

# On the Importance of Protecting $\Delta$ in SFLASH against Side Channel Attacks

Katsuyuki Okeya  
Hitachi, Ltd.  
Systems Development Laboratory  
Yokohama, Japan

ka-okeya@sdl.hitachi.co.jp

Tsuyoshi Takagi  
Technische Universität Darmstadt  
Fachbereich Informatik  
Darmstadt, Germany

takagi@informatik.tu-darmstadt.de

Camille Vuillaume  
Technische Universität Darmstadt  
Fachbereich Informatik  
Darmstadt, Germany

cvuillau@sdl.hitachi.co.jp

## Abstract

*SFLASH was chosen as one of the final selection of the NESSIE project in 2003. It is one of the most efficient digital signature scheme and is suitable for implementation on memory-constrained devices such as smartcards. Side channel attacks (SCA) are a serious threat to memory-constrained devices. If the implementation on them is careless, we are able to break the secret key. In this paper, we experimentally analyze the effectiveness of a side channel attack on SFLASH. There are two different secret keys for SFLASH, namely the proper secret key  $(s, t)$  and the random seed  $\Delta$  used for the hash function SHA-1. Whereas many papers discussed the security of  $(s, t)$ , little is known about that of  $\Delta$ . Steinwandt et al. proposed a theoretical DPA on finding  $\Delta$  by observing the XOR operations. We propose another DPA on  $\Delta$  using the addition operation modulo  $2^{32}$ , and present an experimental result of the DPA. After obtaining the secret key  $\Delta$ , the underlying problem of SFLASH can be reduced to the  $C^*$  problem broken by Patarin. From our simulation, about 1408 pairs of messages and signatures are needed to break SFLASH. Consequently, we have to carefully implement SHA-1 in order to resist SCA on SFLASH.<sup>1</sup>*

**Keywords:** Asymmetric Signature, SFLASH, Side Channel Attacks, SHA-1,  $C^*$ ,  $C^{*-}$ , Hidden Monomial Problem

---

<sup>1</sup>Very recently, Courtois et al. have proposed a new version of SFLASH, called SFLASHv3 [3]. However, the attacks described in this paper are applicable to the new version.

## 1. Introduction

In order to achieve higher security, the secret key is usually stored on smartcards. However, side channel attacks (SCA) enable to break the secret key on them, if their implementation is careless. We have to carefully treat with SCA. SFLASH is an efficient digital signature scheme suitable for the implementation on smartcards. Akkar et al. reported that its signature generation requires only about 60 ms on an Infineon SLE 66 [1]. The NESSIE project chose SFLASH in the final selection of cryptographic algorithms [4].

The security of SFLASH is based on the  $C^{*-}$  problem, which is a modification of the  $C^*$  problem (some public quadratic equations are hidden) [2, 15]. Patarin proposed an attack able to break the  $C^*$  problem by means of linear algebra techniques [14]. The signature function of SFLASH is a map  $(\mathbb{F}_{2^7})^{37} \rightarrow (\mathbb{F}_{2^7})^{37}$  which uses the secret keys  $(\Delta, s, t)$  and comprises (1) computing the hash value of a message with the secret key  $\Delta$  (Note that the first 27 elements of  $\mathbb{F}_{2^7}$  depend only on the message), (2) generating the signature of the message with the other secret keys  $(s, t)$ . In order to forge a signature, we need to compute step (2) without the knowledge of the secret keys  $(s, t)$ , namely we have to solve the  $C^{*-}$  problem. On the contrary, step (1) deals only with the secret key  $\Delta$  and is therefore independent from the  $C^{*-}$  problem. If the secret key  $\Delta$  is broken, then step (2) is not secure, in other words, the  $C^{*-}$  problem is reduced to the  $C^*$  problem. The security of  $\Delta$  is strongly depending on the structure of SHA-1. Steinwandt et al. [16] described a side channel attack on SHA-1 in SFLASH. Klíma-Rosa [7] also described a side channel attack on SHA-1 in the RSAES-OAEP [17], and their attacks

are applicable to the SHA-1 portion of SFLASH. These attacks allow to recover the secret key  $\Delta$  by analyzing XOR operations of SHA-1.

In this paper, we demonstrate the first experimental results about a complete attack on SFLASH relying on side channel information leakage of the secret key  $\Delta$ . We observe the side channel information arisen from the addition modulo  $2^{32}$  related to  $\Delta$  in SHA-1. About 200,000 samples from the signing oracle enable us to break the secret key  $\Delta$ . Note that this number of oracle calls strongly depends on the computational environment, therefore it would not be unrealistic to foresee an attack requiring less oracle calls. Then, we implemented Patarin’s algebraic attack on  $C^*$  using the previous work of Patarin [14] and Kobitz [8], and we constructed a dummy signature function without the knowledge of the secret keys  $(s, t)$ . From our simulation, about 1408 pairs of messages and their signatures can generate the dummy signature function within a few hours. This algebraic attack can be achieved without asking the signing oracle — we can perform it totally off-line, only using the public key. If the  $C^*$  problem is obtained, under realistic conditions, Patarin’s attack makes it possible to forge a signature within a relatively short time. Therefore, once an attacker successfully breaks the secret key  $\Delta$ , SFLASH is no longer secure. The protection of  $\Delta$  against side channel attacks is a critical security issue for SFLASH: the implementers of SFLASH have to carefully consider the implementation of SHA-1.

## 2. Hidden Monomial Cryptosystems

Hidden monomial cryptosystems were first introduced by Imai and Matsumoto in [11]. In [14], Patarin broke the Imai-Matsumoto cryptosystem; soon after that, he proposed enhanced versions which, he assumed, were able to resist his cryptanalysis, but most of them were broken, too. However, one of the variants of the Imai-Matsumoto cryptosystem remains: SFLASH.

In this section, we describe a simplified version of the cryptosystem introduced by Imai and Matsumoto, then show how Patarin’s cryptanalysis can break it. Finally, the variants of the Imai-Matsumoto cryptosystem are shortly surveyed.

### 2.1. Imai-Matsumoto Cryptosystem

In [11], Matsumoto and Imai presented a fast signature scheme based on the following property: two secret affine transformations are “blinded” by a non-linear monomial map. Since the affine transformations “hide” the monomial map, a general denomination for such cryptosystems is “hidden monomial”.

In the case of the Imai-Matsumoto cryptosystem (IM), the two secret affine applications are  $s : K^n \rightarrow K^n$  and  $v : K^n \rightarrow K^n$  where  $K = \mathbf{F}_q$  and  $q$  is a power of two. Let  $\mathcal{L}$  be an extension of degree  $n$  of the finite field  $K$ ; then, given a basis  $\beta = (\beta_0, \dots, \beta_{n-1})$  of  $\mathcal{L}$ , a  $n$ -tuple  $x = (x_0, \dots, x_{n-1}) \in K^n$  can be expressed as  $X = \sum_{i=0}^{n-1} x_i \beta_i \in \mathcal{L}$ . Let  $\varphi(x = (x_0, \dots, x_{n-1}) \in K^n) \mapsto X = \sum_{i=0}^{n-1} x_i \beta_i \in \mathcal{L}$  be the transformation map from  $K^n$  to  $\mathcal{L}$  and  $\varphi^{-1}$  its inverse. The “blinding” monomial map of IM is  $F : A \in \mathcal{L} \mapsto A^h \in \mathcal{L}$ , where  $h = q^\theta + 1$ . The exponent  $h$  further satisfies  $\gcd(h, q^n - 1) = 1$ : in this case,  $F$  is a bijection and its inverse is  $F^{-1} : A \mapsto A^{h'}$ , where  $h' = h^{-1} \bmod q^n - 1$ . Because  $A \mapsto A^{q^\theta}$  is a linear map of  $\mathcal{L}$ ,  $F : A \mapsto A^{q^\theta} \cdot A$  is a quadratic map of  $\mathcal{L}$ .

Given a message  $y = (y_0, \dots, y_{n-1}) \in K^n$ , the signature is computed as follows:

$$x = s^{-1} \circ \varphi^{-1} \circ F^{-1} \circ \varphi \circ t^{-1}(y) = [G^*]^{-1}(y) \quad (1)$$

Then, the signature  $y = (y_0, \dots, y_{n-1}) \in K^n$  is valid if

$$t \circ \varphi^{-1} \circ F \circ \varphi \circ s(x) = G^*(x) = y \quad (2)$$

$G^*$  is a quadratic map as  $F$  is. Therefore,  $G^*$  can be written as:  $G^*(x_0, \dots, x_{n-1}) = (y_0, \dots, y_{n-1})$ , where

$$\begin{aligned} y_k &= P_k(x_0, \dots, x_{n-1}) \\ &= \sum_{0 \leq i < j < n} \rho_{k,i,j} x_i x_j + \sum_{0 \leq i < n} \sigma_{k,i} x_i + \tau_k \end{aligned} \quad (3)$$

The public verification function consist of the  $n$  quadratic equations  $P_k$ .

### 2.2. Patarin’s Cryptanalysis of $C^*$

The straight-forward attack against IM is to find a map  $[\tilde{G}^*]^{-1} : K^n \rightarrow K^n$ , which satisfies

$$G^*([\tilde{G}^*]^{-1}(Y)) = Y \quad (4)$$

If the attacker knows all of the  $n$  quadratic equations  $P_i$ , that is, full dimensional  $G^*(X)$ , then the problem to find  $[\tilde{G}^*]^{-1}$  is known as the  $C^*$  problem [11]. The straight-forward attack against the  $C^*$  problem is to solve the quadratic system consisting of the  $n$  public equations  $P_i$ . However, Patarin’s cryptanalysis [14] demonstrated that breaking  $C^*$  is not equivalent to solving a quadratic system over  $K^n$ . His attack is classified into chosen message attacks: the attacker gathers enough pairs of messages and signatures by computing the verification function  $G^*(X)$ . From  $B = F(A) = A^{q^\theta+1}$ , we obtain the equation

$$A^{q^{2\theta}} \cdot B = A \cdot B^{q^\theta} \quad (5)$$

by raising both sides of the first equation to the power  $q^\theta - 1$  and multiplying them with  $A \cdot B$ . That is, equation (5) is

linear in both  $A$  and  $B$ . As a result, each pair of message and signature has an equation of the form

$$\sum_{0 \leq j, k < n} \gamma_{j,k,l} x_j y_k + \sum_{0 \leq j < n} (\delta_{j,l} x_j + \epsilon_{j,l} y_j) + \eta_l = 0, \quad (6)$$

where  $\gamma_{j,k,l}$ ,  $\delta_{j,l}$ ,  $\epsilon_{j,l}$ ,  $\eta_l$  are elements of  $K$ , and  $(x_0, \dots, x_{n-1})$ ,  $(y_0, \dots, y_{n-1})$  respectively denote the input and the output of the function  $G^*(X)$ . These equations are linear in both sets of variables  $(x_0, \dots, x_{n-1})$  and  $(y_0, \dots, y_{n-1})$ . An input-output pair provides a linear equation of the coefficients of  $\gamma_{j,k,l}$ ,  $\delta_{j,l}$ ,  $\epsilon_{j,l}$  and  $\eta_l$ . If a sufficient number of input-output pairs are gathered, the attacker can recover at most  $n$  linear independent equations of the form (6) [14, 8]. From these equations, the attacker is able to construct a dummy signature function  $[\tilde{G}^*]^{-1}$ . Patarin's cryptanalysis is particularly efficient: breaking the  $C^*$  problem becomes roughly equivalent to finding the kernel of a  $n^2 \times (n+1)^2$  matrix in  $K$ .

### 2.3. Variants of IM

In [14], Patarin broke IM with an attack involving linear algebra. Then, several variants which were assumed to be resistant against Patarin's cryptanalysis were proposed: among other Little Dragon and Big Dragon [15]. However, they were all broken, except FLASH [2] and its even faster variant SFLASH [12].

## 3. SFLASH

In this section we review SFLASH [12]. We describe the specifications required for our attack and the currently known attacks are shortly surveyed. For detailed description of SFLASH, see [12].

### 3.1. Specification of SFLASH

SFLASH is a variant of Imai-Matsumoto cryptosystem [11], which was the first of hidden monomial cryptosystems. SFLASH particularly uses the following finite fields:  $K = \mathbf{F}_2[X]/(X^7 + X + 1)$  and  $\mathcal{L} = K[X]/(X^{37} + X^{12} + X^{10} + X^2 + 1)$ . In order to embed a message to the finite fields we utilize the following maps:  $\pi : \{0, 1\}^7 \rightarrow K$  and  $\varphi : K^{37} \rightarrow \mathcal{L}$ , where we define  $\pi(b) = b_6 X^6 + \dots + b_1 X + b_0 \pmod{X^7 + X + 1}$  and  $\varphi(\omega) = \omega_{36} X^{36} + \dots + \omega_1 X + \omega_0 \pmod{X^{37} + X^{12} + X^{10} + X^2 + 1}$ , respectively. SFLASH uses a permutation function  $F : \mathcal{L} \rightarrow \mathcal{L}$  defined by  $F(A) = A^{128^{11} + 1}$ . Since  $X \mapsto X$  and  $X \mapsto X^{128^{11}}$  are linear in  $\mathcal{L}$ ,  $F$  is a quadratic map of  $\mathcal{L}$ . In addition to  $K, \mathcal{L}, F$ , the hash function SHA-1 [13] is a system parameter.

SFLASH has three secret parameters: an 80-bit secret signing  $\Delta$ , and two affine secret bijections  $(s, t): K^{37} \rightarrow$

$K^{37}$ . The public key  $G$  of SFLASH is constructed as follows:  $G$  is a map of  $K^{37} \rightarrow K^{26}$ , which is the restriction of the map  $G^*$  onto  $K^{26}$ , where  $G^*$  is the map of  $K^{37} \rightarrow K^{37}$  defined by  $G^*(X) = t \circ \varphi^{-1} \circ F \circ \varphi \circ s(X)$ . Because  $F$  is a quadratic map,  $G^*$  is a quadratic map of  $K^{37}$ , too, and can be written as follows:  $G^*(x_0, \dots, x_{36}) = (y_0, \dots, y_{36})$ , where

$$y_k = \sum_{0 \leq i < j < 37} \rho_{k,i,j} x_i x_j + \sum_{0 \leq i < 37} \sigma_{k,i} x_i + \tau_k \quad (7)$$

That is  $G(X) = [G^*(X)]_{0 \rightarrow 181}$ , where  $[x]_{c \rightarrow d}$  means the  $(d - c + 1)$ -bit string consisting of the bits from the  $c$ -th bit to the  $d$ -th bit of  $x$ . In other words,  $G$  consists of 26 out of the 37 quadratic equations (7).

In the following we describe the signing and verification algorithms of SFLASH; Figure 1 illustrates these algorithms with flow charts.

**Signature Generation:** The signature generation function SFLASH\_Signature computes a signature  $S$  for a given message  $M$  using the secret keys  $(\Delta, s, t)$  as follows:

SFLASH_Signature
Input: A message $M$ , secret parameters $(\Delta, s, t)$
Output: The signature $S$
1. $M_1 \leftarrow \text{SHA-1}(M)$ , $M_2 \leftarrow \text{SHA-1}(M_1)$
2. $V \leftarrow [M_1]_{0 \rightarrow 159}    [M_2]_{0 \rightarrow 21}$
3. $W \leftarrow [\text{SHA-1}(V    \Delta)]_{0 \rightarrow 76}$
4. $Y \leftarrow (\pi([V]_{0 \rightarrow 6}), \pi([V]_{7 \rightarrow 13}), \dots, \pi([V]_{175 \rightarrow 181}))$
5. $R \leftarrow (\pi([W]_{0 \rightarrow 6}), \pi([W]_{7 \rightarrow 13}), \dots, \pi([W]_{70 \rightarrow 76}))$
6. $X \leftarrow s^{-1} \circ \varphi^{-1} \circ F^{-1} \circ \varphi \circ t^{-1}(Y    R)$
7. $S \leftarrow \pi^{-1}(X_0)    \dots    \pi^{-1}(X_{36})$
8. return( $S$ )

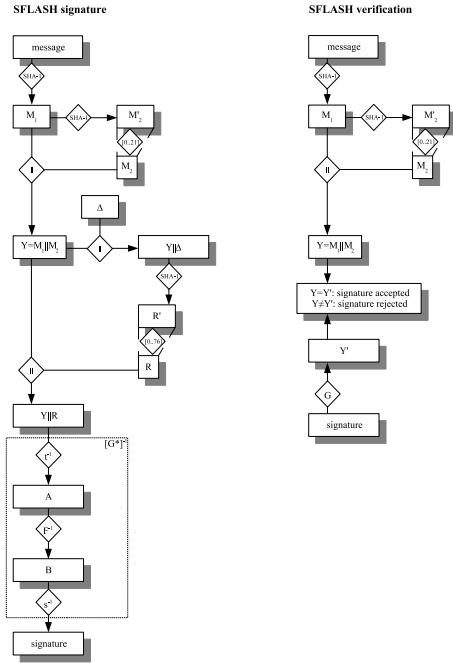
Note that  $x || y$  denotes the concatenation of two strings  $x, y$ . SFLASH\_Signature first computes the hash values  $(Y || R)$  using  $\Delta$  and then the signature  $[G^*]^{-1}(Y || R)$  using  $(s, t)$ , where  $[G^*]^{-1} = s^{-1} \circ \varphi^{-1} \circ F^{-1} \circ \varphi \circ t^{-1}$ . The map  $G^*$  is regarded as a trapdoor one-way function in SFLASH. Note that the first hash value  $Y$  does not depend on the secret key  $\Delta$ , but the second hash value  $R$  does. The 11 elements of  $K$  for input to  $[G^*]^{-1}$  are not public, so that the security is based on the  $C^{*--}$  problem [15]. The  $C^{*--}$  problem will be discussed in the next section.

**Signature Verification:** For a given message  $M$  and its signature  $S$ , we can verify the validity of the signature as follows:

SFLASH_Verification
Input: A message $M$ , a signature $S$ , a public parameter $G$
Output: "valid" or "invalid"
1. $M_1 \leftarrow \text{SHA-1}(M)$ , $M_2 \leftarrow \text{SHA-1}(M_1)$
2. $V \leftarrow [M_1]_{0 \rightarrow 159}    [M_2]_{0 \rightarrow 21}$
3. $Y \leftarrow (\pi([V]_{0 \rightarrow 6}), \pi([V]_{7 \rightarrow 13}), \dots, \pi([V]_{175 \rightarrow 181}))$
4. $X_0 \leftarrow \pi([S]_{0 \rightarrow 6}), \dots, X_{36} \leftarrow \pi([S]_{252 \rightarrow 258})$
5. $Y' \leftarrow G(X_0, \dots, X_{36})$
6. If $Y = Y'$ then return "valid", otherwise "invalid".

The verification function of SFLASH does not require the secret information  $\Delta$ , and only the first 26 elements of  $K$  are compared for checking the validity of a signature.

**Figure 1. SFLASH signature generation and verification**



### 3.2. Known Attacks on SFLASH

SFLASH uses the public key  $G$  instead of  $G^*$ . This means that  $r$  out of the 37 equations  $P_i (i = 0, 1, \dots, 36)$  that belong to  $G^*$  are hidden. Patarin et al. discussed that the problem to find  $[G^*]^{-1}$  for given  $P_i (i = 0, 1, \dots, n-r)$  is hard if  $\#K^r$  is large enough [15]. This modified  $C^*$  problem with larger  $\#K^r$  is called  $C^{*-}$  problem according to [15]. The choice  $r = 11$  of SFLASH is secure due to  $(2^7)^r = 2^{77}$ .

Geiselmann et al. discussed the security of  $(s, t)$ , which is the affine part of the secret key [5]. Denote by  $s = (S_L, S_C)$  and  $t = (T_L, T_C)$  the linear transformation matrix and the constant vector of the affine bijection maps  $s, t : K^{37} \rightarrow K^{37}$ , respectively. They showed that the constant vectors  $S_C, T_C \in K^{37}$  do not really randomize the map  $G$ .

Gilbert and Minier presented an attack on the original version of SFLASH (we refer to it as SFLASHv1) [6]. The secret keys  $(s, t)$  of SFLASHv1 were chosen not in  $K^{37} \rightarrow K^{37}$  but  $\mathbf{F}_{2^{37}} \rightarrow \mathbf{F}_{2^{37}}$ . From this choice of  $(s, t)$ , the hidden quadratic equations  $P_i (i = 26, \dots, 36)$  can be

represented as

$$P_i(x_0, \dots, x_{36}) = \sum_{0 \leq j < k < 37} \alpha_{i,j,k} x_j x_k + \sum_{0 \leq j < 37} \beta_{i,j} x_j, \quad (8)$$

where  $\alpha_{i,j,k}, \beta_{i,j} \in \mathbf{F}_2$ . This representation induces a tiny SFLASH problem which is defined over  $\mathbf{F}_2$  instead of  $K$ . This tiny SFLASH problem is not infeasible anymore. The attacker can recover the hidden polynomials  $P_i (i = 26, \dots, 36)$ . In other words, he/she can transform the  $C^{*-}$  problem into the  $C^*$  problem. Patarin's cryptanalysis is now applicable to SFLASHv1. They showed that the complexity of finding these equations is reduced to less than  $2^{38}$ . The revised version of SFLASH (i.e. SFLASHv2) chooses  $(s, t)$  in the extension field  $K = \mathbf{F}_{2^{27}}$ .

### 3.3. Reducing $C^{*-}$ to $C^*$ using $\Delta$

In the following, we assume that the secret key  $\Delta$  is known. Under this condition, we describe how SFLASH is vulnerable to Patarin's cryptanalysis [14], which enables us to efficiently construct a dummy signature generation function.

Recall that SFLASH\_Signature generates a signature  $S$  for a given message  $M$  using the secret key  $\Delta$  and the publicly known hash function SHA-1. The algebraic part related to the  $C^{*-}$  problem is the function

$$S = s^{-1} \circ \varphi^{-1} \circ F^{-1} \circ \varphi \circ t^{-1}(Y||R) = [G^*]^{-1}(Y||R), \quad (9)$$

where  $Y \in K^{26}$  is depending only on the message  $M$  but  $R \in K^{11}$  is additionally on  $\Delta$ . Note that we know both  $Y$  and  $R$  from secret key  $\Delta$ . Therefore, the attacker is able to obtain full input  $Y||R \in K^{37}$  and output  $S \in K^{37}$ .

Next, a signature is a random element of  $K^{37}$ , because  $[G^*]^{-1}$  is a bijection function. Let  $(x_{0,k}, \dots, x_{36,k})$  be randomly chosen signatures. We can compute the image of these signatures by the map  $G^*$  using the public verification function  $G$  and  $\Delta$ . We denote these images by  $(y_{0,k}, \dots, y_{36,k})$ . Therefore the attacker can have pairs of whole input and output to  $[G^*]^{-1}$ , and the  $C^*$  problem is obtained.

Thus, we have proved the following claim:

**Claim 1** *Under the assumption that the secret key  $\Delta$  is known, the security of SFLASH is reduced from  $C^{*-}$  to  $C^*$ .*

### 4. Side Channel Attacks

In this section we review side channel attacks. Side channel attacks (SCA) allow to access the additional information linked to the operations using the secret key, e.g., timings [9], power consumptions [10], etc. The attack

aims at guessing the secret key (or some related information). If the attacker is allowed to observe side channel information only a few times, it is called simple power analysis (SPA). If the attacker can analyze several side channel information using a statistical tool, it is called differential power analysis (DPA). The standard DPA utilizes the correlation function that can distinguish whether a specific bit is related to the observed calculation. In particular, XOR operation between the secret key and temporary data are often utilized as a target.

Akkar et al. proposed an efficient implementation of SFLASH, which is suitable for smartcards [1]. Implementation of cryptographic algorithms on memory-constrained environments is generally vulnerable to side channel attacks [9, 10]. In order to resist side channel attacks, they show how to randomize the secret key  $(s, t)$  based on the homomorphic property of the function  $F^{-1}$ .

On the other hand, Steinwandt et al. presented a theoretical DPA on the secret key  $\Delta$  by analyzing the operations of the hash function SHA-1 [16]. Klíma-Rosa also proposed another side channel attack against SHA-1 [7] in the RSAES-OAEP [17]. These attacks utilize XOR for detecting the secret. We stress that these attacks are only theoretical works and are not confirmed experimentally. In fact, assumptions of these attacks are controversial. We discuss these side channel attacks against SHA-1 in the following.

#### 4.1. Side Channel Attacks against SHA-1

First we briefly review the hash function SHA-1. For detailed description of SHA-1, see [13]. The algorithm of SHA-1 is as follows:

##### SHA-1

Input: A message  $M$

Output: The hashed value SHA-1( $M$ )

1. (padding)  $M' = M_0 || M_1 || \dots || M_n$
2. (initialization)  $H_0 \leftarrow h_0, H_1 \leftarrow h_1, H_2 \leftarrow h_2, H_3 \leftarrow h_3, H_4 \leftarrow h_4$
3.  $A \leftarrow H_0, B \leftarrow H_1, C \leftarrow H_2, D \leftarrow H_3, E \leftarrow H_4$
4. The 512-bit word  $M_i$  is divided into sixteen 32-bit words  $W_0, \dots, W_{15}$ ;  
 $M_i = W_0 || W_1 || \dots || W_{15}$
5. for  $t = 16$  to  $79$  do the following:
  - 5.1.  $W_t \leftarrow (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) \lll 1$
6. for  $t = 0$  to  $79$  do the following:
  - 6.1.  $TEMP1 \leftarrow (A \lll 5) + f_t(B, C, D) + E \text{ mod } 2^{32}$
  - 6.2.  $TEMP2 \leftarrow TEMP1 + W_t \text{ mod } 2^{32}$
  - 6.3.  $TEMP3 \leftarrow TEMP2 + K_t \text{ mod } 2^{32}$
  - 6.4.  $E \leftarrow D, D \leftarrow C, C \leftarrow B \lll 30, B \leftarrow A, A \leftarrow TEMP3$
7.  $H_0 \leftarrow H_0 + A \text{ mod } 2^{32}, H_1 \leftarrow H_1 + B \text{ mod } 2^{32}, H_2 \leftarrow H_2 + C \text{ mod } 2^{32}, H_3 \leftarrow H_3 + D \text{ mod } 2^{32}, H_4 \leftarrow H_4 + E \text{ mod } 2^{32}$
8. return( $H_0, H_1, H_2, H_3, H_4$ )

SHA-1 uses initial values  $h_0, \dots, h_4$ , constants  $K_0, \dots, K_{79}$ , and logical functions  $f_1, \dots, f_{79}$ , which are defined as follows: (initial values)  $h_0 = (67452301)_H, h_1 = (efcdab89)_H, h_2 =$

$(98badcfe)_H, h_3 = (10325476)_H, h_4 = (c3d2e1f0)_H$ . (Constants)  $K_t = (5a827999)_H$  for  $t = 0, \dots, 19, K_t = (6ed9eba1)_H$  for  $t = 20, \dots, 39, K_t = (8f1bbcdc)_H$  for  $t = 40, \dots, 59, K_t = (ca62c1d6)_H$  for  $t = 60, \dots, 79$ .

The functions  $f_t(x, y, z)$  are the following logical functions

$$\begin{aligned} f_t(x, y, z) &= (x \wedge y) \vee (\bar{x} \wedge z) & (t = 0, \dots, 19) \\ f_t(x, y, z) &= x \oplus y \oplus z & (t = 20, \dots, 39) \\ f_t(x, y, z) &= (x \wedge y) \vee (x \wedge z) \vee (y \wedge z) & (t = 40, \dots, 59) \\ f_t(x, y, z) &= x \oplus y \oplus z & (t = 60, \dots, 79). \end{aligned}$$

The message  $M$  is padded at Step 1. First,  $M$  is divided into 512-bit words  $M_0, \dots, M_{n-1}$ , and the remainder  $M_n$ . The remainder  $M_n$  is appended to  $10\dots 0$  and the bit-length of the message  $M$  in order to be a 512-bit word. The padded message  $M'$  is  $M' = M_0 || M_1 || \dots || M_n$ . Step 4 to 7 is performed for each 512-bit word  $M_i$ .

**Side Channel Attacks on  $\Delta$ :** At Step 3 of SFLASH\_Signature, the message  $M = (V || \Delta)$  is inputted to SHA-1. Since the bit-length of the message is 262, the padded message  $M'$  consists of one 512-bit word  $M_0$ , namely  $M' = M_0$ . At Step 4 of SHA-1, the 512-bit word  $M_0$  is divided into sixteen 32-bit words  $W_0, \dots, W_{15}$ . Therefore, the secret key  $\Delta$  corresponds to  $[W_5]_{22-31} || W_6 || W_7 || [W_8]_{0-5}$ . The attacker's task is to determine above  $W_t$  using side channel information. Note that the other words  $W_0, \dots, W_4, [W_5]_{0-21}, [W_8]_{6-31}, W_9, \dots, W_{15}$  are known to the attacker, since the value of  $V$  and the bit-length of the message  $(V || \Delta)$  are known. However he/she cannot control these values, since  $V$  is an output of SHA-1, and to compute the inverse value of SHA-1 is infeasible.

At Steps 5.1 and 6.2, SHA-1 uses the target  $W_t$ . Thus we consider side channel attacks against the following operations in order to detect  $W_t$ :

- (a)  $W_t \leftarrow (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) \lll 1$  at Step 5.1
- (b)  $TEMP2 \leftarrow TEMP1 + W_t \text{ mod } 2^{32}$  at Step 6.2

**Known Attacks against (a):** In respect of case (a), two attacks are known; the attack of Steinwandt et al. [16] and that of Klíma-Rosa [7]. These attacks utilize XOR operation for revealing the secret. Steinwandt et al. [16] showed that a DPA is applicable to this operation theoretically, and concluded that the secret key  $\Delta$  is completely revealed. The attack assumes that the result of XOR among three words and the target bit is randomly distributed. This seems to be true, however, in this situation many words are equal to  $0x00000000$  and XOR is performed not simultaneously

but one after another in the operation of (a). This might become a cause of failure of the attack. Note that this might reduce the noise of the power consumption, however, we need to confirm the effectiveness of the attack through the experiment, and Steinwandt et al. [16] did not experiment.

On the other hand, Klíma-Rosa [7] proposed another side channel attack against SHA-1 portion in the RSAES-OAEP scheme [17]. The attack is straight-forwardly applicable to SHA-1 in SFLASH. The attacker utilizes Hamming weight of  $W_t$  for detecting unknown word  $W_t$ . This attack is also discussed theoretically. The attack assumes that the attacker can detect the Hamming weight of intermediate data using a single observation. However, this assumption is not realistic, since the power consumption is noisy in general, and the difference of power consumptions with different Hamming weight is below the signal-to-noise (S/N) ratio. In order to distinguish the difference, we often use the averaging technique, which reduces the noise and improves the S/N ratio. However, Klíma-Rosa [7] did not discuss the averaging technique. Even if the averaging technique is applied to their attack, the same data is necessary to input into SHA-1. This is a particular drawback of this attack.

Not to mention, these are possible ways to detect the secret key  $\Delta$ , but we stress that these methods were discussed theoretically and were not confirmed experimentally. In the next section, we propose a side channel attack against the operation (b), which is both theoretical and experimental works.

## 5. Proposed Attacks

In this section we propose a new attack against SFLASH. The proposed attack tries to construct a signature generation function by the help of side channel information.

### 5.1. Main Idea and Attack Model

We describe the main idea of the proposed attacks.

The proposed attacks consist of a side channel attack on the secret parameter  $\Delta$  and an algebraic attack on the signature function `SFLASH_Signature`. The goal of the proposed attack is to generate a new signature function which produces a valid signature for any given message. The proposed attack is a chosen message attack. At first the attacker gathers a lot of pairs  $(M_i, S_i)$  along with the power consumption  $P_i$  (side channel information) of `SFLASH_Signature`. The attacker then proceeds as follows:

**Side Channel Attack on  $\Delta$ :** He/She tries to recover the secret key  $\Delta$  by analyzing the power consumption  $P_i$ . We examine the related operation to  $\Delta$  for SHA-1, and point out the possible operations that are vulnerable to SCA. In

section 5.2, we will discuss the possibility of detecting  $\Delta$  using a side channel attack.

**Algebraic Attack on Signature Function:** In section 3.3 we discussed that the knowledge of  $\Delta$  reduces the security of SFLASH to the  $C^*$  problem for finding the signature function  $[\tilde{G}^*]^{-1}$ . Thus, Patarin's attack is able to construct a dummy signature function. We discuss the complete implementation of Patarin's attack and estimate the required number of signatures in section 5.3.

### 5.2. Side Channel Attacks on $\Delta$

**Proposed Attack against (b):** We consider the case (b). Unfortunately, no known attacks against arithmetic operation exist at present. Note that countermeasures using arithmetic mask were proposed. On one hand, logical operation such as XOR is computed bit-wise, and the target bit depends only on some specific bit in it. Thus, to construct an attack against logical operation is simple. On the other hand, an arithmetic operation is calculated using carry/borrow, and the target bit depends on several bits in it. Thus, to construct an attack against arithmetic operation is complicated.

We propose an attack against arithmetic addition of (b). Assume that the attacker knows  $W_0, W_1, \dots, W_{t-1}$ ,  $[W_t]_{j+1 \rightarrow 31}$ , and his/her target is now the  $j$ -th bit of  $[W_t]$ . At Step 6.1 of the  $t$ -th iteration, TEMP1 is computed. Under the assumption, TEMP1 is computable for the attacker, since TEMP1 is computed using known values only. At Step 6.2,  $\text{TEMP1} + W_t \bmod 2^{32}$  is computed, and the result is stored at TEMP2.

The  $j$ -th bit of TEMP2 depends on  $[\text{TEMP1}]_{j \rightarrow 31}$  and  $[W_t]_{j \rightarrow 31}$ , and is independent from  $[\text{TEMP1}]_{0 \rightarrow j-1}$  and  $[W_t]_{0 \rightarrow j-1}$ . The attacker temporarily put the  $j$ -th bit of  $[W_t]$  to be 0, and computes the  $j$ -th bit of TEMP2 for several inputs  $(V||\Delta)$ . Note that he/she can compute the  $j$ -th bit of TEMP2, since he/she knows  $[\text{TEMP1}]_{j \rightarrow 31}$  and  $[W_t]_{j+1 \rightarrow 31}$ , but he/she does not know whether his/her guess is correct or not. Thus, he/she needs to confirm whether the guess is correct or not, using side channel information.

He/she classifies the inputs  $(V||\Delta)$  into two categories; the  $j$ -th bit of TEMP2 is 0 or 1. Then he/she observes the power consumption for each input, and calculates the average power consumption for each category. The difference of these averages provides the information whether attacker's guess is correct or not. If the guess is correct, the difference involves clear spikes. On the contrary, if the guess is incorrect, the difference tends to vanish or involves small spikes. Note that small spikes appear if the attacker correctly guesses some lower bits of  $W_t$ . This is because the lower bits of TEMP2 are determined by those of  $W_t$  and

TEMP1, since  $\text{TEMP2} = \text{TEMP1} + W_t \bmod 2^l$  holds for  $l = 1, \dots, 32$ . Thus the attacker's task is to distinguish the size of spikes.

Once the  $j$ -th bit of  $W_t$  has been revealed, the attacker tries to reveal the next bit in the same way. He/she repeats this process, and reveals the secret key  $\Delta$ . Note that he/she does not need to re-observe the power consumption, once he/she has gathered a sufficient number of power consumption samples for detecting the bit of  $W_t$ , because the samples can be re-used. Thus his/her task for the next bit is to classify the samples into two categories based on the bit, and to calculate the average of each category.

We should note that the attacker may attempt to reveal several bits of  $W_t$  simultaneously, instead of single bit. Concretely speaking, when he/she attempts to reveal consecutive  $a$  bits of  $W_t$  simultaneously, he/she classifies the inputs into two categories according to the Hamming weight of TEMP2 for each candidate of  $W_t$ . Then for each candidate, he/she computes the difference of average power consumption for two categories. After that, he/she selects the sample with the biggest peak. The sample indicates the secret  $a$  bits of  $W_t$ . Note that the sample with the second biggest peak corresponds to the candidate of  $W_t$  which is close to the correct  $W_t$  under the Hamming distance. The use of such samples convinces the attacker of the correctness of his/her guess.

### 5.3. Implementation of Patarin's Attack

In the following, we describe an implementation of Patarin's algebraic attack [14], and we estimate the number of signature/message pairs required for it. Let  $(x_{0,k}, \dots, x_{36,k})$  be randomly chosen signatures, and denote by  $(y_{0,k}, \dots, y_{36,k})$  their respective images, computed with the public equations (7). These pairs are the whole input and output to  $[G^*]^{-1}$ , and Patarin's cryptanalysis is applicable.

Patarin's attack consists of two parts, namely the constructions of the equations (6) and an exhaustive search. The equations (6) are computed only once for a given SFLASH instance, and they give us candidates of forged signatures for any given message. Then an exhaustive search determines the proper signature, which corresponds to the inputted message.

**Generation of the equations (6):** At first the attacker tries to find the coefficients  $\gamma_{j,k,l}, \delta_{j,l}, \epsilon_{j,l}, \eta_l$  of the equations (6). These coefficients can be found by solving a linear system. There are at most 37 independent equations (6) [8]. In order to generate these 37 equations, the attacker deals with a  $1407 \times 1444$  matrix in  $K$ , which consists of four submatrices constructed with the gathered input-output pairs  $(x_{0,k}, \dots, x_{36,k}) - (y_{0,k}, \dots, y_{36,k})$ :

- The first  $1407 \times 1407$  submatrix is square; its coefficients are the cross-products  $x_{i,k}y_{j,k}$ .
- The dimensions of the second and the third submatrices are  $1407 \times 37$ ; their lines consist of the individual coefficients  $x_{i,k}$  for the second and  $y_{i,k}$  for the third submatrix.
- The last submatrix is the 1407-column-vector  $(1, \dots, 1)$ .

The lines of the matrix have to be independent, so that its rank is exactly 1407 and the dimension of its kernel 37: in this case, the coefficients  $\gamma_{j,k,l}, \delta_{j,l}, \epsilon_{j,l}$  and  $\eta_l$  of the 37 equations (6) are vectors of the kernel of this matrix.

We explain how to efficiently construct the matrix and its kernel in the following. We denote by  $line(k)$  the line constructed with the  $k$ -th input-output pair  $(x_{0,k}, \dots, x_{36,k}) - (y_{0,k}, \dots, y_{36,k})$ : first the 1407 cross-products  $x_{i,k}y_{j,k}$ , then the 37  $x_{i,k}$ , the 37  $y_{j,k}$  and finally 1. The matrix is recursively constructed by augmenting it with new lines which are not a linear combination of the other lines of the matrix. This can be achieved with the following algorithm:

#### Generation of the matrix

Input:  $(x_{0,k}, \dots, x_{36,k}), (y_{0,k}, \dots, y_{36,k})$

Output: An upper matrix  $M$

1.  $M \leftarrow line(0)$
2. while  $M$  do not have 1407 lines do the following:
  - 2.1.  $M' \leftarrow M$
  - 2.2. augment  $M'$  with  $line(k), k \leftarrow k + 1$
  - 2.3. make  $M'$  upper (Gauss reduction on the last line)
  - 2.4. if there is a non-zero field on the last line then
    - 2.4.1. if the diagonal of the last line is zero then swap columns so that the diagonal becomes non-zero
    - 2.4.2.  $M \leftarrow M'$
3. return( $M$ )

At the end of the procedure, the matrix has 1444 columns and 1407 lines, the coefficients are non-zeros on the diagonal and only zeros are to be found under the diagonal.

Next, we describe the computation of the kernel of the matrix. Given the matrix described above, the simplest method to compute its kernel is to carry on the gauss reduction and eliminate the unknowns on the right of the diagonal to obtain such a matrix:

$$\left\{ \begin{array}{cccc|ccc} 1 & 0 & \cdots & 0 & u_{0,0} & \cdots & u_{0,36} \\ 0 & \ddots & \ddots & \vdots & \vdots & & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots & & \vdots \\ 0 & \cdots & 0 & 1 & u_{1406,0} & \cdots & u_{1406,36} \end{array} \right\}$$

Then a basis of the kernel can be computed with the following 37 vectors:

$$b_i = \left( -u_{0,i}, \dots, -u_{1406,i}, \underbrace{0, \dots, 0}_{i \text{ times}}, 1, 0, \dots, 0 \right)$$

where  $i = 0, \dots, 36$ .

The columns of the matrix were possibly swapped as described in the matrix generation procedure, thus swapping the fields of these vectors in the inverse order provides a basis of the kernel. Finally, the fields of these vectors are the coefficients  $\gamma_{j,k,l}, \delta_{j,l}, \epsilon_{j,l}, \eta_l$  of the 37 equations (6).

**Exhaustive search:** We explain how to generate a signature for a given message  $(\bar{y}_0, \dots, \bar{y}_{36})$  using an exhaustive search. If the attacker substitutes the message to the equations (6), he/she gets a linear system of 37 equations and 37 unknowns  $(x_0, \dots, x_{36})$ .

$$\sum_{0 \leq j < 37} \left( \sum_{0 \leq k < 37} \gamma_{j,k,l} \bar{y}_k + \delta_{j,l} \right) x_j + \sum_{0 \leq k < 37} \epsilon_{k,l} \bar{y}_k + \eta_l = 0 \quad (10)$$

where  $l = 0, \dots, 36$ .

This system is equivalent to the equation  $A^{128^{22}} \cdot \bar{B} = A \cdot \bar{B}^{128^{11}}$  from section 3.2, when  $\bar{B}$  is fixed. The solutions of this equation are:

- $A = 0$
- $A_0 = B^{(128^{11}+1)^{-1}}$ , which corresponds to the valid signature.
- if  $A$  is another solution, then  $A$  and  $A_0$  differ from a factor which is a  $(128^{11} - 1)$ -root of unity in  $\mathcal{L}$ . Reciprocally, if  $A_0$  is multiplied with a  $(128^{11} - 1)$ -root of unity, we get another solution of the equation [8].

There are 128  $(128^{11} - 1)$ -roots of unity in  $\mathcal{L}$ . Therefore, the system (10) has also 128 solutions, which build an affine space of dimension 1, and the attacker has to search for a valid signature among them. Conceptually, finding these solutions is similar to finding the equations (6), except that the dimension of the system is 37 instead of 1444: one has to find a particular solution  $X_0$  of the system and a basis  $X_1$  of the kernel of the matrix associated to the system. The solutions of the system are  $X = X_0 + \lambda X_1$ , where  $\lambda \in K$  and  $|K| = 128$ . This means that there are only 128 candidates. The valid signature besides verifies  $G^*(X) = \bar{Y}$ , where  $\bar{Y} = (\bar{y}_0, \dots, \bar{y}_{36})$ .

**Experimental Results:** We have implemented the algebraic attack described above. Our practical simulation confirms that 1410 message-signature pairs already allow to build the equations (6) with a high probability, and 1408 pairs are needed on average. These results show that once  $\Delta$  is known, the recommended parameters of SFLASH make this cryptosystem particularly weak to Patarin's attack. Our implementation proves that the threat is not only theoretical but also practical, since the running time of Patarin's attack makes it feasible for an attacker to forge a signature in a relatively short time.

## 5.4. Dependence on the Signing Oracle

On the one hand, once an attacker knows both of  $\Delta$  and the public parameters of a SFLASH instance, the algebraic attack is able to forge a signature without any oracle calls, as we explained in section 5.3. Therefore, the attacker is able to construct the dummy signature function totally offline — he/she does not have to access the signing oracle. On the other hand, the side channel attack in section 5.2 needs to observe the operation using the secret key  $\Delta$  — SCA has to access the signing oracle. The success of SCA strongly depends on the architecture of the computational environment of the devices.

## 5.5. Experimental Results for Detecting $\Delta$

In this section we describe the experimental result for the proposed side channel attack for detecting  $\Delta$ .

We implemented SFLASH\_Signature with SHA-1 on an IC chip, and attempted to reveal  $\Delta$  using the proposed side channel attack. We observed 200,000 samples<sup>2</sup> of power consumption while the device performed SFLASH\_Signature. The experiments assumed the followings: The attacker aims to reveal the secret word  $W_6$  that corresponds to a portion of  $\Delta$ , and has already detected the lower sixteen bits of  $W_6$  using the proposed attack. His/Her target is now the next four bits of  $W_6$ .

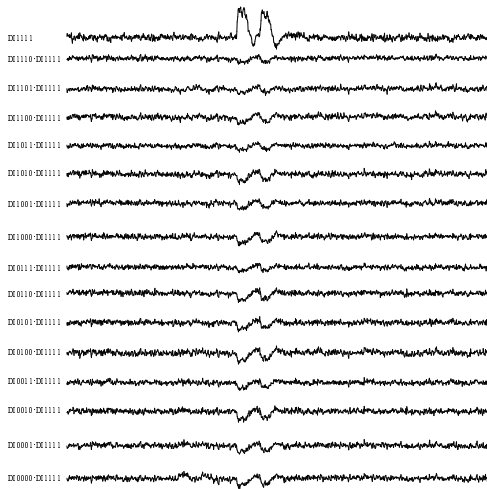
First the attacker assumes that the target four bits are 0000, and classifies the inputs. That is, if an input has a Hamming weight of lower 20(= 16 + 4) bits of TEMP2 which is larger than or equal to 13, it belongs to the class  $\mathcal{C}_{13}$ . If an input has a Hamming weight of lower 20 bits of TEMP2 which is smaller than or equal to 7, it belongs to the class  $\mathcal{C}_7$ .<sup>3</sup> Then the attacker respectively computes the average power consumptions  $PC_{13}$  and  $PC_7$  of  $\mathcal{C}_{13}$  and  $\mathcal{C}_7$  using the observed 200,000 samples. Then he/she computes the difference  $DI_{0000} = PC_{13} - PC_7$ . In the same way, the attacker computes  $DI_{0001}, DI_{0010}, \dots, DI_{1111}$ . In fact, every  $DI$  shows peaks, and  $DI_{1111}$  has the largest peak. The uppermost curve of Fig. 2 shows  $DI_{1111}$ , in which two peaks are clearly visible.

In order to show that  $DI_{1111}$  has the largest peak, we computed the differences  $DI_{0000} - DI_{1111}, DI_{0001} - DI_{1111}, \dots$ , and  $DI_{1110} - DI_{1111}$ . Fig. 2 shows these difference curves. The lowest curve indicates  $DI_{0000} - DI_{1111}$ , the second lowest curve indicates  $DI_{0001} - DI_{1111}$ , and so on. Each curve has depressions, which shows that

<sup>2</sup>In order to show the clear difference in Fig.2, we observed the 200,000 samples. The use of fewer samples is enough to distinguish the difference. However, this fact may show both the effectiveness and ineffectiveness of the proposed attack in some practical situation.

<sup>3</sup>Of course,  $(\mathcal{C}_{14}, \mathcal{C}_6)$  for example is also acceptable. In our experiment, the use of  $(\mathcal{C}_{13}, \mathcal{C}_7)$  provided the clearest result. This is the reason why we use these classes in this paper.

**Figure 2. The experimental result on the proposed side channel attack**



$DI_{1111}$  has the largest peak. In addition,  $DI_{1110} - DI_{1111}$ ,  $DI_{1101} - DI_{1111}$ ,  $DI_{1011} - DI_{1111}$ , and  $DI_{0111} - DI_{1111}$  have the shallowest depressions. This implies that 1110, 1101, 1011, and 0111 are almost correct, that is, the difference from the correct four bits is just one bit. Thus, the four bits of  $W_6$  are certainly 1111. In fact, this is the case.

The remaining bits of  $W_6$  were detected in the same way. The attacker can detect other secret words  $W_t$  using the observed samples. Therefore, the proposed side channel attack against  $\Delta$  certainly works.

## 6. Conclusion

We showed a total experimental result about a side channel attack on SFLASH. The critical point of breaking SFLASH is the treatment of the secret key  $\Delta$ . We demonstrated that if the implementation of SHA-1 is naive or careless, the attacker can recover  $\Delta$ . The knowledge of  $\Delta$  reduces the security of the SFLASH to the  $C^*$  problem, which was broken by Patarin. From our simulation, about 1408 pairs of messages and their signatures can generate a dummy signature function in a few hours. Patarin's attack can be totally performed off-line, without asking the signing oracle. In order to resist the proposed attack, the developers of SFLASH should carefully implement SHA-1 related to  $\Delta$ .

## References

- [1] M. Akkar, N. Courtois, L. Goubin, and R. Duteuil. A fast and secure implementation of SFLASH. In *PKC 2003*, volume 2567 of *LNCS*, pages 267–278, 2003.
- [2] N. Courtois, L. Goubin, and J. Patarin. FLASH, a fast multivariate signature algorithm. In *CT-RSA 2001*, volume 2020 of *LNCS*, pages 298–307, 2001.
- [3] N. Courtois, L. Goubin, and J. Patarin. SFLASHv3, a fast asymmetric signature scheme. Cryptology ePrint Archive, Report 2003/211, 2003. <http://eprint.iacr.org/>.
- [4] European Commission. *NESSIE, New European Schemes for Signatures, Integrity, and Encryption*. <https://www.cosic.esat.kuleuven.ac.be/nessie/>.
- [5] W. Geiselmann, R. Steinwandt, and T. Beth. Attacking the affine parts of SFLASH. In *Cryptography and Coding, 8th IMA International Conference*, volume 2260 of *LNCS*, pages 355–359, 2001.
- [6] H. Gilbert and M. Minier. Cryptanalysis of SFLASH. In *Eurocrypt 2002*, volume 2332 of *LNCS*, pages 288–298, 2002.
- [7] V. Kilima and T. Rosa. Further results and considerations on side channel attacks on RSA. In *CHES 2002*, volume 2523 of *LNCS*, pages 244–259, 2003.
- [8] N. Koblitz. *Algebraic Aspects of Cryptography*, volume 3 of *Algorithms and Computation in Mathematics*. Springer-Verlag, 1998.
- [9] C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *CRYPTO '96*, volume 1109 of *LNCS*, pages 104–113, 1996.
- [10] C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *CRYPTO '99*, volume 1666 of *LNCS*, pages 388–397, 1999.
- [11] T. Matsumoto and H. Imai. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In *EUROCRYPT '88*, volume 330 of *LNCS*, pages 419–453, 1988.
- [12] MinRank Foundation. *The SFLASH signature scheme*. <http://www.minrank.org/sflash/>.
- [13] NIST. *Secure Hash Standard, Federal Information Processing Standards Publication 180-1*. <http://csrc.nist.gov/>.
- [14] J. Patarin. Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt '88. In *CRYPTO '95*, volume 963 of *LNCS*, pages 248–261, 1995.
- [15] J. Patarin, L. Goubin, and N. Courtois.  $C^{*+}$  and HM: Variations around two schemes of T. Matsumoto and H. Imai. In *ASIACRYPT '98*, volume 1514 of *LNCS*, pages 35–49, 1998.
- [16] T. B. R. Steinwandt, W. Geiselmann. A theoretical DPA-based cryptanalysis of the NESSIE candidates FLASH and SFLASH. In *ISC 2001*, volume 2200 of *LNCS*, pages 280–293, 2001.
- [17] RSA Laboratories. *Public-Key Cryptography Standards, PKCS # 1*. <http://www.rsasecurity.com/rsalabs/pkcs/>.