SFLASH^{v3}, a fast asymmetric signature scheme

Specification of SFLASH, version 3.0., 17 October 2003
The authors still recommend SFLASH-v2, see below.

Nicolas T. Courtois[1], Louis Goubin[1] and Jacques Patarin[2]

[1]Axalto Cryptographic Research & Advanced Security,
36-38 rue de la Princesse, BP 45, 78430 Louveciennes Cedex, France
`courtois@minrank.org, LGoubin@slb.com`

[2] PRISM, University of Versailles, France
`Jacques.Patarin@prism.uvsq.fr`

Note: SFLASH^{v2} is one of the three asymmetric signature schemes recommended by the Nessie European consortium for low-cost smart cards [21, 16]. The latest implementation report shows that SFLASH^{v2} is the fastest signature scheme known, see [1] for details.

This document is a specification of SFLASH-v3 produced for fear of SFLASH-v2 being broken (see [3]). HOWEVER after detailed analysis by Chen, Courtois and Yang, see [2], **SFLASH-v2 is not broken** and we still recommend the old version Sflash-v2, also recommended by Nessie, instead of this version.

In this document SFLASH^{v3} is fully specified. In addition in the appendix we summarize all the changes in SFLASH, for readers and developers that are acquainted with the previous version(s).

1 Introduction

In the present document, we describe the updated version SFLASH^{v3} of the SFLASH public key signature scheme proposed in [20] and already revised in [21].

SFLASH belongs to the family of "multivariate" public key schemes, *i.e.* each signature and each hash of the messages to sign are represented by some elements of a small finite field K. The SFLASH signature scheme is based on a C^{*--} trapdoor function algorithm introduced in [25], with a special choice of the parameters. Known attacks on SFLASH, and important results that have a direct impact on its security, can be found in [25, 9, 8, 13, 24, 3, 2, 22, 19, 27, 26, 11, 12, 6].

SFLASH has been designed for very specific applications where the cost of classical cryptographic algorithms (RSA, Elliptic Curves, DSA, etc) becomes prohibitive: they are too slow or/and the signature size if too big. Thus SFLASH have been created to satisfy an extreme property that no standardized public key scheme have reached so far: efficiency on low-price smart cards.

SFLASH is a very fast signature scheme, **both** for signature generation and signature verification. It is much faster than RSA and much easier to implement on smart cards without any arithmetic coprocessor. See [1] for implementation report and some concrete speed results for SFLASH^{v2} compared to its competitors. These results extend easily to SFLASH^{v3}.

The price to pay for speed, and the main drawback of SFLASH, is the public key size being noticeably larger than for RSA. In SFLASH^{v1} and SFLASH^{v2} it was respectively 2.2 and 15.4 Kbytes, see [21, 20, 16] which could easily fit in low-end smart cards. For the new version SFLASH^{v3} it is less obvious: the size of the public key becomes 112 Kbytes. This is why the authors still do recommend SFLASH-v2. SFLASH^{v3} remains competitive and it should remain one of the fastest signature schemes known, that can only be rivalled by NTRU. Though SFLASH public keys are longer than NTRU, the signatures are shorter. Therefore both have their advantages.

It seems that SFLASH approaches its maturity. Starting from the previous version SFLASH^{v2}, there is no method known to distinguish the public key of SFLASH from a random system of quadratic equations over $GF(2^k)$ (see Section 7.2. of [10]). Solving such a system, is a famous hard problem MQ, that also underlies the security of other multivariate encryption and signature schemes. The hardness of this problem is also required for the security of many block ciphers, see [15, 5, 4], including AES, and for several stream ciphers, see [6, 7].

SFLASH was designed to have a security level of 2^{80} with the present state of the art in cryptanalysis, as required in the NESSIE project. A security margin is kept with respect to this goal: the best currently known attack on SFLASH^{v3} requires at least 2^{100} operations, see [3] and in fact much more, as the estimations of [3] turn out to be over-optimistic, see [2].

2 Notation

In all the present document, $||$ will denote the "concatenation" operation. More precisely, if $\lambda = (\lambda_0, \ldots, \lambda_m)$ and $\mu = (\mu_0, \ldots, \mu_n)$ are two strings of elements (in a given field), then $\lambda||\mu$ denotes the string of elements (in the given field) defined by:

$$\lambda||\mu = (\lambda_0, \ldots, \lambda_m, \mu_0, \ldots, \mu_n).$$

For a given string $\lambda = (\lambda_0, \ldots, \lambda_m)$ of bits and two integers r, s, such that $0 \le r \le s \le m$, we denote by $[\lambda]_{r \to s}$ the string of bits defined by:

$$[\lambda]_{r \to s} = (\lambda_r, \lambda_{r+1}, \ldots, \lambda_{s-1}, \lambda_s).$$

3 Parameters of the Algorithm

The SFLASH algorithm uses three finite fields.

- The first one, $K = \mathbf{F}_{128}$ is precisely defined as $K = \mathbf{F}_2[X]/(X^7 + X + 1)$. We will denote by π the bijection between $\{0,1\}^7$ and K defined by:

$$\forall b = (b_0, \ldots, b_6) \in \{0,1\}^7,\ \pi(b) = b_6X^6 + \ldots + b_1X + b_0 \pmod{X^7 + X + 1}.$$

- The second one is $\mathcal{L} = K[X]/(X^{67} + X^5 + X^2 + X + 1)$. We will denote by φ the bijection between K^{67} and $\mathcal{L}$ defined by:

  $\forall \omega = (\omega_0, \ldots, \omega_{66}) \in K^{67}$,

$$\varphi(\omega) = \omega_{66}X^{66} + \ldots + \omega_1X + \omega_0 \pmod{X^{67} + X^5 + X^2 + X + 1}.$$

The SFLASH algorithm uses two affine bijections s and t from K^{67} to K^{67}. Each of them is composed of a secret linear part S_L resp. T_L and of a constant part S_C resp. T_C.

3.1 Secret Parameters

1. A linear secret bijection from K^{67} to K^{67} that is represented by a 67×67 square matrix with entries in K, written with respect to the canonical basis of K^{67}. We denote by S_L this matrix ("L" means "linear").

2. Another linear secret bijection from K^{67} to K^{67} represented by a 67×67 square matrix over K denoted by T_L.

3. A 80-bit secret string denoted by Δ.

3.2 Semi-Public Parameters

In addition, constant parts of s and t are specified:

1. A vector in K^{67} represented by a 67×1 column matrix S_C ("C" means "constant") with respect to the canonical basis of K^{67}.

2. Another vector in K^{67} represented by a column matrix T_C.

Explanation: It is illusory to make these constant parts of s and t secret, see Section 7.1 and [23]. They can be made public does not impact the security of SFLASH. However, we recommend not to publish them. First, because they are not used in signature verification process, and also in order to save space and transmission time for the public key. As a consequence, these values are neither secret (as they can be recovered, see [23]) nor public (as they are not made public). For this reason we call them semi-public.

3.3 Public Parameters

The public key consists in the function G from K^{67} to K^{56} defined by:

$$G(X) = \Big[t\Big(\varphi^{-1}\big(F(\varphi(s(X)))\big)\Big)\Big]_{0\to 391}.$$

Here the subscript ${}_{0\to 391}$ allows to pick 56 equations out of 67 (and $56 \cdot 7 = 392$). F is the function from $\mathcal{L}$ to $\mathcal{L}$ defined by:

$$\forall A \in \mathcal{L},\ F(A) = A^{128^{33}+1}.$$

By construction of the algorithm, G is a quadratic transformation over K, *i.e.* $(Y_0, \ldots, Y_{55}) = G(X_0, \ldots, X_{66})$ can be written, equivalently:

$$\begin{cases} Y_0 = P_0(X_0, \ldots, X_{66}) \\ \quad\vdots \\ Y_{55} = P_{55}(X_0, \ldots, X_{66}) \end{cases}$$

with each P_i being a quadratic polynomial of the form

$$P_i(X_0, \ldots, X_{66}) = \sum_{0\le j<k<67} \zeta_{i,j,k} X_j X_k + \sum_{0\le j<67} \nu_{i,j} X_j + \rho_i,$$

all the elements $\zeta_{i,j,k}$, $\nu_{i,j}$ and ρ_i being in K.

Efficient generation of keys for such multivariate schemes goes beyond the topic of this paper, see [29].

4 Key Generation

In the SFLASH scheme, the public is deduced from the secret key, as explained in section 3.2. We need only to describe how the secret key is generated. As described in section 3.1, the following secret elements have to be generated:

- The secret invertible 67×67 matrix S_L, and the secret 67×1 (column) matrix S_C, all the coefficients being in K.
- The secret invertible 67×67 matrix T_L, and the secret 67×1 (column) matrix T_C, all the coefficients being in K.
- The 80-bit secret string Δ.

4.1 Detailed Step-by-Step Key Generation

Note that, through the π transformation, generating an element of K is equivalent to generating a 7-bit string. In what follows, we call

`next_7bit_random_string`

the string of 7 bits obtained by calling 7 times the CSPRBG (we obtain first the first bit of the string, then the second bit, ..., until the seventh bit).
To generate all these parameters, we apply the following method, which uses a cryptographically secure pseudorandom bit generator (CSPRBG). From a seed whose entropy is at least 80 bits, this CSPRBG is supposed to produce a new random bit each time it is asked to.

1. To generate the invertible 67×67 matrix S_L, two methods can be used:

   First Method ("Trial and error"): Generate the matrix S_L by repeating

   ```
   for i=0 to 66
       for j=0 to 66
           S_L[i,j]=pi(next_7bit_random_string)
   ```

   until we obtain an invertible matrix.

   Second Method (with the LU decomposition): Generate a lower triangular 67×67 matrix L_S and an upper triangular 67×67 matrix U_S, all the coefficients being in K, as follows:

   ```
   for i=0 to 66
       for j=0 to 66
       {
           if (i<j) then
                     {U_S[i,j]=pi(next_7bit_random_string); L_S[i,j]=0;};
           if (i>j) then
                     {L_S[i,j]=pi(next_7bit_random_string); U_S[i,j]=0;};
           if (i=j) then
                    {repeat (z=next_7bit_random_string)
                            until z!=(0,0,0,0,0,0,0);
                     U_S[i,j]=pi(z);
                     L_S[i,j]=1;};
       };
   ```

Define then $S_L = L_S \times U_S$.

2. Generate S_C by using the CSPRBG to obtain 67 new random elements of K (from the top to the bottom of the column matrix). Each of these elements of K is obtained by

```
pi(next_7bit_random_string)
```

3. The generation of the invertible 67×67 matrix T_L, is done exactly as for S_L.

4. The generation of T_C is done exactly as for S_C by using the CSPRBG to obtain 67 new random random elements of K.

5. Finally, generate Δ by using the CSPRBG to obtain 80 random bits.

5 Signing a Message

In the present section, we describe the signature of a message M by the SFLASH algorithm.

5.1 The Signing Algorithm

The message M is given by a string of bits. Its signature S is obtained by applying successively the following operations (see figure 1):

1. Let M_0, M_1, M_2 and M_3 be the three 160-bit strings defined by:

$$M_0 = \text{SHA-1}(M),$$

$$M_1 = \text{SHA-1}(M_0||0x00),$$
$$M_2 = \text{SHA-1}(M_0||0x01).$$
$$M_3 = \text{SHA-1}(M_0||0x02).$$

With 0x00 through 0x02 denoting one single 8-bit character appended to M_0.

2. Let V be the 392-bit string defined by:

$$V = [M_1]_{0\to 159}||[M_2]_{0\to 159}||[M_3]_{0\to 71}.$$

3. Let W be the 77-bit string defined by:

$$W = [\text{SHA-1}(V||\Delta)]_{0\to 76}.$$

4. Let Y be the string of 56 elements of K defined by:

$$Y = \Big(\pi([V]_{0\to 6}), \pi([V]_{7\to 13}), \ldots, \pi([V]_{385\to 391})\Big).$$

5. Let R be the string of 11 elements of K defined by:

$$R = \Big(\pi([W]_{0\to 6}), \pi([W]_{7\to 13}), \ldots, \pi([W]_{70\to 76})\Big).$$

6. Let B be the element of $\mathcal{L}$ defined by:

$$B = \varphi\Big(t^{-1}(Y||R)\Big).$$

7. Let A be the element of $\mathcal{L}$ defined by:

$$A = F^{-1}(B),$$

F being the function from $\mathcal{L}$ to $\mathcal{L}$ defined by:

$$\forall A \in \mathcal{L},\ F(A) = A^{128^{33}+1}.$$

8. Let $X = (X_0, \ldots, X_{66})$ be the string of 67 elements of K defined by:

$$X = (X_0, \ldots, X_{66}) = s^{-1}\Big(\varphi^{-1}(A)\Big).$$

9. The signature S is the 469-bit string given by:

$$S = \pi^{-1}(X_0)||\ldots||\pi^{-1}(X_{66}).$$

5.2 Computing $A = F^{-1}(B)$

The function F, from $\mathcal{L}$ to $\mathcal{L}$, is defined by:

$$\forall A \in \mathcal{L},\ F(A) = A^{128^{33}+1}.$$

As a consequence, $A = F^{-1}(B)$ can be obtained by the following formula:

$$A = B^h,$$

the value of the exponent h being the inverse of $128^{33}+1$ modulo $128^{67}-1$. We have $h =$ 76814678895570674984783396402390903192632828313606573593975318888419579839689 692467845262460558857411822805228876364367384259613839534324934 4.

Finding a fast method of computing $A = B^h$ is not trivial, and is outside the scope of this paper. (It can be done more than ten times faster than naive square and multiply method, and the solution for SFLASH^{v2} is given in [1].)

6 Verifying a signature

Given a message M (*i.e.* a string of bits) and a signature S (a 259-bit string), the following algorithm is used to decide whether S is a valid signature of M or not:

1. Let M_0, M_1, M_2 and M_3 be the three 160-bit strings defined by:

$$M_0 = \text{SHA-1}(M),$$

$$M_1 = \text{SHA-1}(M_0||0x00),$$

$$M_2 = \text{SHA-1}(M_0||0x01).$$

$$M_3 = \text{SHA-1}(M_0||0x02).$$

With 0x00 through 0x02 denoting one single 8-bit character appended to M_0.

2. Let V be the 392-bit string defined by:

$$V = [M_1]_{0\to159}||[M_2]_{0\to159}||[M_3]_{0\to71}.$$

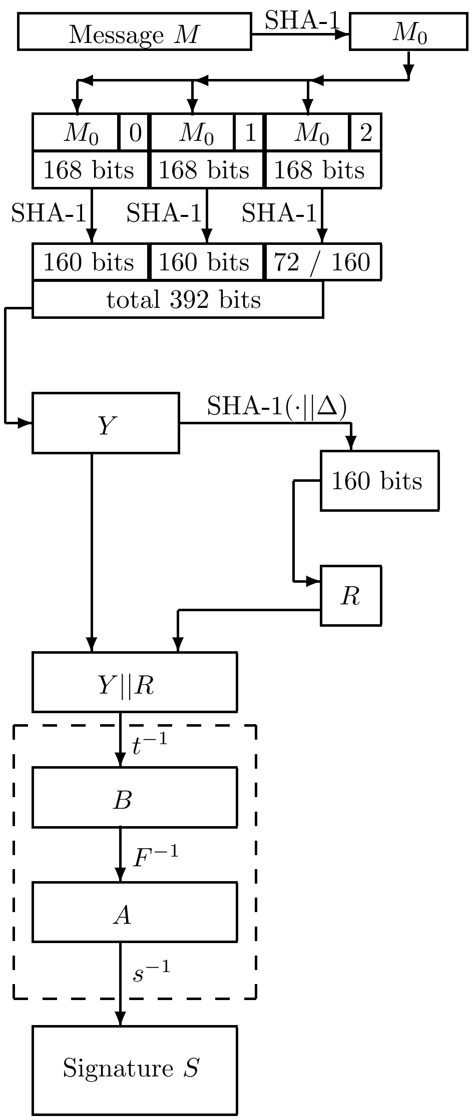

Figure 1: Signature generation with SFLASH

3. Let Y be the string of 56 elements of K defined by:

$$Y = \Big(\pi([V]_{0\to 6}), \pi([V]_{7\to 13}), \ldots, \pi([V]_{385\to 391})\Big).$$

4. Let Y' be the string of 56 elements of K defined by:

$$Y' = G\Big(\pi([S]_{0\to 6}), \pi([S]_{7\to 13}), \ldots, \pi([S]_{385\to 391})\Big).$$

5. • If Y equals Y', accept the signature.
   • Otherwise reject the signature.

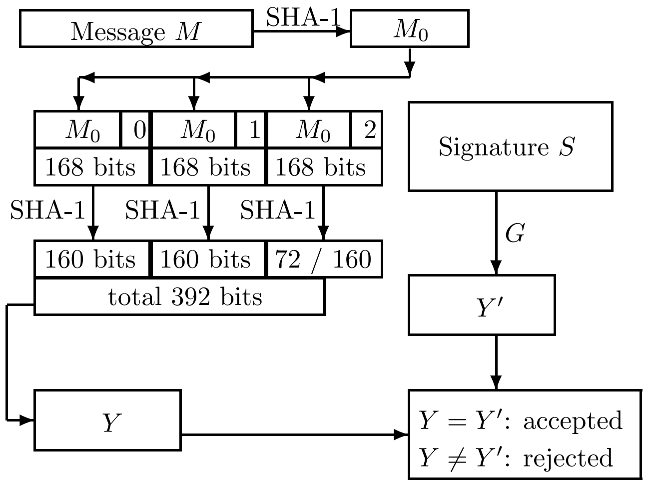

Figure 2: Signature verification with SFLASH

7 Security of SFLASH^{v3}

SFLASH is a signature scheme based on a C^{*--} trapdoor function introduced in [25].

There are three levels on which its security should be considered.

1. The security of signature schemes based on multivariate polynomials versus the security of the underlying trapdoor one-way function. This topic, as well as generic attacks on signature scheme independent on the trapdoor function, have been studied in [9].

2. Structural attacks on C^{*--} and similar multivariate trapdoor functions have been studied in [25, 8, 13, 24, 22]. Originally, C^{*--} is a product of the research on repairing the original Matsumoto-Imai cryptosystem [14] with a method of removing equations introduced independently by Shamir [28] and Patarin [19]. The method itself of removing equations is analysed in [25, 8, 13, 24]. From the recent paper presented at Crypto 2003 [13], it is possible to see that, due to he fact that we are over $GF(2^k)$, and unlike the attacks with Gröbner bases shown for C^{*--} over $GF(2)$ in [8], it is expected that for SFLASH schemes, starting from the previous version SFLASH^{v2}, there is no method known to distinguish them from a random system of quadratic equations over $GF(2^k)$. Some simulations on this can be found in Section 7.2. of [10].

3. Then the security of SFLASH will be based mainly on the problem MQ of solving a system of quadratic equations over $GF(2^k)$ (see [9] to see if can be based only on this assumption). This is a very important hard problem known in cryptography. For example, recently Murphy and Robshaw showed that the security of AES can be expressed as such a problem, see [15, 5]. It also has implications on the security of several stream ciphers, see [6, 7].

This problem MQ is however NP-complete, and is expected to be really very hard in many cases of practical relevance, see [9, 27, 26, 3, 2]. Even though attacks on such problems have known huge huge progress in the recent years, see [27, 26, 11, 12, 3, 2, 5], it is possible to see that they are limited by some algebraic properties of the ideal generated by the public polynomials, that are more or less invariant, whatever is the method used. They are also limited by the speed of well known fundamental algorithms such as linear algebra.

Therefore we hope that these schemes arrive at the maturity point, where their security is fully understood and is not much lower than expected. SFLASH was designed to have a security level of 2^{80} with the present state of the art in Cryptanalysis, as required in the NESSIE project. The best currently known attack on SFLASH^{v3} requires at least 2^{100} operations, see [3] but in fact much more, as the estimations of [3] turn out to be over-optimistic, see [2]. This gives still some security margin to resist future improved attacks.

7.1 Remark - Possibility of Making the Affine Parts Public

This remark has been added after publication of this specification on e-print server, on October 2nd 2003. We received some criticism from W. Geiselmann, R. Steinwandt and Th. Beth about the possibility of recovering the affine parts of SFLASH^{v3}. This result is interesting, except it is not new, see [22, 23]. We were aware of this fact at the time of specifying SFLASH^{v3}. Yet, we claim that it does not matter at all for the security of SFLASH^{v3}. In SFLASH, the parts S_C and T_C may, or may not be made public. Anyone that implements SFLASH^{v3} is free to publish them. However for efficiency reasons, in Section 3.2 we recommended not to publish them.

In general, the current state of knowledge makes us think that, unlike what is suggested in [22, 23], not only the affine parts are **not** a weakness of SFLASH^{v3} but they in fact probably do improve the strength of SFLASH^{v3}. One should not think that since they can be recovered they should be removed (equal to 0). We claim that they should remain random non-zero vectors: indeed, this makes some attacks more difficult, see for example [19].

To summarize:

- ◇ In SFLASH^{v1} and SFLASH^{v2} the parts S_C and T_C were random secrets, and it was discovered that they could have been made public, see [22, 23].
- ◇ In SFLASH^{v3} the parts S_C and T_C are not published either. In fact, they are neither secret, as they can be recovered (see [23]), nor public, as they are not made public. We call them "semi-public".

8 Summary of the characteristics of SFLASH^{v3}

- Length of the signature: 469 bits.
- Length of the public key: 112.3 Kbytes.
- Length of the secret key: the secret key (7.8 Kbytes) can be generated from a small seed of 128 bits or more.
- Time to sign a message[1]: less than 1 ms.
  We expect that it should take less than 200 ms on a Infineon SLE66 component without cryptoprocessor.
- Time to verify a signature[1]: less than 1 ms: (*i.e.* approximately $67 \times 67 \times 56$ multiplications and additions in K).
- Time to generate a pair of public key/secret key: less than 1 s. (cf. [29]).
- Best known attack: more than 2^{90} TDES computations.

SFLASH^{v3} is specified for reference only, the authors still do recommend SFLASH^{v2} that has better performances.

Acknowledgements We would like to thank Bo-Yin Yang, Louis Granboulan, Bart Preneel, Gwenaëlle Martinet, and all the other members of the Nessie project that by quality of their work contributed to maturing of several important branches of cryptography.

[1] On a PC computer. Pessimistic estimation.

References

[1] Mehdi-Laurent Akkar, Nicolas Courtois, Louis Goubin, Romain Duteuil: *A Fast and Secure Implementation of Slash,* PKC 2003, LNCS 2567, Springer, pp. 267-278.

[2] Jiun-Ming Chen, Nicolas Courtois and Bo-Yin Yang: *On Asymptotic Security Estimates in XL and Gröbner Bases-Related Algebraic Cryptanalysis,* ICICS'04, LNCS 3269, pp. 401-413, Springer, 2004.

[3] Nicolas Courtois: *Algebraic Attacks over $GF(2^k)$, Application to HFE Challenge 2 and Sflash-v2.* In PKC 2004, LNCS 2947, pp. 201-217, Springer, 2004.

[4] Alex Biryukov and Christophe De Canniere: *Block Ciphers and Systems of Quadratic Equations.* FSE 2003, to appear in LNCS, Springer, 2003.

[5] Nicolas Courtois and Josef Pieprzyk, *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations*, Asiacrypt 2002, LNCS 2501, pp.267-287, Springer, a preprint with a different version of the attack is available at `http://eprint.iacr.org/2002/044/`.

[6] Nicolas Courtois: *Higher Order Correlation Attacks, XL algorithm and Cryptanalysis of Toyocrypt,* LNCS 2587, pp. 182-199, Springer. An updated version is available at `http://eprint.iacr.org/2002/087/`.

[7] Nicolas Courtois and Willi Meier: *Algebraic Attacks on Stream Ciphers with Linear Feedback,* Eurocrypt 2003, Warsaw, Poland, LNCS 2656, pp. 345-359, Springer. An extended version is available at `http://www.minrank.org/toyolili.pdf`

[8] Nicolas Courtois, Magnus Daum and Patrick Felke: *On the Security of HFE, HFEv- and Quartz,* PKC 2003, LNCS 2567, Springer, pp. 337-350.

[9] Nicolas Courtois: *Generic Attacks and the Security of Quartz,* PKC 2003, , LNCS 2567, pp. 351-364. A preliminary version was presented at the second Nessie workshop, Royal Holloway, University of London, September 2001.

[10] Nicolas Courtois: *Short Signatures, Provable Security, Generic Attacks and Computational Security of Multivariate Polynomial Schemes such as HFE, Quartz and Sflash,* Draft available on `http://eprint.iacr.org/2004/143`.

[11] Jean-Charles Faugère: *A new efficient algorithm for computing Gröbner bases (F_4),* Journal of Pure and Applied Algebra 139 (1999) pp. 61-88. See `www.elsevier.com/locate/jpaa`

[12] Jean-Charles Faugère: *A new efficient algorithm for computing Gröbner bases without reduction to zero (F5),* Workshop on Applications of Commutative Algebra, Catania, Italy, 3-6 April 2002, ACM Press.

[13] Antoine Joux, Jean-Charles Faugère: *Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases*, Crypto 2003, LNCS 2729, pp. 44-60, Springer.

[14] Tsutomu Matsumoto, Hideki Imai: *Public Quadratic Polynomial-tuples for efficient signature-verification and message-encryption,* Eurocrypt'88, Springer 1998, pp. 419-453.

[15] S. Murphy, M. Robshaw: *Essential Algebraic Structure within the AES*, Crypto 2002, LNCS 2442, Springer.

[16] NESSIE Portfolio of recommended cryptographic primitives, available at `https://www.cosic.esat.kuleuven.ac.be/nessie/deliverables/decision-final.pdf`

[17] NESSIE Security Report, revised final version 2.0, available at `https://www.cosic.esat.kuleuven.ac.be/nessie/deliverables/D20-v2.pdf`

[18] Jacques Patarin: *Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88*, In Crypto'95, Springer, LNCS 963, pp. 248-261, 1995.

[19] Jacques Patarin: *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of Asymmetric Algorithms*, Eurocrypt'96, Springer, pp. 33-48. The extended version can be found at `http://www.minrank.org/hfe.ps`

[20] Jacques Patarin, Louis Goubin, Nicolas Courtois: *Flash, a fast multivariate signature algorithm*; Cryptographers' Track Rsa Conference 2001, LNCS 2020, pp. 298-307, Springer.
This paper introduces FLASH and SFLASH^{v1}.

[21] The specification of Sflash^{v2}. Available from: `http://www.cryptosystem.net/sflash/`.

[22] W. Geiselmann, R. Steinwandt, Th. Beth, *Attacking the Affine Parts of SFLASH*, Initially presented at the second NESSIE Workshop, 12-13 September 2001, Egham, UK. Published in IMA Int. Conf. 2001, LNCS 2260, Springer, pp. 355-359, 2001.

[23] W. Geiselmann, R. Steinwandt, and Th. Beth: *Revealing the Ane Parts of SFLASHv1, SFLASHv2, and FLASH.* In Santos Gonzalez Jimenez and Consuelo Martnez Lopez, editors, Actas de la VII Reunion Española de Criptologa y Seguridad de la Informacion. Tomo I, pages 305-314, 2002.

[24] Henri Gilbert, Marine Minier: *Cryptanalysis of SFLASH.* Eurocrypt 2002, LNCS 2232, Springer, pp. 288-298, 2002.

[25] J. Patarin, L. Goubin, N. Courtois, *C^{*-+} and HM: Variations around two schemes of T. Matsumoto and H. Imai*, in Advances in Cryptology, Proceedings of ASIACRYPT'98, LNCS n° 1514, Springer, 1998, pp. 35-49.

[26] Jacques Patarin and Nicolas Courtois *About the XL Algorithm over $GF(2)$*, Cryptographers' Track RSA 2003, LNCS 2612, pages 141-157, Springer 2003.

[27] Adi Shamir, Jacques Patarin, Nicolas Courtois, Alexander Klimov, *Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations*, Eurocrypt'2000, LNCS 1807, Springer, pp. 392-407.

[28] Adi Shamir: *Efficient signature schemes based on birational permutations*; Crypto'93, Springer, pp. 1-12.

[29] Christopher Wolf: *Efficient Public Key Generation for Multivariate Cryptosystems*, `http://eprint.iacr.org/2003/089/`

A Appendix - Summary of Modifications in SFLASH

In this section we explain the difference between the updated version SFLASH^{v3} and the previous versions SFLASH^{v1} and SFLASH^{v2}. A complete history of changes is given.

A.1 Increasing the Number of Variables/Equations

- ◇ In SFLASH^{v1} and SFLASH^{v2} the "big" field is $\mathcal{L} = K[X]/(X^{37} + X^{12} + X^{10} + X^2 + 1)$ with $K = \mathbf{F}_2[X]/(X^7 + X + 1)$.
  As a consequence, in the public key there are 37 variables.
  The number of public equations is 37-11=26, as 11 equations have been removed.
- ◇ In SFLASH^{v3} the big field is $\mathcal{L} = K[X]/(X^{67} + X^5 + X^2 + X + 1)$. The "small" field $K = \mathbf{F}_2[X]/(X^7 + X + 1)$ remains the same.
  As a consequence, in the public key there are 67 variables.
  The number of public equations is 67-11=56, as again exactly 11 equations have been removed.

A.2 Changing the Way the Message is Hashed

- ◇ In SFLASH^{v1} and SFLASH^{v2} the first two steps in computing the signature (and also in verifying it) were:

  1 Let M_1 and M_2 be the three 160-bit strings defined by:

  $$M_1 = \text{SHA-1}(M),$$

  $$M_2 = \text{SHA-1}(M_1).$$

  2 Let V be the 182-bit* string defined by:

  $$V = [M_1]_{0\to 159}||[M_2]_{0\to 21}.$$

  * Note that $182 = (37 - 11) \cdot 7$.

- ◇ In SFLASH^{v3} it is a bit more complicated:

  1 Let M_0, M_1, M_2 and M_3 be the three 160-bit strings defined by:

  $$M_0 = \text{SHA-1}(M),$$

  $$M_1 = \text{SHA-1}(M_0||0x00),$$

  $$M_2 = \text{SHA-1}(M_0||0x01).$$

  $$M_3 = \text{SHA-1}(M_0||0x02).$$

  With 0x00 through 0x02 denoting one single 8-bit character appended to M_0.

  2 Let V be the 392-bit** string defined by:

  $$V = [M_1]_{0\to 159}||[M_2]_{0\to 159}||[M_3]_{0\to 71}.$$

  ** Note that $392 = (67 - 11) \cdot 7$.

This modification allows the whole value V to be modelised by a random oracle, and allows to apply to SFLASH^{v3} the partial provable security results of [9].

A.3 Increasing the Degree of the Internal Polynomial

- ◇ In SFLASH^{v1} and SFLASH^{v2} the internal polynomial is $F : \mathcal{L} \rightarrow \mathcal{L}$ defined by:

$$\forall A \in \mathcal{L},\ F(A) = A^{128^{11}+1}.$$

- ◇ In SFLASH^{v3} the internal polynomial is $F : \mathcal{L} \rightarrow \mathcal{L}$ defined by:

$$\forall A \in \mathcal{L},\ F(A) = A^{128^{33}+1}.$$

This change has, as far as we know, no incidence on the security of SFLASH^{v3}. It has been done to speed-up the signature computation by using less multiplications in $\mathcal{L}$.

A.4 Choosing the coefficients of s and t in GF(128) instead of GF(2)

This change does not concern the current version and was done between the old versions SFLASH^{v1} and SFLASH^{v2}. We recall it to make our SFLASH history complete.

- ◇ In SFLASH^{v1} the coefficients of the secret affine transformations s and t were chosen in the subfield $K' = \mathrm{GF}(2)$ instead of the field $K = \mathrm{GF}(128)$. This very special property allowed to substantially decrease the size of the public key. Unfortunately it also allowed Gilbert and Minier to break SFLASH^{v1}, see [24] for details.

- ◇ In SFLASH^{v2} and SFLASH^{v3} the coefficients of the secret affine transformations s and t are chosen in $K = \mathrm{GF}(128)$. This makes the public key bigger, but also makes attacks such as described in [24] impractical.

  This change also prevents the attack of Geiselmann, Steinwandt and Beth from [22], that can after all be done also in this case, see [23]. However, we claim that this attack is not a threat to the security of SFLASH, see Section 7.1.

A.5 Resulting Changes in Signature Size and in Public Key Size

These are a straight consequence of all the changes described above. We summarize them here:

- ◇ In SFLASH^{v1} the public key had 2.2 Kbytes.
  The signature length was 259 bits.

- ◇ In SFLASH^{v2} the public key had 15.4 Kbytes.
  The signature length was 259 bits.

- ◇ In SFLASH^{v3} the public key has 112.3 Kbytes.
  The signature length is 469 bits.