, the quality of the S-box also influences the signal-to-noise ration for the keys independent of the number of the round. For these reasons we suggest to assign a low weight to this criterion.

- DPA on hardware implementations: Here, the structure of a cipher is an issue, so we decided to use it as an evaluation criterion. The statistical properties of an individual S-box is not so important since all S-box look-ups occur at the same time, so the statistical properties are not an issue for hardware implementations.

3.2 Protecting an implementation

We distinguish again between simple and differential power-analysis attacks.

- Hamming-weight SPA: To counteract the type of simple power-analysis attack that uses Hamming weight information, a designer has to assure that the Hamming weight information which is leaked is not correlated with the intermediate values that are processed. In dedicated hardware implementations this can be achieved by using a special logic-style or by masking intermediate values (this can be achieved by bus encryption as well as by masking the operations of the algorithm in general). In software implementations the intermediate values have to be masked.
- DPA on software implementations: Considering software implementations this can be done by masking the intermediate values or (and) introducing random operations to introduce randomness in the order of instructions.
- DPA on hardware implementations: Also the correlation between intermediate values and the power consumption has to be destroyed. Considering dedicated hardware implementations this can be done by either using special logic styles or by masking the intermediate values.

To protect the implementation of an algorithm either a special *logic style* or a *masking scheme* has to be used. The drawback of using special logic-styles is usually that they require more space and energy than standard logic-styles. An algorithm which requires fewer different types of gates and which requires only simple types of gates is therefore easier to implement in such a logic-style.

The drawback of the masking strategy is that in order to be also resistant against higher-order differential power-analysis attacks the mask has to be changed frequently during the execution of the algorithm. It is clear that an algorithm including operations that are difficult to mask, or where changing the mask is difficult, is less suited to be implemented securely. This is why we chose one of our criteria to be the usage of only simple (boolean) operations.

To conclude this section we define the criteria that are related to the protection of an implementation as follows. To simplify implementations in special logic-style an algorithm should be implementable with few different types of gates and with simple gates. Thus, one criterion is the usage of simple (boolean) operations. Furthermore, if the S-box has a simple description it can be implemented in combinational logic and does therefore not require the implementation of a secure RAM (or ROM). Thus, one criterion is the simplicity of the S-Box(es). To simplify the implementations of masking schemes only only simple operations (from the same algebraic group) should be used. Thus, this requirement is already included in the criterion of the usage of simple operations.

4 Choice of Algorithms

The algorithms which we consider in this article are chosen corresponding to the ciphers considered in the NESSIE project in phase two. These algorithms are IDEA, Khazad, Misty1, Safer++, Camellia, RC6, Shacal, Rijndael and Triple-DES. Rijndael and Triple-DES have been chosen to serve as benchmarks for the other algorithms.

5 Evaluation Results

In section 3, we already reasoned about the evaluation criteria which we are going to use to classify the algorithms. We briefly recall the criteria from the previous section.

Software Implementations

- **S_1. Difficulty of attacking the key schedule.** The more parts of the key schedule that have to be attacked the more amount of work an attacker has to spend (to identify the parts in the power trace which correspond to the parts of the key schedule that should be attacked).
- **S_2. The statistical properties of the S-box(es) or the round function.** Only if the bits after computing the S-box(es) or the first round are randomly distributed the statistical test can give a significant result for the correct key guess. Otherwise it is possible that several keys are suggested, or even a wrong key seems more likely than the correct one, or one key is especially easy (i.e. with significantly less samples) detected. Since it is reasonable to assume that the effects of the statistical properties of a cipher are covered by implementation specific effects, we will not include it directly in our rating for the ciphers.
- **S_3. The operations used by the algorithm.** If a countermeasure based on masking the intermediate values should be implemented, the operations that have to be masked are of interest. If boolean and arithmetic

operations occur in a cipher, it is rather complex to implement a masking scheme.

Hardware Implementations

- **H$_1$. The structure of the cipher.** We reasoned already in the sections before why Feistel-like ciphers are potentially easier to attack than Substitution-Permutation networks.

- **H$_2$. The description of the S-box.** S-boxes which can be implemented in combinational logic do not require the design of a special RAM (ROM)

- **H$_3$** $= S_3$. **The used operations.** If a countermeasure based on masking the intermediate values should be implemented, the operations that have to be masked are of interest. If boolean and arithmetic operations occur in a cipher, it is rather complex to implement a masking scheme, also in dedicated hardware.

Not all of these criteria are equally important, consequently we assign numbers (weights) to each criterion representing it's importance. If an algorithm fulfills a criterion completely then we give the maximum number of points which is 3 (very good, good, poor, not at all). The minimum number of points is 0. The general formula to rate an algorithm is then given in equation 1.

$$S = w_1 * S_1 + w_2 * S_2 + w_3 * S_3 + w_4 * H_1 + w_5 * H_2 \tag{1}$$

In the next sections we discuss for each algorithm if or if not it fulfills the criteria $S_1, \ldots, H_2$.

5.1 Rijndael

Rijndael ([DR02]) is a cipher which follows the substitution-permutation structure. It is defined for several key and data-lengths (the minimum is 128-bit). Its S-boxes are defined over GF(256). It has a rather simple key schedule, whereby the first sub-key is the secret key. An attack which would only use the pure Hamming weight information for the first sub-key would not be feasible in practice. However, in [Man02] Mangard describes how to determine the complete secret key by using Hamming weight information from a few sub-keys. Rijndael is well suited for 8-bit architectures and because of the definition of the S-box, it also leads to very small dedicated hardware implementations (see for example [MS02], [SMTM01], [WOL02], [IH01] and [APC$^+$01]). The operations used are very simple, so it is easy to implement a simple masking scheme (see [AG01]). This is also true for dedicated hardware implementations since the S-box can be implemented in combinational logic (with only exclusive or operations).

5.2 Triple-DES

Triple-DES basically consists of three DES executions with different keys. The DES itself is a cipher based on the Feistel-structure and operates on 64-bit blocks. The key schedule algorithm of the DES is rather simple and [BS99] presents an attack which can determine the secret key of the DES uniquely by attacking several sub-keys. The DES is fast and small in software implementations and it is very well suited for small hardware implementations as well. For the S-boxes however, no small implementation using combinational logic has ever been reported, thus when using a masking scheme in hardware the S-boxes have to be recalculated for every new mask.

5.3 IDEA

IDEA follows the Substitution-Permutation Network structure and is based on 16-bit data blocks. The key schedule is simple. It only consists of cyclic shifting of the whole 128-bit key, which makes it in fact susceptible to SPA attacks. Both, arithmetic and boolean operations are used. Since they work on 16-bit data blocks hardware implementations are still not too large. However boolean and arithmetic operations make the efficient implementation of a masking scheme very difficult.

5.4 Khazad

Kazhad is a substitution-permutation network and follows the design principles of Rijndael. It is an 8-bit oriented cipher and has a simple key schedule. But, the key schedule distributes the individual parts of the secret key quite well into the round keys, so it can be expected that Hamming weights of several round keys have to be determine to succeed with the Hamming weight attack. Its main difference from Rijndael (in the context of power-analysis attacks) is the S-box. It is not longer based on an inversion in GF(256), but was randomly chosen. Therefore, hardware implementations can be expected to be slightly larger in terms of gate count. Masking schemes can be implemented without too much penalty, only the S-box is problematic since it has to be recalculated for every new mask.

5.5 Misty1

Misty1 is a Feistel cipher working on 64-bit data words (thus L and R are 32-bit) and a 128-bit key. There are only boolean operations and table lookups used. The Key schedule is simple. The S-boxes where chosen in such a way that they are also implementable using combinational logic. This facilitates the implementation of masking schemes and the implementation in special logic styles. It is not easily possible to make use of the advantage of the Feistel structure in an attack since also the left half of the data undergoes a transformation involving the secret key during the first and the final round. However, since the cipher operates on relatively small data blocks, the effort that has to spend on the key guessing is possible in at least a very few year also on standard PCs.

5.6 Safer++

Safer++ (and the rest of the Safer family) are substitution-permutation networks. They basically operate on 8-bit data blocks and use exclusive or operations, table-lookups and addition modulo 128. The key schedule is not too difficult in [CJRR99] it is argued that at most half of the round keys need to be attacked in order to determine the secret key. Because the operations used are simple, masking schemes should be efficiently implementable.

5.7 Camellia

Camellia has a Feistel-structure and needs 18 rounds for encryption (or decryption). The key schedule has medium complexity (it applies the round function with fixed key values to generate sub-keys) and comprises also a key whitening phase. The pre-whitened keys are XOR-ed to the left and the right half of the plaintext. Thus, an attacker cannot make use of the Feistel-structure as explained in section 2. The S-boxes consist, as in the case of Rijndael, of an inversion operation in GF(256) plus other transformations. There are no arithmetic operations used. Thus, the cipher can be implemented in very few gates in dedicated hardware. Because of the nice properties of the S-box, masking schemes both in software and in hardware should be feasible without recalculating the S-boxes. Four different S-boxes are used.

5.8 RC6

RC6 is a Substitution-Permutation Network which basically works on 32-bit data blocks. The key schedule is difficult and it seems difficult to extract the complete key with the Hamming weight analysis. There are arithmetic and boolean operations used, but there is no S-box. Masking schemes are rather difficult to implement in both software and dedicated hardware because of the used operations. Also, a dedicated hardware implementation is larger than dedicated hardware implementations of ciphers that only use boolean operations, because of the arithmetic operations. Again, an implementation in a special logic style might be difficult because of the arithmetic operations.

5.9 Shacal

Shacal is the hash function SHA-1 in encryption mode. This encryption mode is defined for 160-bit data blocks and 512-bit key blocks. If shorter key/data blocks should be used, they are padded to reach the defined block lengths. In the encryption mode, the plain-text is used as initial vector and the key is used as message. SHA-1 uses both arithmetic and boolean operation, but no S-box. Masking schemes are rather difficult to implement in both software and dedicated hardware because of the used operations . Also, a dedicated hardware implementation is larger than dedicated hardware implementations of ciphers that only use boolean operations because of the arithmetic operations. Again, an implementation in a special logic style might be difficult because of the arithmetic operations.

6 Comparing the Algorithms

In this section we classify the algorithms according to formula 1 and we compare them with the benchmark algorithms Rijndael and Triple-DES, respectively. As already mentioned in section , we will not include criterion S2 in our comparison. We summarize the Evaluation results in table 1.

	S_1	S_3	H_1	H_2
Rijndael	1.5	3	3	3
Triple-DES	1	3	0	1
IDEA	1.5	2	3	3
Khazad(tw)	1.5	3	3	2
Misty1	1.5	3	1.5	2
Safer++	1.5	1	3	1
Camellia	1.5	3	2.5	2.5
RC6	2	0	3	3
Shacal	1.5	1	3	3

Table 1: Evaluation results of the NESSIE candidates

We give the mark *very good* (3 points) for S_1 if the Hamming weights of parts from all round keys have to be determined to deduce the secret key. The fewer parts have to be attacked the worse the marks get. As basis for our estimation of the difficulty serve [CJRR99], [BS99] and [Man02].

We give the 3 points for S_3 if only boolean operations are used. We give two points if only small (up to 16-bit) integer addition is used and one point if a larger than the small integer addition is used. Zero points are given if also multiplication is used in an algorithm.

We give 3 points for H_1 if a substitution-permutation network structure is used. We give 2.5 points if a Feistel cipher with preliminary key addition was used and the cipher operates on large blocks (larger or equal to 64 bit) otherwise we give 1.5 points. For a pure Feistel-cipher we give 0 points.

We give 3 points for H_2 if only one S-box is used which leads to a small implementation in combinational logic or if no S-box is used. We give less points the more S-boxes have to implemented and the larger they get.

Yet another matter is which weights should be assigned to the criteria. Since S_3 equals H_3 we suggest to set the weight for S_3 at least twice as high as the weights of the remaining criteria. Thus, we set w_3 to 2 and the remaining w_i to 1.

This leads to the following ranking:

Rijndael	13.5
Khazad(tw)	12.5
Camellia	12.5
IDEA	11.5
Misty1	11
Shacal	9.5
Triple-DES	8
RC6	8
Safer++	7.5

Table 2: A ranking of the NESSIE candidates according to the evaluation results and the assigned weights. The ranking will be different for different weights!

7 Conclusions

We derived some criteria for estimating the suitability of an algorithm to be implemented securely against power-analysis attack based on different attack scenarios. Then, we theoretically evaluated each of the NESSIE candidate algorithms against these criteria. We also included results previously published results from other researchers in our analysis. Finally, we gave a comparison between the candidates according to the defined criteria and a specific weighting of these criteria.

It appears, that the most important criterion is related to the operations which are used in an algorithm. It is important that an implementation only requires few different types of operations. This facilitates the implementation of software countermeasures such as masking, but also the implementation of hardware countermeasures, such as special logic styles.

References

[AG01] M.-L. Akkar and C. Giraud. An Implementation of DES and AES, Secure against Some Attacks. In *Workshop on Cryptographic Hardware and Embedded Systems - CHES 2000*, volume 2162 of *Lecture Notes in Computer Science*, pages 309–318. Springer, 2001.

[APC+01] A.Rudra, P.Dubey, C.Jutla, V.Kumar, J.Rao, and P.Rohatgi. Efficient Rijndael Encryption Implementation with Composite Field Arithmetic. In *Workshop on Cryptographic Hardware and Embedded Systems - CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 171–184. Springer, 2001.

[BS99] E. Biham and A. Shamir. Power Analysis of the Key Scheduling of the AES Candidates, 1999.

[CJRR99] S. Chari, C. Jutla, J. R. Rao, and P. Rohatgi. A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards. In *Second Advanced Encryption Standard (AES) Candidate Conference*, Rome, Italy, 1999.

[DR99] J. Daemen and V. Rijmen. Resistance Against Implementation Attacks. A Comparative Study of the AES Proposals, 1999.

[DR02] J. Daemen and V. Rijmen. *The Design of Rijndael.* Number ISBN 3-540-42580-2 in Information Security and Cryptography. Springer, 2002.

[IH01] I.Verbauwhede and H.Kuo. Architectural Optimization for a 1.82 Gbits/sec VLSI Implementation of the AES Rijndael Algorithm. In *Workshop on Cryptographic Hardware and Embedded Systems - CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 51–64. Springer, 2001.

[KJJ99] P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *Advances in Cryptology-CRYPTO 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[Man02] S. Mangard. A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion. Technical report, IAIK, TU-Graz, Inffeldgasse 16a, A-8010 Graz, 2002.

[MDS99] T.S. Messerges, E. A. Dabbish, and R.H. Sloan. Investigations of Power Analysis Attacks on Smartcards. In *Proceedings of USENIX Workshop on Smartcard Technology*, pages 151–162, 1999.

[MS02] S. Morioka and A. Satoh. An Optimized S-Box Circuit Architecture for Low Power AES Design. In *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems (CHES 2002)*, Lecture Notes in Computer Science. Springer, 2002. to appear.

[SMTM01] A. Satoh, S. Morioka, K. Takano, and S. Munetoh. A Compact Rijndael Hardware Architecture with S-Box Optimization. In *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 239–254. Springer, 2001.

[WOL02] J. Wolkerstorfer, E. Oswald, and M. Lamberger. An ASIC implementation of the AES SBoxes. In *Cryptographer's Track at the RSA Conference 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 67–78. Springer, 2002.