Hamming Weight Attacks on Cryptographic Hardware – Breaking Masking Defense[⋆]

Marcin Gomułkiewicz[1] and Mirosław Kutyłowski[12]

[1] Cryptology Centre, Poznań University

[2] Institute of Mathematics, Wrocław University of Technology,
ul. Wybrzeże Wyspiańskiego 27
50-370 Wrocław, Poland

Abstract. It is believed that masking is an effective countermeasure against power analysis attacks: before a certain operation involving a key is performed in a cryptographic chip, the input to this operation is combined with a random value. This has to prevent leaking information since the input to the operation is random.

We show that this belief might be wrong. We present a Hamming weight attack on an addition operation. It works with random inputs to the addition circuit, hence masking even helps in the case when we cannot control the plaintext. It can be applied to any round of the encryption. Even with moderate accuracy of measuring power consumption it determines explicitly subkey bits. The attack combines the classical power analysis (over Hamming weight) with the strategy of the saturation attack performed using a random sample.

We conclude that implementing addition in cryptographic devices must be done very carefully as it might leak secret keys used for encryption. In particular, the simple key schedule of certain algorithms (such as IDEA and Twofish) combined with the usage of addition might be a serious danger.

Keywords: cryptographic hardware, side channel cryptanalysis, Hamming weight, power analysis

1 Introduction

Symmetric encryption algorithms are often used to protect data stored in insecure locations such as hard disks, archive copies, and so on. The key used for this purpose is stored inside a cryptographic device and should not leave the device in an unencrypted form during its lifetime. The device must offer reasonable protection against retrieving the key, since security cannot be based on physical security of the device. For this reason, investigating cryptanalytic attacks against such devices has significant importance.

[⋆] This research was initiated when the second author visited University of Mannheim, this paper appeared in ESORICS'2002, Lecture Notes in Computer Science 2502, (Springer-Verlag, Berlin, 2002), 90-103, copyright: Springer Verlag

The threats for devices considered are not confined to such classical "mathematical" methods as differential or linear cryptanalysis, not less important are other methods taking into account not only plaintexts and ciphertexts, but also any sources of information due to physical nature of computation. Particularly dangerous might be information sources that are overlooked during designing of the algorithm or those that emerge at the time an algorithm is implemented.

Cryptanalysis of cryptographic algorithms based on such *side channel* information was initiated a few years ago. In the meantime it became a recognized and extremely dangerous line of attacks.

Timing attack. Execution time of encryption may leak information on the data involved. It has been first observed for asymmetric ciphers [11] – the operations such as modular exponentiation are optimized in order to reduce the computation time (which is a problem in this case). However, computation time becomes strongly dependent on data involved and in this way may leak information on the key. In the case of a symmetric ciphers timing attack may also be relevant, as shown in the example of non careful implementation of IDEA [10].

Power analysis attack. The second side channel source of information on computations inside cryptographic device is power consumption [12]. It depends very much on the operations performed and the data manipulated inside the device.

For technical reasons, we cannot measure consumption occurring at a chosen place in the circuit (although it is possible to do similar measurements for electromagnetic radiation [8]). We can measure only global consumption and only up to some extend. However, the power consumption can be sampled over the time providing information on different stages of computation separately.

Simple power analysis (SPA) takes the data on the global power consumption of the device during different moments of the computation. A more sophisticated method is **differential power analysis** (DPA) [12]. In that case, one considers the power consumption data and some other data computed, for instance, for all candidate subkeys that may occur at some point of the computation. The idea is that for the right subkey, power consumption and the data computed for the subkey should be somehow statistically related. On the other hand, for the wrong key relation between these data should be purely random. In this way, one hopes to retrieve information on the local keys despite the fact that only global information on power consumption is available.

Hamming attack. Some authors [10] indicate that for certain hardware technologies there is a strong correlation between the number of bits set to one and power consumption. Based on this phenomenon, they propose a *Hamming-weight attack* on DES-chips (and similar ciphers) with the aim to derive a secret key stored inside the device. The attack is potentially quite dangerous for block ciphers such that small portions of a key act independently on small portions of data – just like in the case of DES and its S-boxes.

It is required for the attack to determine the total Hamming weights of the output of the Feistel function of DES during the last round in a series of encryptions. This weight consists of weight of the output of an S-Box S we are attacking and the output of the remaining S-boxes. For finding the key one guesses the subkey bits which go into the XOR gate immediately before S. Then it is possible to compute the (hypothetical) outputs of S for the ciphertexts at hand. If the subkey bits were guessed correctly, then there is a statistical correlation between the weights of the computed outputs of S and the Hamming weights measured. For a wrong guess, there is no such a correlation. This is a way to determine which guess is correct and thereby to derive the key bits.

This attack cannot be applied as it is for non-Feistel ciphers. Also operations performed on larger data blocks increase the complexity of an attack significantly.

Saturation attack. Somewhat related to this paper is saturation attack, which was applied to such ciphers as Square [5], Crypton [17], Rijndael [6] and Twofish [13]. It can be used if a part, say U, of an encryption circuit performs a (key dependent) permutation on blocks of bits of a fixed length, say m. The general idea is that if we input each possible bit string of length m into U exactly once, then each bit string of length m occurs as an output exactly once. The order in which these outputs appear depends on an unknown key, but the set of values is known.

The idea of a saturation attack is that "global" behaviour of encryption algorithm of some part is key independent, even if "local" behaviour (that is, for a single input) is key dependent. This is used to "get through" U. However, a little bit different strategy can be applied: since "global" behaviour does not depend on particular input, it may depend on the key only. In this way it may leak some information on the key involved.

Masking. Some of the attacks mentioned rely on specific properties of certain operations. The methods such as timing attack, SPA, DPA and Hamming attack explore the property that certain parameters of an operation depend heavily on input/output data. Having found a specific situation we may conclude on data involved in the operation. A part of this data might be the key bits.

In order to prevent such attacks two techniques have been introduced. The first one proposes to "split" or "duplicate" all intermediate results [9, 2, 3]. However, this method has large memory or time overhead. The second much more efficient method, called *masking*, "blinds" input data to arithmetic and boolean operations [14, 4, 7]. The idea is to combine the input with a random value, then to perform an operation with the key, and finally extract the random factor. For instance, adding subkey k to an intermediate result a can be implemented by the following operations:

```
choose $r$ at random
$z = a + r$
$z = z + k$
$z = z - r$
```

Note that in the above procedure subkey k is added to a random number. This prevents the attacks mentioned. Similarly, one may mask bitwise XOR and multiplication operations. It is also possible to mask together addition and XOR [7], even if algebraically these operations are remote to each other.

1.1 New Hamming Weight Attack

We present a Hamming weight attack on addition operation where one operand is a subkey. In this way we derive efficiently most significant bits of the subkey. The attack has the following features:

– complexity of the attack depends mainly on the size of the words added, for the ciphers with 16-bit operations and simple key schedule (like IDEA) the attack reveals the whole main key; when key schedule is not reversible, then the attack reduces key space significantly,
– accuracy of measuring Hamming weight need not to be high, even poor preciseness suffices,
– it is irrelevant when the addition operation is performed, the attack does not use input and output data for the analysis,
– the data entering addition with the subkey may be masked, it does not prevent the analysis; moreover, it even helps to since the data to be added to the subkey should be fairly random: so in the situation when it is not possible to encrypt a random input, masking helps to get useful side-channel information,
– the attack computes directly subkey bits.

The side-channel information used for the attack is the Hamming weight of the sequence of carry bits. Please note: apart from Hamming weight of inputs and outputs, we also consider Hamming weight of internal data. If that weight can be measured in some way (even with a quite large error), then we are done. It does not disturb, if this weight is measured together with some other data like weight of input and output of addition, of the subkey, or any of these quantities. For the sake of clarity of exposition we describe the simplest case, in which addition is done with the school method. However, it follows from our considerations that the Hamming weight of a small number of operands existing in the addition circuit may be treated as an error of measurement and thereby our attack works in the same way.

Even if Hamming weight attacks seem at the moment to be more theoretical than practical, one has to be very careful when implementing addition in cryptographic devices. For instance, resetting registers holding the carry bits to zero before addition might help to use the standard power analysis devices: in this case power consumption due to putting the correct values of carry bits is closely related to the number of bit values changed, that is to the Hamming weight of the sequence of carry bits.

On top of that, although our attack is aimed to addition, it is probably not impossible to mount similar attack on e.g. (modular) multiplication. Multiplication is of course much more complicated operation, and finding and proving the closed formula similar to the one we prove for addiction seems very hard at the moment. Nevertheless, it is another possible threat for the hardware implementation of several block ciphers.

2 Properties of Addition

We consider addition of n-bits numbers modulo 2^n with the carry bit for position n thrown away. One of the operands will be the subkey to be derived, while the other one a random n-bit number.

We analyse the Hamming weight of numbers that occur during addition. We assume that addition is performed according to the school method. The effects specific to particular architectures of addition circuits are postponed to further research.

2.1 Expected Value of Hamming Weight

We adopt the following notation convention: random variables are denoted with capital letters, and their realisations with small letters. We make an exception for K - an unknown but fixed key.

First let us consider the total number of the ones that occur when we add a fixed key K to (chosen uniformly from $\{0, 1, \ldots, 2^n - 1\}$) number a. Note that we **do not** care about the value of a: we only want it to be chosen uniformly, and we need not know either input a or output $K + a$. If internal data are masked (as protection against power analysis), then uniform distribution is guaranteed by masking. If the device does not use masking we should use randomly chosen inputs (e.g. perform a chosen-plaintext attack).

Let $|x|$ denote the Hamming weight of number x. We consider the ones that occur while a is being added to (fixed) K. These ones fall into four categories: the ones occurring in K itself, the ones occurring in a, the ones occurring in $K + a$ and in the carry bits. Let $c = c(K, a)$ denote the number of carry-bits with the value 1. So the Hamming weight corresponding to adding a and K equals:

$$w = |K| + |a| + |K + a| + c(K, a)\,.$$

In terms of random variables we have:

$$W = |K| + |A| + |K + A| + C(K)\,,$$

where $C = C(K)$ is a random variable whose distribution depends on K alone: C realises as $c = c(K, a)$ which is a function of two arguments, one (K) fixed, and the other chosen at random (from a known distribution). Let $\mathbf{E}\,[\,X\,]$ denote the expected value of X. Since a is chosen uniformly from the set of all n-bit strings, we expect its Hamming weight to be close to $\frac{n}{2}$ and $\mathbf{E}\,[\,|A|\,] = \frac{n}{2}$. Similarly, Hamming weight of $K + a$ is supposed to be close to $\frac{n}{2}$ and $\mathbf{E}\,[\,|K + A|\,] = \frac{n}{2}$. Since expected value of a sum is a sum of expected values (even for dependent variables), we have:

$$\mathbf{E}\,[\,W\,] = |K| + \tfrac{n}{2} + \tfrac{n}{2} + \mathbf{E}\,[\,C(K)\,] = |K| + \mathbf{E}\,[\,C(K)\,] + n\,.$$

$\mathbf{E}\,[\,W\,]$ depends on the unknown key K only, so we shall denote it as a function of K: $\varphi(K)$. Basically, an attack could look as follows:

1. Collect a large pool of Hamming weight data corresponding to addition to the unknown key K. Compute average value $\widehat{\varphi}(K)$.
2. For any K' compute the theoretical values of $\varphi(K')$ and compare with $\widehat{\varphi}(K)$. If $|\varphi(K') - \widehat{\varphi}(K)|$ is large, reject K'.

Of course, such an attack would be not very practical. The attack we present is much simpler. We do not have to execute the second step: we derive certain bits of K directly from $\widehat{\varphi}(K)$. Obviously, the key point is to determine the relation between $C(K)$ and the (sub)key used.

2.2 Formula for Expected Hamming Weight of Carry Bits

In this subsection we prove the following key lemma.

Lemma 1. *Let $K = (k_{n-1}, \ldots, k_0)$ be a n-bit string. Let $C = C(K)$ denote the number of carry bits set to 1 that occur during computation of $K + a \bmod 2^n$ with the school method, where a is chosen uniformly at random from the set of all n-bit strings. Then*

$$\mathbf{E}\,[\,C(K)\,] = \sum_{i=0}^{n-1} k_i - 2^{1-n} \cdot K\,. \tag{1}$$

Proof. Let $A = (a_{n-1}, \ldots, a_0)$. Let c_i denote the ith carry bit generated while adding K to A. Obviously, $c_0 = 0$ (there is no carry at the beginning), and c_1 depends only on k_0 and a_0 while for $i > 0$ the bit c_i depends on a_{i-1}, k_{i-1} and c_{i-1}. This dependence complicates computation of $\mathbf{E}\,[\,C(K)\,]$.

In order to count the number of carry bits set to 1 we consider $a_{n-1}, \ldots, a_0$ as independent random variables with values $0, 1$ and uniform distribution. Let $\mathbf{P}\,[\,X\,]$ denote probability of an event X. For $i = 1$ we have:

$$\begin{aligned} \mathbf{P}\,[\,c_1 = 0\,] &= \mathbf{P}\,[\,k_0 + a_0 \le 1\,] &&= 1 - \tfrac{1}{2} \cdot k_0\,, \\ \mathbf{P}\,[\,c_1 = 1\,] &= \mathbf{P}\,[\,k_0 + a_0 > 1\,] &&= \tfrac{1}{2} \cdot k_0\,. \end{aligned}$$

For $i > 1$ we may easily derive that

$$\begin{aligned} \mathbf{P}\,[\,c_i = 0 \mid c_{i-1} = 0\,] &= \mathbf{P}\,[\,c_{i-1} + k_{i-1} + a_{i-1} \le 1 \mid c_{i-1} = 0\,] \\ &= \mathbf{P}\,[\,k_{i-1} + a_{i-1} \le 1\,] = 1 - \tfrac{1}{2} \cdot k_{i-1}\,, \\ \mathbf{P}\,[\,c_i = 0 \mid c_{i-1} = 1\,] &= \mathbf{P}\,[\,c_{i-1} + k_{i-1} + a_{i-1} \le 1 \mid c_{i-1} = 1\,] \\ &= \mathbf{P}\,[\,k_{i-1} + a_{i-1} \le 0\,] = \tfrac{1}{2} - \tfrac{1}{2} \cdot k_{i-1}\,, \\ \mathbf{P}\,[\,c_i = 1 \mid c_{i-1} = 0\,] &= \mathbf{P}\,[\,c_{i-1} + k_{i-1} + a_{i-1} > 1 \mid c_{i-1} = 0\,] \\ &= \mathbf{P}\,[\,k_{i-1} + a_{i-1} > 1\,] = \tfrac{1}{2} \cdot k_{i-1}\,, \\ \mathbf{P}\,[\,c_i = 1 \mid c_{i-1} = 1\,] &= \mathbf{P}\,[\,c_{i-1} + k_{i-1} + a_{i-1} > 1 \mid c_{i-1} = 1\,] \\ &= \mathbf{P}\,[\,k_{i-1} + a_{i-1} > 0\,] = \tfrac{1}{2} + \tfrac{1}{2} \cdot k_{i-1}\,. \end{aligned}$$

One may treat the random variables $c_1, c_2, \ldots$ as a non-homogeneous Markov chain where transition probabilities depend on the values of $k_0, k_1, \ldots, k_{n-1}$. For each i we consider vector $P_i = [1 - p_i, p_i]$ where p_i stands for $\mathbf{P}\left[\, c_i \,\right] = 1$. Then

$$P_1 = [1 - \tfrac{1}{2} \cdot k_0 \ , \ \tfrac{1}{2} \cdot k_0] \ ,$$

and for $i > 1$

$$P_i = P_{i-1} \cdot \begin{bmatrix} 1 - \frac{1}{2} \cdot k_{i-1} & \frac{1}{2} \cdot k_{i-1} \\ \frac{1}{2} - \frac{1}{2} \cdot k_{i-1} & \frac{1}{2} + \frac{1}{2} \cdot k_{i-1} \end{bmatrix} \ .$$

We get

$$\begin{aligned} 1 - p_i &= (1 - p_{i-1}) \cdot (1 - \tfrac{1}{2} \cdot k_{i-1}) + p_{i-1} \cdot (\tfrac{1}{2} - \tfrac{1}{2} \cdot k_{i-1}) \\ &= 1 - \tfrac{1}{2} \cdot (p_{i-1} + k_{i-1}) \end{aligned}$$

and so

$$p_i = \tfrac{1}{2} \cdot (p_{i-1} + k_{i-1}) \ . \tag{2}$$

By equality (2) we get easily

$$p_i = \tfrac{1}{2}(k_{i-1} + \tfrac{1}{2} \cdot (k_{i-2} + \tfrac{1}{2} \cdot (\ldots (k_1 + \tfrac{1}{2} k_0) \ldots))) \ . \tag{3}$$

Since c_i is a random variable with values $0, 1$, we have $\mathbf{E}\left[\, c_i \,\right] = p_i$. Then, by linearity of expectation,

$$\begin{aligned} \mathbf{E}\left[\, \textstyle\sum_{i=1}^{n-1} c_i \,\right] &= \sum_{i=1}^{n-1} p_i \\ &= \tfrac{1}{2} \cdot k_0 + \tfrac{1}{2} \cdot (k_1 + \tfrac{1}{2} \cdot k_0) + \tfrac{1}{2} \cdot (k_2 + \tfrac{1}{2} \cdot (k_1 + \tfrac{1}{2} \cdot k_0)) + \ldots \\ &\qquad + \tfrac{1}{2} \cdot (k_{n-2} + \tfrac{1}{2} \cdot (\ldots)) \\ &= k_0 \left(\tfrac{1}{2} + \tfrac{1}{4} + \ldots + \tfrac{1}{2^{n-1}} \right) + k_1 \left(\tfrac{1}{2} + \tfrac{1}{4} + \ldots + \tfrac{1}{2^{n-2}} \right) + \ldots \\ &\qquad + k_{n-3} \left(\tfrac{1}{2} + \tfrac{1}{4} \right) + k_{n-2} \left(\tfrac{1}{2} \right) \\ &= k_0 \left(1 - \tfrac{1}{2^{n-1}} \right) + \ldots + k_{n-2} \left(1 - \tfrac{1}{2} \right) + k_{n-1} \left(1 - 1 \right) \ . \end{aligned}$$

So we see that

$$\mathbf{E}\left[\sum_{i=1}^{n-1} c_j \right] = \sum_{i=0}^{n-1} k_i - 2^{1-n} \cdot K \ .$$

This concludes the proof of Lemma 1. □

We note that the expected value $\mathbf{E}\left[\, C(K) \,\right]$ depends on K in a very nice way – the bits of K influence separate bites of $\mathbf{E}\left[\, C(K) \,\right]$ (except the most significant part of $\mathbf{E}\left[\, C(K) \,\right]$).

2.3 Deviations from Expected Value on Random Inputs

The expected value $\varphi(K)$ may be experimentally determined as a mean value of a series of random experiments. Now it is crucial to determine how close this approximation is.

Lemma 2. *With probability at least* $1 - n \times 6.33 \times 10^{-5}$*, the observed value* C *does not deviate from its expectation in* N *experiments more than* $2n \cdot \sqrt{N}$.

Proof. Now we consider N stochastically independent experiments in which a random number is added modulo 2^n to the same (sub)key K. In each experiment some carry bits are set, e.g. like this ($n = 16$):

	c_{15}	c_{14}	c_{13}	c_{12}	c_{11}	c_{10}	c_9	c_8	c_7	c_6	c_5	c_4	c_3	c_2	c_1	c_0
Exp. 1	1	0	0	0	1	0	0	0	0	1	1	1	0	1	0	0
Exp. 2	0	0	0	1	1	1	1	1	0	1	1	0	0	1	1	0
Exp. 3	1	1	0	0	0	1	0	1	0	1	0	0	1	0	0	0
…																
Exp. N	0	0	0	1	1	1	0	0	0	0	1	1	0	0	1	0

($c_0 = 0$)

We shall think of columns separately, e.g. column i represents outcomes of independent experiments which yield result 1 with (some) probability p_i and 0 with probability $1 - p_i$. Variation of one such experiment is $p_i \cdot (1 - p_i)$. We sum up the elements in the column, let a random variable X_i denote this sum. Clearly, $\mathbf{E}[X_i] = N \cdot p_i$. We shall estimate the probability of large deviations from this expected value.

If $p_i \in \{0, 1\}$, then we always get the expected value. If $p_i \notin \{0, 1\}$, by Central Limit Theorem, $X_i \sim N(\mu, \sigma^2)$, where $\mu = Np_i$ and $\sigma = \sqrt{Np_i(1 - p_i)}$, and $N(\mu, \sigma^2)$ denotes normal distribution with expectation μ and variation σ^2. Since the maximum of function $p \mapsto p(1 - p)$ is in $p = \frac{1}{2}$ and equals $\frac{1}{4}$, we see that $\sigma \leq \sqrt{\frac{N}{4}}$. For a normal distribution the probability that observation deviates from expectation by more than 4σ is about $6.33 \cdot 10^{-5}$. Therefore with the probability of at least $1 - 6.33 \cdot 10^{-5}$:

$$\mathbf{E}[X_i] - 2\sqrt{N} \ \leq \ X_i \ \leq \ \mathbf{E}[X_i] + 2\sqrt{N}$$

for every $i \in \{1, 2, \ldots, n - 1\}$. By summing up n these inequalities we get

$$\mathbf{E}[C] - 2n\sqrt{N} \leq C \leq \mathbf{E}[C] + 2n\sqrt{N}\,.$$

(since $c_0 = 0$, we could sum $(n - 1)$ inequalities only; we sum n to simplify further calculations a bit)

This event occurs with probability at least $1 - n \times 6.33 \times 10^{-5}$ (more precisely: at least $1 - (n - 1) \times 6.33 \times 10^{-5}$). This concludes the proof of Lemma 2. □

For $n = 16$ mentioned probability equals about 0.99898, for $n = 32$ about 0.99797.

3 Deriving Subkeys

3.1 Ideal Case

As we have shown above, the expected value of Hamming weight in N experiments equals

$$\begin{aligned} N \cdot n + N \cdot \textstyle\sum_{i=1}^{n} k_i + N \cdot (\sum_{i=1}^{n} k_i - 2^{1-n} K) \\ = N \cdot n + 2N \cdot (\textstyle\sum_{i=1}^{n} k_i - 2^{-n} \cdot K)\,. \end{aligned} \tag{4}$$

The values N and n are known, $K = (k_{n-1} \dots k_1 k_0)$ is sought. For a moment let us assume that $N = 2^n$, the Hamming weights are measured exactly, and each input number occurs exactly once (just like in saturation attack). Let x be the total Hamming weight measured. Then:

$$x = 2^n \cdot n + 2 \cdot (2^n \sum_{i=1}^{n} k_i - K).$$

Since $K < 2^n$,

$$\sum_{i=1}^{n} k_i = \left\lceil \frac{x - 2^n \cdot n}{2^{n+1}} \right\rceil \tag{5}$$

and

$$K = \left(-\frac{x - 2^n \cdot n}{2} \right) \bmod 2^n\,. \tag{6}$$

As we can see, it would be possible to compute directly key K from x using information gathered in our experiment.

3.2 Taking Deviations into Account

Of course, the likelihood of an ideal situation considered above is rather small. We will get $N \cdot |K|$ ones (belonging to K), but we will not get exactly $N \cdot n/2$ ones before and after addition, nor exactly $N \cdot (\sum_{i=1}^{n} k_i - 2^{1-n} K)$ carry bits. On top of all, we have to consider measurement errors. Nevertheless, we shall see that it is still possible to gather significant information about the key.

Let us assume we want to find some information about n-bit key and can make $N = 2^m$ additions. We expect about $2^m \cdot n/2$ ones in the input (and the same amount of ones in the output) of the adding circuit. We shall think of input bits as of $2^m \cdot n$ independent random variables (say, X_i) with uniform distribution on $\{0, 1\}$. Then $\mathbf{E}[\,X_i\,] = \frac{1}{2}$ and $\mathbf{Var}[X_i] = \frac{1}{4}$. By Central Limit Theorem we may say that:

$$\textstyle\sum_{i=1}^{n \cdot 2^m} X_i \sim N(\mu, \sigma^2)$$

where $\mu = 2^m \cdot \frac{n}{2}$ and $\sigma^2 = \left(\sqrt{\frac{1}{4} \cdot 2^m \cdot n} \right)^2 = 2^{m-2} \cdot n$.

It is known that with probability close to 1 (that is at least $1-6.33\cdot 10^{-5}$) the distance between the actual value and expectation of a normal random variable is not greater than 4σ. In our case, $4\sigma = 4\cdot\left(\sqrt{2^{m-2}\cdot n}\right) = 2^{m/2+1}\cdot\sqrt{n}$. The same calculations are valid for the output ones.

In the previous subsection we have shown that the total number C of carry bits falls into the interval

$$\left[\mathbf{E}\left[\,C\,\right] - 2n\cdot 2^{m/2}\ ,\ \mathbf{E}\left[\,C\,\right] + 2n\cdot 2^{m/2}\right]$$

with the probability of at least $1 - n\cdot 6.33\cdot 10^{-5}$.

Finally, there are errors due to measurement inaccuracies. We assume that error ε_i in experiment i is a random variable with the values in the interval $[-u, u]$ uniformly distributed. Then $\mathbf{E}\left[\,\varepsilon_i\,\right] = 0$ and $\mathbf{Var}\left[\varepsilon_i\right] = \frac{u^2}{3}$. The values of u depend on the equipment available, but for $n = 16$ it is sound to assume for instance that $u \leq 2$ (error of magnitude 12.5%). By Central Limit Theorem

$$-4\cdot 2^{m/2}\cdot\frac{u}{\sqrt{3}}\ \leq\ \sum_{i=1}^{2^m}\varepsilon_i\ \leq\ 4\cdot 2^{m/2}\cdot\frac{u}{\sqrt{3}}$$

with the probability of at least $1 - 6.33\cdot 10^{-5}$.

Together, with the probability of at least $1 - (n+2)\cdot 6.33\cdot 10^{-5}$ all kinds of deviations from the expected value sum up to a value not exceeding

$$2\cdot 2^{m/2+1}\cdot\sqrt{n} + 2n\cdot 2^{m/2} + 4\cdot 2^{m/2}\cdot\frac{u}{\sqrt{3}} = 2^{m/2+1}\left(2\sqrt{n} + n + 2\cdot\frac{u}{\sqrt{3}}\right)$$

We have to compare the above value with the expected value of the Hamming weight (4):

$$2^m\cdot n + 2^{m+1}\cdot(\sum_{i=1}^{n} k_i - 2^{-n}\cdot K)\,.$$

In the last expression we are interested in bit positions $m-1, m-2, \ldots, m-n$ representing the key K . On the other hand, the deviations from the expected value and measurement errors influence the positions up to $m/2+1+\log(2\sqrt{n}+n+2\cdot\frac{u}{\sqrt{3}})$. For a very large m we can obviously read all key bits. However, we have to balance the number of experiments N and gaining key bits. In the next subsection we discuss some practically relevant settings of parameters.

3.3 Examples

Below we put concrete parameters for N, n and u and check how many key bits we may gain from the experiment.

We describe separately three kinds of deviations from the expected value: *error 1–* deviations from the expected Hamming weight of carry bits, *error 2–*

deviations from the expected value of the total Hamming weight of the inputs and outputs, *error 3*– deviations due to measurement errors. *Total error* denotes the estimated value of the largest deviations that may occur. *Signal level* 2^i means that i is the first bit position corresponding to the key K in the expression for the expected value of the total Hamming weight (we mean that the least significant bit of a binary number has position 0).

The following table describes the situation for $n = 16$ (the subkey length is like for IDEA) and $u \approx 2^{2.5}$:

N	error 1 level	error 2 level	error 3 level	total error	signal level	key bits found
2^{16}	$\sim 2^{13}$	$\sim 2^{12}$	$< 2^{9.2} \cdot u$	$< 2^{14}$	2^{1}	3
2^{18}	$\sim 2^{14}$	$\sim 2^{13}$	$< 2^{10.2} \cdot u$	$< 2^{15}$	2^{3}	4
2^{20}	$\sim 2^{15}$	$\sim 2^{14}$	$< 2^{11.2} \cdot u$	$< 2^{16}$	2^{5}	5
2^{22}	$\sim 2^{16}$	$\sim 2^{15}$	$< 2^{12.2} \cdot u$	$< 2^{17}$	2^{7}	6
2^{24}	$\sim 2^{17}$	$\sim 2^{16}$	$< 2^{13.2} \cdot u$	$< 2^{18}$	2^{9}	7
2^{26}	$\sim 2^{18}$	$\sim 2^{17}$	$< 2^{14.2} \cdot u$	$< 2^{19}$	2^{11}	8

For $n = 32$ and $u \approx 2^{3.5}$ we get:

N	error 1 level	error 2 level	error 3 level	total error	signal level	key bits found
2^{16}	$\sim 2^{14}$	$\sim 2^{12.5}$	$< 2^{9.2} \cdot u$	$< 2^{15}$	2^{-15}	2
2^{20}	$\sim 2^{16}$	$\sim 2^{14.5}$	$< 2^{11.2} \cdot u$	$< 2^{17}$	2^{-11}	4
2^{24}	$\sim 2^{18}$	$\sim 2^{16.5}$	$< 2^{13.2} \cdot u$	$< 2^{19}$	2^{-7}	6
2^{28}	$\sim 2^{20}$	$\sim 2^{18.5}$	$< 2^{15.2} \cdot u$	$< 2^{21}$	2^{-3}	8
2^{32}	$\sim 2^{22}$	$\sim 2^{20.5}$	$< 2^{17.2} \cdot u$	$< 2^{23}$	2^{1}	10
2^{36}	$\sim 2^{24}$	$\sim 2^{22.5}$	$< 2^{19.2} \cdot u$	$< 2^{25}$	2^{5}	12
2^{40}	$\sim 2^{26}$	$\sim 2^{24.5}$	$< 2^{21.2} \cdot u$	$< 2^{27}$	2^{9}	14
2^{44}	$\sim 2^{28}$	$\sim 2^{26.5}$	$< 2^{23.2} \cdot u$	$< 2^{29}$	2^{13}	16

It is quite astonishing that measurement accuracy u might be quite poor, but we are still able to find a significant number of key bits.

4 Vulnerability of Popular Algorithms

4.1 IDEA

IDEA encryption algorithm uses 18 subkeys of length 16 for addition. The key schedule is so simple that the subkey bits are directly the bits of the main key.

If our aim is to break completely the main key, we should find at least 4 (preferably 5) bits per subkey – that would yield 72 or 90 bits of the main key. Given that we could find the remaining bits of the main key by an exhaustive search (at most 2^{56} or 2^{38} trials). By inspecting the tables in the previous section we see that we need about 2^{20} experiments per subkey, and the error in measuring Hamming weight should not exceed some 12%. Note that subkey bits are **exactly** the bits of the main key.

4.2 Twofish

Twofish uses two 32-bit subkeys in each of its 16 rounds for addition. The key schedule, although not as simple as in IDEA, is still reversible. It can be easily shown that to find the main key it suffices:

128-bit key: 4 subkeys used for addition during the first two rounds and about $8 \cdot 2^8$ very simple operations,

192-bit key: 6 subkeys used for addition during the first three rounds and about $8 \cdot 2^{16}$ very simple operations,

256-bit key: 8 subkeys used for addition during the first four rounds and about $8 \cdot 2^{24}$ very simple operations.

As we know, finding more than 16 bit per 32-bit long subkey is rather infeasible; however, for breaking 128-bit main key we do not require more. We find 16 most important bits of 4 keys, and guess the rest. Then we reverse the key schedule, and test the keys. Of course, we should begin our guessing from value suggested by the procedure: if we have measured x_i, $1 \leq i \leq 4$ ones during 2^m additions, then we should start with $K_i = \left(-\frac{x_i - 2^m \cdot 32}{2}\right) \bmod 2^{32}$ (see equation 5), and then increase and decrease guessed keys by 1; also note, that expression $\left\lceil \frac{x_i - 2^m \cdot 32}{2^{m+1}} \right\rceil$ (see equation 6) gives us an approximation of K_i's Hamming weight; keys with Hamming weights "much" different than those may be checked later. In worst case we will have to check 2^{64} possibilities. Since an exhaustive search of 2^{64} queries is possible, we are able to recover the whole main key. The total number of operations required is in the range of 2^{64}. This is not few, but compared with the claimed 128-bit security it is very attractive.

Complete breaking 192 and 256-bit keys are not practically possible: on average it would require 2^{95} or 2^{127} attempts (note, that reversing of the key schedule is also more tedious). However, it is still a great improvement over the exhaustive search in the set of 2^{192} or 2^{256} keys.

4.3 MARS

MARS encryption algorithm [1] uses several 32-bit long subkeys for addition: in fact key addition is the first, and key subtraction (which may be also implemented as addition) is the last operation performed on data. Apart from that, another subkey addition takes place inside the E function used in each of 16 rounds of the keyed transformation. Summarily we have $4 + 16$ to $4 + 16 + 4$ subkeys used for addition. As shown above, at a cost of about 2^{44} encryptions we may find half of each subkey, i.e. 320 to 384 bits of total 1280 bits of an expanded key. Due to its complex structure, the key-schedule does not seem to be reversible, so an attack would require to execute key expansion scheme and falsify wrong key guesses. Although it does not seem very practical, it is still an improvement over the exhaustive search.

4.4 RC5 and RC6

RC5 and RC6 use addition as one of their building blocks. RC5 uses $2 + 2r$ subkeys for addition: two at the beginning and two more per each round. RC6 works similarly, on top of that it uses another two subkeys at the end of computation.

Case of RC5 and RC6 is in a way similar to MARS: since the key expansion scheme seems to be irreversible, our attack is rather a potential than an actual threat. It is worth noting, though, that there are no other subkeys than those used for addition. This property would theoretically allow us to break the cipher by finding all the subkeys. Let us assume that we have equipment of great (preferably: indefinite) accuracy, so we can measure (almost) exact Hamming weights (as in section **Ideal case**). Therefore we start with a saturated set of plaintexts and find the keys one by one from the first round to the last (we peel off consecutive rounds starting from the beginning; since we need to recover $2 + 2r$ keys, we want to minimize all the errors as much as possible). In that case, even though we still do not know the main key, we are able to duplicate the encrypting device or decrypt encrypted messages. Of course, such a theoretical attack would not be possible against Twofish which uses key-dependent permutation, or MARS which uses another subkeys for operations different from addition.

5 Conclusions

Our attack reveals that addition should be used with special care in hardware implementations of symmetric block ciphers. Potentially, a side channel attack may retrieve completely the secret key stored in a protected device, or at least provide significant amount of information on this key.

It also indicates that masking with random values the inputs of the arithmetic operations is not a universal technique that may be used as a defense against side channel attack on protected cryptographic hardware storing secret symmetric keys. Unexpectedly, masking may even help to perform the attack.

The attack presented is one more argument that the subkey schedules of symmetric ciphers should evolve into irreversible schemes.

References

1. Burwick C., Coppersmith D., D'Avignon E., Gennaro R., Halevi S., Jutla C., Matyas S., O'Connor L., Peyravian M., Safford D., Zunic N., *MARS — A Candidate Cipher for AES*, `http://www.research.ibm.com/security/mars.html`
2. Chari S., Jutla Ch., Rao J. R., Rohatgi P., *A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards*, Second Advanced Encryption Standard (AES) Candidate Conference,
3. Chari S., Jutla Ch., Rao J. R., Rohatgi P., *Towards sound approaches to counteract power-analysis attacks*, CRYPTO'99, Lecture Notes in Computer Science 1666. Springer-Verlag, 398-412.

4. Coron J. S., *On Boolean and arithmetic masking against differential power analysis*, CHES'2000, Lecture Notes in Computer Science 1965. Springer-Verlag, 231–237.
5. Daemen J., Knudsen L., Rijmen V., *The block cipher Square*, Fast Software Encryption'97, Lecture Notes in Computer Science 1267. Springer-Verlag, 149–165.
6. Daemen J., Rijmen V., *The block cipher Rijndael*, `http://www.esat.kuleuven.ac.be/~rijmen/rijndael`
7. Goubin L., *A sound method for switching between Boolean and arithmetic masking*, CHES'2001, Lecture Notes in Computer Science 2162. Springer-Verlag, 3–15.
8. Gandolfi K., Mourtel Ch., Olivier F., *Electromagnetic Analysis: Concrete Results*, CHES'2001, Lecture Notes in Computer Science 2162. Springer-Verlag, 251–261.
9. Goubin L., Patarin J., *DES and Differential Power Analysis (The "Duplication" Method)*, CHES'99, Lecture Notes in Computer Science 1717. Springer-Verlag, 158-172
10. Kesley J., Schneier B., Wagner D., Hall Ch., *Side channel cryptanalysis of product ciphers*, Journal on Computer Security 8 (2000), 141–158.
11. Kocher P., *Timing Attacks on Implementations of Diffie-Hellman*, RSA, DSS, and Other Systems. CRYPTO'96, Lecture Notes in Computer Science 1109. Springer-Verlag, 104-113.
12. Kocher P., Jaffe J., Jun B., *Differential power analysis*, CRYPTO'99, Lecture Notes in Computer Science 1666. Springer-Verlag, 388–397, also: *Introduction to differential power analysis and related attacks*, `http://www.cryptography.com/dpa/technical`
13. Lucks S., *The saturation attack - a bait for Twofish*, `http://eprint.iacr.org/2000/046/`
14. Messerges Th., *Securing AES finalists against power analysis attack*, FSE'2000, Lecture Notes in Computer Science 1978. Springer-Verlag, 150–164.
15. Rivest R., Robshaw M., Sidney R., *The RC6 Block Cipher*, `http://theory.lcs.mit.edu/~rivest/rc6.ps`
16. Schneier B., Kesley J., Whiting D., Wagner D., Ch. Hall, N.Ferguson, *The Twofish Encryption Algorithm: a 128-Bit Block Cipher*, Wiley, 1999, ISBN 0-471-35381-7.
17. AES Development Effort, NIST, `http://www.nist.gov/aes`