Differential Power Analysis of Stream Ciphers

W. Fischer, B.M. Gammel, O. Kniffler, and J. Velten

Infineon Technologies AG, Germany

Wieland.Fischer@infineon.com
Berndt.Gammel@infineon.com
Oliver.Kniffler@infineon.com
Joachim.Velten@infineon.com

Abstract. Side-channel attacks on block ciphers and public key algorithms have been discussed extensively. However, there is only sparse literature about side-cannel attacks on stream ciphers. The few existing references mainly treat timing [8] and template attacks [10], or provide a theoretical analysis [6], [7] of weaknesses of stream cipher constructions. In this paper we present attacks on two focus candidates, Trivium and Grain, of the eSTREAM stream cipher project. The attacks exploit the resynchronization phase of ciphers. A novel concept for choosing initial value vectors is introduced, which totally eliminates the algorithmic noise of the device, leaving only the pure side-channel signal. This attack allows to recover the secret key with a small number of samples and without building templates. To prove the concept we apply the attack to hardware implementations of the ciphers. For both stream ciphers we are able to reveal the complete key.

Keywords: side-channel attack, power analysis, DPA, stream cipher, Trivium, Grain.

1 Introduction

Differential power analysis (DPA) is a well-known and thoroughly studied threat for implementations of block ciphers, like DES and AES, and public key algorithms, like RSA. However, in the field of stream ciphers this topic is rather unknown. More generally, this is even true for any kind of side-channel analysis (SCA). Side-channel attacks are built on the fact that cryptographic algorithms are implemented on a physical device. SCA can use all kinds of physical emanation from the device, like current consumption, electromagnetic radiation, or execution time variations. This so-called side-channel may leak information about secret data. Even the regular output of the algorithm can be seen as a side-channel—in the case that an attacker was able to induce a fault into the data or control path of the computation. Although there is vast literature about SCA on implementations of block ciphers and public key algorithms, only few publications can be found about attacks on stream ciphers. In [4] the authors study fault attacks on stream ciphers like LILI-128, RC4, and SOBER-t32. The

M. Abe (Ed.): CT-RSA, LNCS 4377, pp. 257–270, 2007.

latter one was the target of a timing attack in [8]. Template attacks, which were introduced in [2], were mounted on RC4 in [9]. So far, there are no reports on a practical DPA targeting a hardware implementation of a stream cipher. There is only one work, [7], which describes theoretically DPA attacks on A5/1 and E0. These are classical DPA attacks which aim at raising the side-channel signal above the algorithmic noise[1] by statistical means (by averaging over many power traces). In contrast, the presented method cancels out the algorithmic noise exactly, by using specially tailored sets of initial value vectors.

Differential power analysis was introduced by Kocher in [5]. In a DPA an attacker generates a set of hypotheses (about some secret value or a partial key) and tries to identify the (unique) true hypothesis by finding the highest correlation between the power consumption of the physical realization of an algorithm and those internal bits which can be computed by the attacker by virtue of one of these hypotheses. The classical setup for a DPA is illustrated in Figure 1. Some parts S of an implementation of a cryptographic algorithm have the characteristic:

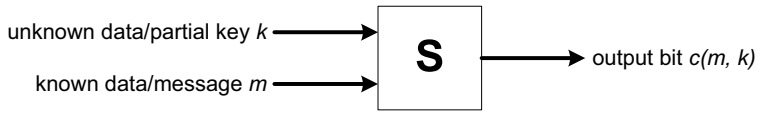

Fig. 1. Generic Setup for DPA

- input: known bits m, a few unknown bits k
- output: a bit $c(m, k)$ with the properties
  - $c(m, k) = c(m, k')$ for all possible m and $k = k'$
  - $c(m, k) = c(m, k')$ for about 50% of all possible values of m if $k \neq k'$—even if k and k' differ only by one bit.

This implies, of course, that S cannot be linear. But in any good cipher one is always able to identify some parts with this property.

Our attack targets the resynchronization phase of stream ciphers. Unlike in [7], were a *known IV DPA attack* is described, we will describe and execute a *chosen IV DPA attack*. It will be shown that the signal-to-noise ratio of the side-channel signal, which carries information about the secret key, can be optimized by specially chosen initial value vectors. These are constructed in such a way, that in the statistical analysis of the power traces contributions to the power consumption, which are not related to the correlation signal, will cancel out. We will elaborate the attacks for two recently published stream ciphers, Grain [3] and Trivium [1]. In the case of the first cipher, we will make partial use of a nonlinear element S. In the case of the second cipher we will *not* use

[1] Algorithmic noise is generated by the power consumption caused by the execution of the algorithm. Contrary to thermal noise it cannot be eliminated by averaging over power traces with identical input parameters.

any nonlinearity. S will rather be an XOR gate. By virtue of the new selection scheme for the initial value vectors we are able to mount practical attacks on hardware implementations of the two ciphers. The attack is efficient in practice, as there is no need to construct templates. Also the number of samples is small.

Outline—We will describe the attacks on both stream ciphers. In each case we will first give a definition of the cipher and shortly describe a straight forward hardware implementation, in order to state a theoretical power model for this implementation. We will describe the actual attack on the cipher and show why the attack works in our chosen power model. Finally the attack on a physical implementation of Grain on a field programmable gate array device (FPGA) will be reported.

2 Differential Power Analysis of Grain

The target of the attack will be the second version of Grain [3]. After a description of the structure and the implementation of the cipher the power model for a CMOS implementation will be defined and the theory for the attack will be elaborated. Finally we will report the results of a practical realization of the attack on a hardware implementation.

2.1 Definition of Grain

Grain is a binary additive synchronous stream cipher with an internal state of 160 bits $s_i, s_{i+1}, \ldots, s_{i+79}$ and $b_i, b_{i+1}, \ldots, b_{i+79}$ residing in a linear feedback shift register (LFSR) and a nonlinear feedback shift register (NLFSR), respectively. It supports a key $k = (k_0, \ldots, k_{79})$ of 80 bits and an initial value $IV = (IV_0, \ldots, IV_{63})$ of 64 bits. After a run-up time of 160 iteration steps it outputs a key stream z_i. During run-up the output bits z_i $(0 \leq i < 160)$ will not be used, but fed back into the LFSR and NLFSR components. Run-up (for $0 \leq i < 160$) and output generation (for $160 \leq i$) is described by the following recursion formula:

$$
\begin{aligned}
(b_0, \ldots, b_{79}) &:= (k_0, \ldots, k_{79}) \\
(s_0, \ldots, s_{79}) &:= (IV_0, \ldots, IV_{63}, 1, \ldots, 1) \\
g_i &:= g(b_{i+0}, b_{i+1}, \ldots, b_{i+63}) \\
f_i &:= s_{i+0} + s_{i+13} + s_{i+23} + s_{i+38} + s_{i+51} + s_{i+62} \\
\tilde{\sigma}_i &:= b_{i+1} + b_{i+2} + b_{i+4} + b_{i+10} + b_{i+31} + b_{i+43} \\
\sigma_i &:= \tilde{\sigma}_i + b_{i+56} \\
z_i &:= \sigma_i + h(s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64}, b_{i+63}) \\
b_{i+80} &:= g_i + s_i + z_i \cdot \delta_{[0,159]}(i) \\
s_{i+80} &:= f_i + z_i \cdot \delta_{[0,159]}(i)
\end{aligned}
$$

All variables represent elements of the binary field $\mathbb{F}_2$. $g\colon \mathbb{F}_2^{64} \to \mathbb{F}_2$ is a nonlinear function which we will not describe any further, since the exact form is not essential for the attack. The function h is defined by

$$\begin{aligned} h\colon \mathbb{F}_2^5 &\longrightarrow \mathbb{F}_2 \\ (x_0, \ldots, x_4) &\longmapsto x_1 + x_4 + x_0x_3 + x_2x_3 + x_3x_4 + \\ &\qquad x_0x_1x_2 + x_0x_2x_3 + x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4. \end{aligned}$$

The indicator function $\delta_{[0,159]}(i)$ is 1 for $0 \le i \le 159$ and 0 otherwise. In addition to the specification in [3] we introduced the two intermediary values $\tilde{\sigma}_i$ and σ_i. They are not essential for the definition of the cipher. However, these values will be a part of our hypothesis.

Notation 1. *For the whole paper we will fix the secret key k. Besides k, the sequences b_i, s_i, g_i, f_i, $\tilde{\sigma}_i$, σ_i, z_i will depend on the initial value. Therefore we will often write $b_i^{(\nu)}$, $s_i^{(\nu)}$, ..., for $IV = \nu$.*

Remark 1. For $i = 0, \ldots, 16$, the elements $b_{i+63}^{(\nu)}$, $\sigma_i^{(\nu)}$, $\tilde{\sigma}_{i+17}^{(\nu)}$, $g_i^{(\nu)}$ do *not* depend on the initial value ν, but only on the key k.

2.2 Implementation in Hardware

A structural view of the hardware implementation is given in Fig. 2.

It consists of the following parts:

- an LFSR of 80 flip-flops $L_0, \ldots, L_{79}$, holding the values $s_i, \ldots, s_{i+79}$
- an NLFSR of 80 flip-flops $N_0, \ldots, N_{79}$, holding the values $b_i, \ldots, b_{i+79}$
- a combinatorial logic block G, realizing the function g
- a combinatorial logic block F, realizing the function f
- a combinatorial logic block H, realizing the function H
- some additional XORs.

Of course there is also some control logic for loading the key and initial value, clocking the FSRs, and switching δ.

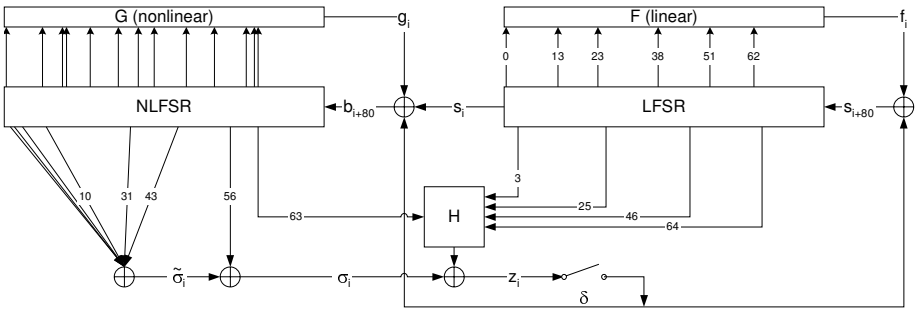

Fig. 2. Implementation of Grain

2.3 Power Model

We will use a discrete, Hamming distance based power model to describe the power consumption, since this suites the power consumption of a CMOS implementation very well. For a fixed key k and an initial value ν the power consumption of Grain is a function

$$P\colon \mathbb{N}_0 \longrightarrow \mathbb{R},$$

where $P(i)$ is the integral over the power consumption during the i-th clock cycle. The i-th clock cycle is the period of time, when the values g_i, f_i, $\tilde{\sigma}_i$, σ_i, z_i, b_{i+80}, s_{i+80} are evaluated and the two FSRs are shifted. Therefore we can write

$$P = P_{\mathrm{G}} + P_{\mathrm{H}} + P_{\mathrm{F}} + \sum_{j=0}^{79} P_{\mathrm{FF},N_j} + \sum_{j=0}^{79} P_{\mathrm{FF},L_j} + \Omega,$$

where P_{G}, P_{H}, P_{F} and P_{FF} denote the power consumption of G, H (including the generation of σ_i), F, and a flip-flop, respectively. Ω describes the noise which is independent of the described architectural elements. It is reasonable to model P_{G}, P_{H}, P_{F} and P_{FF} in a way, such that they only depend on the old and new input values:

$$\begin{aligned} P(i) = {} & P_{\mathrm{G}}(b_{i-1}, b_i, \ldots, b_{i+63}) \\ & + P_{\mathrm{H}}(b_i, \ldots, b_{i+63}, s_{i+2}, s_{i+3}, s_{i+24}, s_{i+25}, s_{i+45}, s_{i+46}, s_{i+63}, s_{i+64}) \\ & + P_{\mathrm{F}}(s_{i-1}, s_i, s_{i+12}, s_{i+13}, s_{i+22}, s_{i+23}, \ldots, s_{i+61}, s_{i+62}) \\ & + \sum_{j=0}^{79} P_{\mathrm{FF},N_j}(b_{i+j}, b_{i+j+1}) + \sum_{j=0}^{79} P_{\mathrm{FF},L_j}(s_{i+j}, s_{i+j+1}) + \Omega. \end{aligned}$$

Note, that this equation may not be fully correct for $i = 0$, since the "old" values may not always exist (e.g. b_{-1}) or could have some default values (after resetting the circuit). In this case the corresponding constant values must be used. We will make *no* further assumption about the functions $P_{\mathrm{G}}\colon \mathbb{F}_2^{65} \to \mathbb{R}$ and $P_{\mathrm{H}}\colon \mathbb{F}_2^{64+8} \to \mathbb{R}$, since this would add an unnecessary difficulty. It turns out that the precise form will not be needed. We define that $P_{\mathrm{F}}\colon \mathbb{F}_2^{12} \to \mathbb{R}$ is only a function of the Hamming distances of consecutive bits of the LFSR:

$$P_{\mathrm{F}}(s_{i-1}, s_i, \ldots, s_{i+61}, s_{i+62}) \equiv P_{\mathrm{F}}(s_{i-1} \oplus s_i, \ldots, s_{i+61} \oplus s_{i+62}).$$

For P_{FF} we make the usual approximation

$$P_{\mathrm{FF}}(0,0) \approx 0 \approx P_{\mathrm{FF}}(1,1) \ll P_{\mathrm{FF}}(1,0), P_{\mathrm{FF}}(0,1).$$

We do *not* assume that all P_{FF,N_j} or all P_{FF,L_j} are equal, as this cannot be expected to hold in an arbitrary implementation. Ω contains all noise contributions which are independent of the key and the initial value, such as the noise generated by the control hardware of the cipher or switching activity of circuits in the environment.

Notation 2. *For a fixed key k the whole cipher depends on the initial values ν. $P^{(\nu)}$, $P_G^{(\nu)}$, ..., denote the respective power consumption functions. $P^{(\nu)}$ will still be variable because of Ω. Therefore we will use its expectation value*

$$\bar{P}^{(\nu)} := \mathbb{E}P^{(\nu)},$$

which can be approximated by measuring the power consumption for the same initial value ν several times and taking the arithmetic mean value. Now $\bar{P}^{(\nu)}$ depends on the initial value ν only.

2.4 Attack on Grain: Theory

The attack on Grain consists of three steps. The first two steps are differential power analyses gaining information of 34 and 16 bits of the key, respectively. The third step is an exhaustive search on the remaining 30 bits of the key.

Step 1. is a DPA with chosen IVs. It is done in 17 rounds. In the i-th round ($0 \le i \le 16$) we set up our hypothesis (b^h_{i+63}, σ^h_i) about the pair (b_{i+63}, σ_i) and try to verify the true hypothesis by using the recorded power traces of the key setup phase for several initial values $\nu \in \mathcal{IV}_i$. The set $\mathcal{IV}_i$ of initial values is tailored in such a way, that the intrinsic power consumption of Grain will cancel out when computing the difference of the power traces (the "correlation function"). In each round the results of the previous ones will be used. Step 1 is illustrated in Table 1. Note that (b^h_{i+63}, σ^h_i) as well as ν must be used in order to

Table 1. DPA Attack on Grain, Step 1

For $i := 0$ **to** 16 **do**
  For all hypotheses $(b^h_{i+63}, \sigma^h_i) \in \mathbb{F}_2^2$ **do**
    Using hypothesis (b^h_{i+63}, σ^h_i) and the known $(b_{j+63}, \sigma_j)_{j=0}^{i-1}$, compute
      $\mathcal{IV}_i^+ := \{\nu \in \mathcal{IV}_i : s^{(\nu)}_{i+79} \ne s^{(\nu)}_{i+80}\}$, $\mathcal{IV}_i^- := \{\nu \in \mathcal{IV}_i : s^{(\nu)}_{i+79} = s^{(\nu)}_{i+80}\}$
    and the "correlation function"
      $$\bar{P}_{(b^h_{i+63}, \sigma^h_i)} := \frac{1}{\#\mathcal{IV}_i^+} \sum_{\nu \in \mathcal{IV}_i^+} \bar{P}^{(\nu)} - \frac{1}{\#\mathcal{IV}_i^-} \sum_{\nu \in \mathcal{IV}_i^-} \bar{P}^{(\nu)}$$
  end
  Accept the hypothesis, for which $\bar{P}_{(b^h_{i+63}, \sigma^h_i)}(i)$ is maximal.
end

compute the (hypothetical) value $s^{(\nu)}_{i+80}$. For computing $s^{(\nu)}_{i+79}$ the already known $(b_{i-1+63}, \sigma_{i-1})$ is used. The families $\mathcal{IV}_i$ (for $0 \le i \le 16$) of initial values are defined as follows:

$$\mathcal{IV}_i := \left\{ (\nu_0, \ldots, \nu_{63}) \in \mathbb{F}_2^{64} : \nu_n = \begin{cases} 0, & \text{for } n - i \ne 3, 13, 22, 23, 25, 46, \\ 1, & \text{for } n = i + 46. \end{cases} \right\}$$

$\mathcal{IV}_i$ contains 32 initial values which toggle the bits ν_{i+3}, ν_{i+13}, ν_{i+22}, ν_{i+23}, ν_{i+25} and set ν_{i+46} to 1.

Remark 2. In our case, we have $\#\mathcal{IV}_i^+ = \#\mathcal{IV}_i^- = 16$, for all $0 \le i \le 16$.

A justification for this algorithm and the choice of IVs is given in the following lemma.

Lemma 1. *In round i ($0 \le i \le 16$) of the above algorithm we have:*

$$\begin{aligned}
P_{(b_{63},\sigma_0)}(0) &= \frac{1}{2}(P_{FF,L_{79}}(1,0) - P_{FF,L_{79}}(1,1)),\\
P_{(b_{i+63},\sigma_i)}(i) &= \frac{1}{2}(P_{FF,L_{79}}(0,1) + P_{FF,L_{79}}(1,0)\\
&\quad -P_{FF,L_{79}}(0,0) - P_{FF,L_{79}}(1,1)),\ \textit{for } 1 \le i,\\
P_{(b_{i+63},1+\sigma_i)}(i) &= -P_{(b_{i+63},\sigma_i)}(i),\\
P_{(1+b_{i+63},\sigma_i)}(i) &= P_{(1+b_{i+63},1+\sigma_i)}(i) = 0.
\end{aligned}$$

For an explanation of this lemma cf. the paragraph after Lemma 2.

Remark 3. Of course, this DPA also works for families $\mathcal{IV}_i$ of randomly chosen initial values. However, the algorithmic noise level caused by the remaining 159 flip-flops (other than L_{79}) will be higher compared to the correlation signal. As a consequence the number of necessary samples $\#\mathcal{IV}_i$ for each family would be larger. The same is true for Step 2.

Step 2. works similarly to the first step, but only in 16 rounds. The hypotheses will be $(g_{i-17}^h, \tilde{\sigma}_i^h)$, for $17 \le i \le 32$. Moreover, the construction of the $\mathcal{IV}_i$ will make use of the previously learned data. The second step of the attack is given

Table 2. DPA Attack on Grain, Step 2

For $i := 17$ **to** 32 **do**
  For all hypotheses $(g_{i-17}^h, \tilde{\sigma}_i^h) \in \mathbb{F}_2^2$ **do**
    Using hypothesis $(g_{i-17}^h, \tilde{\sigma}_i^h)$ and the already known $(b_{j+63}, \sigma_j)_{j=0}^{16}$, $(g_{i-17}, \tilde{\sigma}_i)_{j=17}^{i-1}$, compute the partition of $\mathcal{IV}_i$
      $\mathcal{IV}_i^+ := \{\nu \in \mathcal{IV}_i : s_{i+79}^{(\nu)} \ne s_{i+80}^{(\nu)}\}$, $\mathcal{IV}_i^- := \{\nu \in \mathcal{IV}_i : s_{i+79}^{(\nu)} = s_{i+80}^{(\nu)}\}$
    and the "correlation function"
      $\bar{P}_{(g_{i-17}^h, \tilde{\sigma}_i^h)} := \frac{1}{\#\mathcal{IV}_i^+} \sum_{\nu \in \mathcal{IV}_i^+} \bar{P}^{(\nu)} - \frac{1}{\#\mathcal{IV}_i^-} \sum_{\nu \in \mathcal{IV}_i^-} \bar{P}^{(\nu)}$
  end
  Accept the hypothesis, for which $\bar{P}_{(g_{i-17}^h, \tilde{\sigma}_i^h)}(i)$ is maximal.
end

in Table 2. Note that $(g_{i-17}^h, \tilde{\sigma}_i^h)$, $(b_{j+63}, \sigma_j)_{j=0}^{16}$, $(g_{j-17}, \tilde{\sigma}_j)_{j=17}^{i-1}$ as well as ν must be used in order to compute the (hypothetical) value $s_{i+80}^{(\nu)}$. For computing $s_{i+79}^{(\nu)}$, the already known values $(b_{j+63}, \sigma_j)_{j=0}^{16}$, $(g_{j-17}, \tilde{\sigma}_j)_{j=17}^{i-1}$ are used. The families

$\mathcal{IV}_i$ (for $17 \le i \le 32$) of initial values are defined as follows:

$$\mathcal{IV}_i := \Big\{ \quad (\nu_0, \ldots, \nu_{63}) \in \mathbb{F}_2^{64} : (\nu_{i+3}, \nu_{i+13}, \nu_{i+22}, \nu_{i+23}, \nu_{i+25}) \in \mathbb{F}_2^5,$$

$$\nu_n = \begin{cases} \nu_{i+22}, & \text{if } n = i - 16 \wedge 16 \le i, \\ b_{i+44}, & \text{if } n = i + 6 \wedge 19 \le i, \\ 1, & \text{if } n = i + 27 \wedge 19 \le i < 37, \\ 0, & \text{if } n = i + 1 \wedge 24 \le i, \\ 1, & \text{if } n = i - 21 \wedge 24 \le i, \\ 1, & \text{if } n = 63 \wedge 17 = i, \\ 0, & \text{all other } n \in \{0, \ldots, 63\} \setminus \{3, 13, 22, 23, 25\} \end{cases} \Big\}.$$

Remark 4. Also in this case $\#\mathcal{IV}_i^+ = \#\mathcal{IV}_i^- = 16$, for all $17 \le i \le 32$.

The next lemma gives the justification for the algorithm.

Lemma 2. *In round i ($16 \le i \le 32$) of the above algorithm we have:*

$$\begin{aligned} P_{(g_{i-17}, \tilde{\sigma}_i)}(i) &= \frac{1}{2}(P_{FF,L_{79}}(0,1) + P_{FF,L_{79}}(1,0) \\ &\quad - P_{FF,L_{79}}(0,0) - P_{FF,L_{79}}(1,1)), \\ P_{(g_{i-17}, 1+\tilde{\sigma}_i)}(i) &= -P_{(g_{i-17}, \tilde{\sigma}_i)}(i), \\ P_{(1+g_{i-17}, \tilde{\sigma}_i)}(i) &= P_{(1+g_{i-17}, 1+\tilde{\sigma}_i)}(i) = 0. \end{aligned}$$

The proofs of the last two lemmata are straightforward, but involve rather lengthy calculations. They are preferably supported by an algebraic software package. The families of $\mathcal{IV}_i$ of initial value vectors are constructed such that the following properties hold:

- The toggling of the bits s_{i+13} and s_{i+23} directly causes a toggling of s_{i+80}, but not of s_{i+79}. This distributes the power functions $\bar{P}^{(\nu)}$, $\nu \in \mathcal{IV}_i$ to the two sums in the "correlation function" in a way, such that all power contributions not depending on s_{i+13} and s_{i+23} will cancel out.
- The toggling of the bit s_{i+22} has the same effect by influencing s_{i+79}.
- Function $h(x_0, x_1, x_2, x_3, x_4)$ has the following property: If $x_2 = x_3 = 1$ and $(x_0, x_1) \in \mathbb{F}_2^2$, changing x_4 to its complement results in a change of h in exactly 50% of the cases. This is the classical assumption for a bit to be used in a DPA.
- The additional conditions in the definition of $\mathcal{IV}_i$, for $17 \le i \le 32$ have the following reason. Toggling of bit s_{i+22} does not result in a toggling of other bits which also depend on the bits s_{i+13} and s_{i+23}, asserting that the first two facts are orthogonal.

Remark 5. In a practical attack on a hardware implementation the characteristics described in Lemmata 1 and 2 will transform into a peak for correct hypotheses (b_{i+63}, σ_i) or $(g_i, \tilde{\sigma}_i)$, and a negative peak for hypotheses with reversed σ_i or $\tilde{\sigma}_i$. Each of the other two hypotheses should not show a significant peak. During the following 79 clock cycles peaks can still be expected because the correlating signal remains.

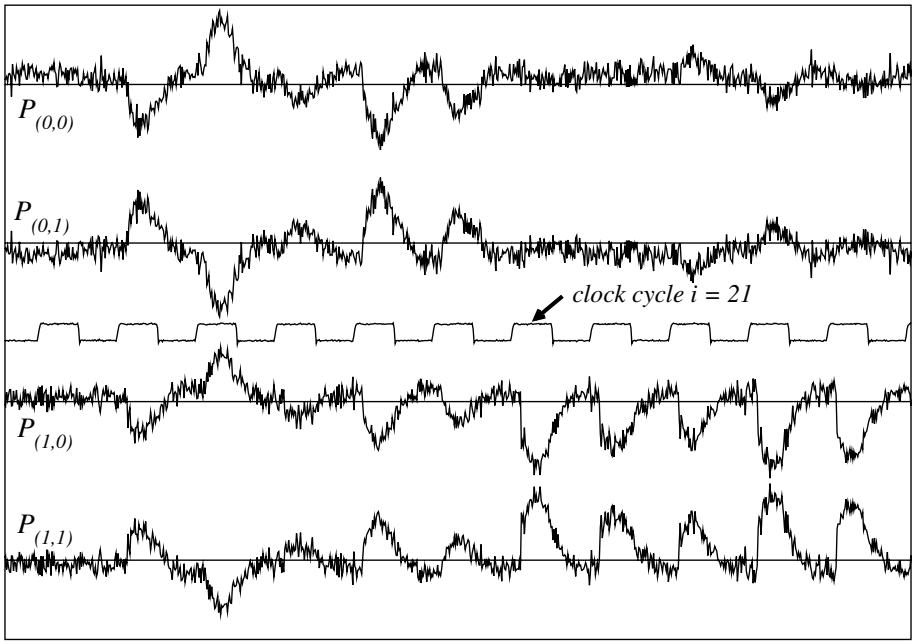

Fig. 3. The correlation functions $\bar{P}_{(0,0)}$, $\bar{P}_{(0,1)}$, $\bar{P}_{(1,0)}$, $\bar{P}_{(1,1)}$ for round 21 in the DPA attack on Grain

Step 3. is now straight forward. After having obtained 50 values $b_{63}, \ldots, b_{79}$, $\sigma_0, \ldots, \sigma_{16}$, $\tilde{\sigma}_{17}, \ldots, \tilde{\sigma}_{32}$, fifty independent linear equations in $k_0, \ldots, k_{79}$ can be written down. Solving these equations a linear map $\kappa\colon \mathbb{F}_2^{30} \to \mathbb{F}_2^{80}$ is obtained, such that the image of κ contains all possible remaining keys. Hence an exhaustive key search can be performed in practice. The complexity of this final step is $O(2^{30})$. The whole key can be recovered with, e.g., one appropriate plain text/cipher text pair. The attack can be improved by using the additional information contained in $(g_0, \ldots, g_{15})$ to exclude more keys.

2.5 Attack on Grain: Practical Realization

We implemented a version of Grain which generates one bit of key stream per clock cycle. This is probably the realization most relevant for hardware constrained environments. We chose an implementation on an Altera FLEX EPF10K100ARC240-2 FPGA as our target of attack. In a standard measurement setup the voltage drop at a shunt in the power supply line of the FPGA was measured. The FPGA was operated at 2.5 MHz and the power traces were recorded using a LeCroy LC684DXL oscilloscope with a sample rate of 2 Giga samples per second. A set of 256 power traces for each initial value in each family $\mathcal{IV}_i$ was obtained. The corresponding sample averages $\bar{P}^{(\nu)}$ were used to verify or falsify the hypotheses. As an example, in Figure 3 the four correlation

functions $\bar{P}_{(g^h_{i-17},\tilde{\sigma}^h_i)}$ for $i = 21$ and $(g^h_{i-17}, \tilde{\sigma}^h_i) \in \{(0,0),(0,1),(1,0),(1,1)\}$ are shown. As indicated by the arrow at clock cycle 21 the onset of peaks clearly verifies the correct hypothesis $(1,1)$.

3 Differential Power Analysis of Trivium

In this section we describe a DPA attack on the stream cipher Trivium [1]. This attack is not based on any nonlinear part, but correlations with the power consumption of the three flip-flops B_{81}, B_{82} and B_{83} are exploited. These flip-flops lie behind an XOR gate, which mixes known and controllable bits with secret bits. Again we are able to recover the whole key.

3.1 Definition of Trivium

Trivium is a stream cipher with an internal state of 288 bits $a_i, a_{i+1}, \ldots, a_{i+92}$, $b_i, b_{i+1}, \ldots, b_{i+83}$ and $c_i, c_{i+1}, \ldots, c_{i+110}$—residing in three coupled feedback shift registers A, B, and C of 93, 84, and 111 bits respectively—using a key $k = (k_0, \ldots, k_{79})$ of 80 bits as well as an initial value $IV = (IV_0, \ldots, IV_{79})$ of 80 bits. After a run-up time of $4 \cdot 288$ iteration steps it outputs a key stream z_i. Run-up (for $0 \leq i < 4 \cdot 288$) and output generation (for $4 \cdot 288 \leq i$) can be described by the following recursion formula:

$$\begin{aligned}
(a_0, \ldots, a_{92}) &:= (0, \ldots, 0, k_{79}, \ldots, k_0) \\
(b_0, \ldots, b_{83}) &:= (0, 0, 0, 0, IV_{79}, \ldots, IV_0) \\
(c_0, \ldots, c_{110}) &:= (1, 1, 1, 0, \ldots, 0) \\
a_{i+93} &:= a_{i+24} + c_i + c_{i+1}c_{i+2} + c_{i+45} \\
b_{i+84} &:= b_{i+6} + a_i + a_{i+1}a_{i+2} + a_{i+27} \\
c_{i+111} &:= c_{i+24} + b_i + b_{i+1}b_{i+2} + b_{i+15} \\
z_i &:= a_i + b_i + c_i + a_{i+27} + b_{i+15} + c_{i+45}
\end{aligned}$$

All variables represent elements in $\mathbb{F}_2$. The intermediate value $\sigma_i := a_i + a_{i+1}a_{i+2} + a_{i+27}$ will be our hypothesis in the DPA.

3.2 Implementation in Hardware

Again, the target of attack will be an implementation of the cipher which generates one bit of key stream per clock cycle. It comprises the following parts: An NLFSR of 93 flip-flops $A_0, \ldots, A_{92}$, holding the values $a_i, \ldots, a_{i+92}$, an NLFSR of 84 flip-flops $B_0, \ldots, B_{83}$, holding the values $b_i, \ldots, b_{i+83}$, an NLFSR of 111 flip-flops $C_0, \ldots, C_{110}$, holding the values $c_i, \ldots, c_{i+110}$, three additional AND, and a few XOR gates (as given in the recursion formula), as well as additional control logic for loading the key and initial value, and clocking the NLFSRs.

3.3 Power Model

We will use the same power model and notation as in the previous attack. The model for the power consumption is

$$P = \sum_{j=0}^{92} P_{\mathrm{FF},A_j} + \sum_{j=0}^{83} P_{\mathrm{FF},B_j} + \sum_{j=0}^{110} P_{\mathrm{FF},C_j} + \Omega,$$

as well as

$$P(i) = \sum_{j=0}^{92} P_{\mathrm{FF},A_j}(a_{i+j}, a_{i+j+1}) + \sum_{j=0}^{83} P_{\mathrm{FF},B_j}(b_{i+j}, b_{i+j+1}) + \sum_{j=0}^{110} P_{\mathrm{FF},C_j}(c_{i+j}, c_{i+j+1}) + \Omega.$$

Furthermore, we make the same assumption regarding the power consumption of the flip-flops as in the previous attack. To simplify the description we ignore the power consumption of the single AND and XOR gates. The notations, like $\bar{P}^{(\nu)}$, will also be used equivalently.

3.4 Attack on Trivium: Theory

To simplify the description we make the assumption that all flip-flops in our power model have the same power characteristic, i.e., for all appropriate j:

$$e_{xy} = P_{\mathrm{FF},A_j}(x,y) = P_{\mathrm{FF},B_j}(x,y) = P_{\mathrm{FF},C_j}(x,y)$$

with some constants $e_{00} \approx 0 \approx e_{11} \ll e_{01}, e_{10}$. This restriction, however, is not necessary to mount the attack.

The DPA is done in 76 rounds. In the i-th round we will know already $(\sigma_j)_{j=0}^{i-1}$ and evaluate σ_i. In fact, we will not need to make any hypothesis. The attack is

Table 3. DPA on Trivium

For $i := 3$ **to** 78 **except** 10 **do**
  Using the knowledge of $(\sigma_j)_{j=0}^{i-1}$, compute
    $\mathcal{IV}_i^+ := \{\nu_+\} := \{\nu \in \mathcal{IV}_i : b_{i+83}^{(\nu)} = 0\}$, $\mathcal{IV}_i^- := \{\nu_-\} := \{\nu \in \mathcal{IV}_i : b_{i+83}^{(\nu)} = 1\}$
  and
    $\bar{P}_i := \frac{1}{\#\mathcal{IV}_i^+} \sum_{\nu \in \mathcal{IV}_i^+} \bar{P}^{(\nu)} - \frac{1}{\#\mathcal{IV}_i^-} \sum_{\nu \in \mathcal{IV}_i^-} \bar{P}^{(\nu)}$
  if $b_{i+81}^{(\nu_+)} + b_{i+82}^{(\nu_+)} + b_{i+83}^{(\nu_+)} \equiv 0 \pmod 2$ **then**
    if $\bar{P}_i(i) > -(e_{01} + e_{10})/2$ **then** $\sigma_i := 1$ **else** $\sigma_i := 0$
  else
    if $\bar{P}_i(i) > (e_{01} + e_{10})/2$ **then** $\sigma_i := 1$ **else** $\sigma_i := 0$
end

illustrated in the following Table 3. We will assume, that the values σ_0, σ_1, σ_2, and σ_{10} are already known (in this case, one may make an "external" hypothesis on these 4 bits).[2]

The families $\mathcal{IV}_i$ (for $3 \le i \le 78$) of initial values are defined as follows:

$$\mathcal{IV}_i := \left\{(\nu_0, \ldots, \nu_{79}) \in \mathbb{F}_2^{80} : \nu_{78-i} = 1 + \nu_{79-i}, \nu_n := 0 \text{ otherwise}\right\}.$$

Note, that $\mathcal{IV}_i$ contains only 2 values. A justification for this algorithm and the choice of IVs is given in the following:

Lemma 3. *For any i, with $3 \le i \le 78$, $i \ne 10$, we have: (i) Writing $\mathcal{IV}_i = \{\nu_1, \nu_2\}$, then $b_{i+81}^{(\nu_1)} = b_{i+81}^{(\nu_2)}$ and $b_{i+82}^{(\nu_1)} + b_{i+83}^{(\nu_1)} \equiv b_{i+82}^{(\nu_2)} + b_{i+83}^{(\nu_2)} \pmod 2$, therefore the respective index "ν_+", in the above algorithm, can be left out. (ii) For $\bar{P}_i(i)$ we have the values:*

$(b_{i+81} + b_{i+82} + b_{i+83}) \bmod 2$	b_{i+81}	$(b_{i+82} + b_{i+83}) \bmod 2$	b_{i+84}	$\bar{P}_i(i)$		approx.
0	0	0	0	$2e_{00} - 2e_{11}$	$\approx$	0
0	0	0	1	$3e_{00} - e_{01} - e_{10} - e_{11}$	$\approx$	$-(e_{01} + e_{10})$
0	1	1	0	0	$\approx$	0
0	1	1	1	$e_{00} + e_{11} - e_{01} - e_{10}$	$\approx$	$-(e_{01} + e_{10})$
1	0	1	0	$e_{01} + e_{10} + e_{00} - 3e_{11}$	$\approx$	$(e_{01} + e_{10})$
1	0	1	1	$2e_{00} - 2e_{11}$	$\approx$	0
1	1	0	0	$e_{01} + e_{10} - e_{00} - e_{11}$	$\approx$	$(e_{01} + e_{10})$
1	1	0	1	0	$\approx$	0

Remark 6. In a practical attack on a hardware realization, by virtue of Lemma 3, the two inequalities will transform into the decisions {no peak↔negative peak} and {positive peak↔no peak}. The boundary $(e_{01} + e_{10})/2$ was just used for illustration purposes.

Extraction of the key: After gaining the 79 values $(\sigma_i)_{i=0}^{78}$ (possibly depending on the 4 hypothetical values σ_0, σ_1, σ_2, and σ_{10})we can write down the equations $\sigma_i = a_i + a_{i+1}a_{i+2} + a_{i+24}$ for $0 \le i \le 78$. These are 79 equations with 80 indeter-

$\sigma_0 = k_{65}$, $\sigma_1 = k_{64}$	$\ldots$	$\sigma_{11} = k_{54}$,
$\sigma_{12} = k_{79}k_{78} + k_{53}$,		
$\sigma_{13} = k_{79} + k_{78}k_{77} + k_{52}$,	$\ldots$	$\sigma_{65} = k_{27} + k_{26}k_{25} + k_0$,
$\sigma_{66} = k_{26} + k_{25}k_{24} + k_{68}$,		
$\sigma_{67} = k_{25} + k_{24}k_{23} + k_{67} + 1$,		
$\sigma_{68} = k_{24} + k_{23}k_{22} + k_{66}$,	$\ldots$	$\sigma_{78} = k_{14} + k_{13}k_{12} + k_{56}$,

minates $(k_i)_{i=0}^{79}$, which are shown explicitly in the following table. One equation is dependent on the others. For solving the system of equations, we may assume any value in $\mathbb{F}_2$ for k_{12} and k_{13}. By reordering the equations in the following Table —and leaving out the equation for σ_{12}—we can solve one equation after

[2] There is an other DPA strategy for evaluating $\sigma_0, \ldots, \sigma_{15}$. For lack of space we omit this evaluation phase.

$\sigma_0 = k_{65}$, $\sigma_1 = k_{64}$,	...	$\sigma_{11} = k_{54}$,	$\rightsquigarrow k_{65}, \ldots, k_{54}$
$\sigma_{27} = k_{65} + k_{64}k_{63} + k_{38}$,	...	$\sigma_{36} = k_{56} + k_{55}k_{54} + k_{29}$,	$\rightsquigarrow k_{38}, \ldots, k_{29}$
$\sigma_{54} = k_{38} + k_{37}k_{36} + k_{11}$,	...	$\sigma_{61} = k_{31} + k_{30}k_{29} + k_4$,	$\rightsquigarrow k_{11}, \ldots, k_4$
$\sigma_{78} = k_{14} + k_{13}k_{12} + k_{56}$,	...	$\sigma_{69} = k_{23} + k_{22}k_{21} + k_{65}$,	$\rightsquigarrow k_{14}, \ldots, k_{23}$
$\sigma_{53} = k_{39} + k_{38}k_{37} + k_{12}$,	...	$\sigma_{42} = k_{50} + k_{49}k_{48} + k_{23}$,	$\rightsquigarrow k_{39}, \ldots, k_{50}$
$\sigma_{26} = k_{66} + k_{65}k_{64} + k_{39}$,	...	$\sigma_{15} = k_{77} + k_{76}k_{75} + k_{50}$,	$\rightsquigarrow k_{66}, \ldots, k_{77}$
$\sigma_{68} = k_{24} + k_{23}k_{22} + k_{66}$,	...	$\sigma_{66} = k_{26} + k_{25}k_{24} + k_{68}$,	$\rightsquigarrow k_{24}, \ldots, k_{26}$
$\sigma_{41} = k_{51} + k_{50}k_{49} + k_{24}$,	...	$\sigma_{39} = k_{53} + k_{52}k_{51} + k_{26}$,	$\rightsquigarrow k_{51}, \ldots, k_{53}$
$\sigma_{38} = k_{54} + k_{53}k_{52} + k_{27}$,		$\sigma_{37} = k_{55} + k_{54}k_{53} + k_{28}$,	$\rightsquigarrow k_{27}, k_{28}$
$\sigma_{14} = k_{78} + k_{77}k_{76} + k_{51}$,		$\sigma_{13} = k_{79} + k_{78}k_{77} + k_{52}$,	$\rightsquigarrow k_{78}, k_{79}$
$\sigma_{62} = k_{30} + k_{29}k_{28} + k_3$,	...	$\sigma_{65} = k_{27} + k_{26}k_{25} + k_0$,	$\rightsquigarrow k_3, \ldots, k_0$

the other, getting a full key for each previously chosen pair (k_{12}, k_{13}). Counting also the hypotheses σ_0, σ_1, σ_2, and σ_{10} we may get at most $2^6 = 64$ different possible keys. Finding the right one is now trivial.

4 Conclusion

In this paper we showed, that DPA attacks on stream ciphers are practically feasible and that they constitute a real threat. We presented efficient differential power analyses of two new stream ciphers, which are focus candidates of the eSTREAM project. In both cases the DPA works with chosen IVs. These are carefully chosen to eliminate the algorithmic noise. It is plausible that this kind of attack can be applied to many stream ciphers with a similar construction philosophy.

References

1. Ch. De Cannière and B. Preneel: Trivium Specifications, 2005. Available at `http://www.ecrypt.eu.org/stream/p2ciphers/trivium/trivium_p2.pdf`.
2. S. Chari, J. R. Rao, and P. Rohatgi: Template Attacks. In B. S. Kaliski Jr., Ç. K. Koç, and Ch. Paar, editors, *Chryptographic Hardware and Embedded Systems – CHES 2002*, Lecture Notes in Computer Science, vol. 2535, pp. 13–28, Springer-Verlag, 2002.
3. M. Hell, Th. Johansson, and W. Meier: Grain – A Stream Cipher for Constrained Environments, 2006. Available at `http://www.ecrypt.eu.org/stream/p2ciphers/grain/Grain_p2.pdf`.
4. J. Hoch and A. Shamir: Fault Analysis of Stream Ciphers. In M. Joye and J.-J. Quisquater, editors, *Chryptographic Hardware and Embedded Systems – CHES 2004*, Lecture Notes in Computer Science, vol. 3156, pp. 240–253, Springer-Verlag, 2004.
5. P. C. Kocher, J. Jaffe, and B. Jun: Differential Power Analysis. In M. J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, Lecture Notes in Computer Science, vol. 1666, pp. 388–397, Springer-Verlag, 1999.
6. S. Kumar, K. Lemke, and Ch. Paar: Some Thoughts about Implementation Properties of Stream Ciphers. In *SASC 2004 – The State of the Art of Stream Ciphers*, (Brugge, Belgium, October 14-15, 2004), Workshop Record, pp. 311–319. Available at `http://www.ecrypt.eu.org/stvl/sasc/record.html`.

7. J. Lano, N. Mentens, B Preneel, and I. Verbauwhede: Power Analysis of Synchronous Stream Ciphers with Resynchronization Mechanism. In *SASC 2004 – The State of the Art of Stream Ciphers*, (Brugge, Belgium, October 14-15, 2004), Workshop Record, pp. 327–333. Available at `http://www.ecrypt.eu.org/stvl/sasc/record.html`.
8. J. Lano and G. Peeters: Cryptanalyse van NESSIE kandidaten (Dutch), Master's thesis, K. U. Leuven, May 2002.
9. Ch. Rechberger: Side Channel Analysis of Stream Ciphers, Master's thesis, Institute for Applied Information Processing and Communications (IAIK), Graz University, 2004.
10. Ch. Rechberger and E. Oswald: Stream Ciphers and Side-Channel Analysis. In *SASC 2004 – The State of the Art of Stream Ciphers*, (Brugge, Belgium, October 14-15, 2004), Workshop Record, pp. 320–326. Available at `http://www.ecrypt.eu.org/stvl/sasc/record.html`.