MQSAS - A Multivariate Sequential Aggregate Signature Scheme

Rachid El Bansarkhani[1], Mohamed Saied Emam Mohamed[1], Albrecht Petzoldt[2]

[1] Technische Universität Darmstadt, Germany
[2] Kyushu University, Japan
{elbansarkhani,mohamed}@cdc.informatik.tu-darmstadt.de
petzoldt@imi.kyushu-u.ac.jp

Abstract. (Sequential) Aggregate signature schemes enable a group of users $u_1, \ldots, u_k$ with messages $m_1, \ldots, m_k$ to produce a single signature Σ which states the integrity and authenticity of all the messages $m_1, \ldots, m_k$. The length of the signature Σ is thereby significantly shorter than a concatenation of individual signatures. Therefore, aggregate signatures can improve the efficiency of numerous applications, e.g. the BGPsec protocol of Internet routing and the development of new efficient aggregate signature schemes is an important task for cryptographic research. On the other hand, multivariate cryptography offers a huge variety of practical signature schemes. However, there is a lack of multivariate signature schemes with special properties such as aggregate signature schemes. In this paper, we propose a technique to extend the HFEv- signature scheme to a sequential aggregate signature scheme. By doing so, we create the first multivariate signature scheme of this kind. Our scheme is very efficient and offers compression rates that outperform current lattice-based constructions for practical parameters.

Keywords: Multivariate Cryptography, HFEv-, Sequential Aggregate Signatures

1 Introduction

Cryptographic techniques are an essential tool to guarantee the security of communication in modern society. Today, the security of nearly all of the cryptographic schemes used in practice is based on number theoretic assumptions such as factoring large integers and solving discrete logarithms. The best known schemes within this domain are RSA, DSA, and ECC. However, such schemes will become insecure as soon as large enough quantum computers are practical. This lack of security can be attributed to Shor's algorithm [26], which solves number theoretic problems such as integer factorization and discrete logarithms in probabilistic polynomial time on a quantum computer. Therefore, alternatives to classical schemes are required, that are based on hard mathematical problems

withstanding quantum attacks. Besides lattice, code and hash based cryptosystems, multivariate cryptography is one of the main candidates for this [1].

Multivariate schemes are in general very fast and require only modest computational resources, which makes them attractive for the use on low cost devices such as smart cards and RFID chips [3,4]. Additionally, at least in the area of digital signatures, there exists a large number of practical multivariate schemes [9,14].

The HFE cryptosystem as proposed by Patarin in [20] is one of the best known and most studied multivariate schemes. While the security of the basic scheme appeared to be very weak, the HFEv- variant seems to represent a good candidate for multivariate signature schemes [21]. The most recent construction, the Gui signature scheme [24], outputs very short signatures (120 bits) while at the same time entailing high performance engines, which are comparable to those of classical schemes such as RSA and DSA.

(Sequential) aggregate signature schemes enable a group of users $U = \{u_1, \ldots, u_k\}$, each of them having a message m_i to be signed, to generate a single signature Σ which guarantees the integrity and authenticity of all the messages $m_1, \ldots, m_k$. The key point hereby is that the length of the aggregate signature Σ is much less than a concatenation of the individual signatures. Therefore, (sequential) aggregate signature schemes have a great deal of application areas and are considered as an important tool in the BGPsec protocol [19], which has the role to secure the global Internet routing system. Each node in a certain path of n hops receives n certificates and the same amount of signatures. It then verifies the signatures and creates its own signature attesting for this path and sends its result together with the previous signatures to the next hop. As a consequence, the number of certificates and signatures increases linearly with the number of nodes on this path. This amount of bandwidth costs can drastically be reduced by the use of a sequential aggregate signature scheme. Similar ideas can in general be applied to public key infrastructures of any depth requiring chains of certificates and signatures in order to authenticate public keys at the leafs. Such schemes come always into use, when chains and paths need to be authenticated as a condition for the protocol to work.

In this paper we show how to extend HFEv- to a multivariate sequential aggregate signature scheme, allowing a set of signers $u_1, \ldots, u_k$, each of them having different keys and different messages m_i, to generate a sequential aggregate signature for all the messages $m_1, \ldots, m_k$. The length of the resulting signature is only slightly larger than a standard HFEv- signature. By use of the public keys pk_i, the verifier can use this signature to check, if every message m_i was indeed signed by signer u_i for $i = 1, \ldots, k$. Our scheme is the first multivariate (sequential) aggregate signature scheme and enables high compression rates and therefore very short sizes of the aggregate signature. Furthermore, with regard

to its performance, our scheme outperforms current lattice-based constructions [2].

The rest of this paper is organized as follows. In Section 2 we repeat the basic concepts of multi and sequential aggregate signatures. Section 3 gives an overview of the area of multivariate cryptography and introduces the HFEv- signature scheme, which is the basis of our construction. In Section 4 we then present our technique to extend the HFEv- based signature scheme to a multivariate multi- and (sequential) aggregate signature schemes. In Section 5, we reduce the security of our scheme from the one-wayness of HFEv- and discuss the security of the underlying scheme. Section 6 gives concrete parameters for our construction and compares our scheme with other existing multi and aggregate signature schemes. Finally, Section 7 concludes the paper.

2 Sequential Aggregate Signatures

In this section we describe the concept of (sequential) aggregate signature schemes. Let $U = \{u_1, \ldots, u_k\}$ be a set of users participating in the protocol, each of them having a key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$ of a digital signature scheme. Each user u_i applies his private key sk_i to generate a signature σ_i for a message m_i (the messages m_i are not necessarily distinct). Let us assume that the k users u_i $(i = 1, \ldots, k)$ desire to prove to a single verifier V that each user u_i signed his message m_i. This could be accomplished by sending all the messages m_i and a signature $\tilde{\sigma} = (\sigma_1, \ldots, \sigma_k)$ to the verifier V. However, the number of signatures σ_i and therefore the length of $\tilde{\sigma}$ to be transmitted grows linearly with the number k of users.
An alternative way of proving the integrity and authenticity of the messages $m_1, \ldots, m_k$ is to combine all the signatures σ_i into a single signature Σ with $|\Sigma| \ll |\tilde{\sigma}|$. To achieve this, one can use a multi or aggregate signature scheme. A multi or an aggregate signature scheme $\mathcal{AS}$ consists of four algorithms `KeyGen`, `Sign`, `Combine` and `Verify`, which are described as follows.

- $\texttt{KeyGen}(1^\ell)$: The probabilistic algorithm `KeyGen` takes as input a security parameter ℓ and outputs a key pair $(\mathsf{sk}, \mathsf{pk})$. In an aggregate signature scheme, this algorithm is performed by every user u_i.
- $\texttt{Sign}(m, \mathsf{sk})$: The (probabilistic) algorithm `Sign` takes as input a message m and a secret key sk and outputs a signature σ for the message m. To generate an aggregate signature, this algorithm is performed by every user u_i.
- $\texttt{Combine}((m_1, \sigma_1), \ldots, (m_k, \sigma_k), (\mathsf{pk}_1, \ldots, \mathsf{pk}_k))$: The algorithm `Combine` takes as input a set of message/signature pairs $(m_1, \sigma_1), \ldots, (m_k, \sigma_k)$ as well as a set of public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_k$ and outputs an aggregate signature Σ.
- $\texttt{Verify}((m_1, \ldots, m_k), \Sigma, (\mathsf{pk}_1, \ldots, \mathsf{pk}_k))$: The deterministic algorithm `Verify` takes as input a set of messages $m_1, \ldots, m_k$, an aggregate signature Σ and a set of public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_k$. It outputs **TRUE**, if Σ is a valid aggregate signature for the messages $m_1, \ldots, m_k$ and the users $u_1, \ldots, u_k$ and **FALSE** otherwise.

In a multisignature scheme, all the messages m_i $(i = 1, \ldots, k)$ are required to be equal, while an aggregate signature scheme allows to combine signatures for arbitrarily chosen messages m_i.

In a sequential aggregate signature scheme, the combining step is done sequentially. The first signer generates a standard signature σ_1 for his message m_1, while the second signer generates a signature σ_2 for his message m_2 and combines it with σ_1 to obtain an aggregate signature Σ_2 for both the messages m_1 and m_2. This step is repeated for the signers $u_3, \ldots, u_k$. The last signer produces the final signature $\Sigma = \Sigma_k$, which is now a valid aggregate signature for all the messages $m_1, \ldots, m_k$. The process of generating a sequential aggregate signature Σ is illustrated in Figure 1.

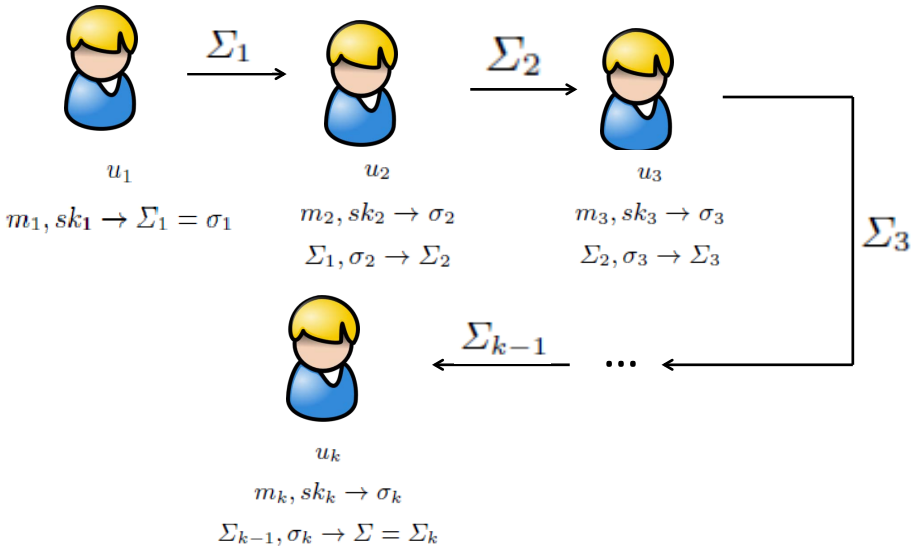

Fig. 1. Generation of a sequential aggregate signature

Compression Rate Let $|\sigma_i|$ be the size of an individual signature σ_i $(i = 1, \ldots, k)$ and $|\Sigma|$ be the size of the (sequential) aggregate signature Σ. Following [2], we define the compression rate of the aggregate signature scheme by

$$\tau(k) = 1 - \frac{|\Sigma|}{\sum_{i=1}^{k} |\sigma_i|}. \tag{1}$$

The size ratio τ expresses therefore the amount of memory that has been saved due to the use of the aggregate signature Σ. In fact, by the aggregate signature

scheme we reduced the signature size from originally $\sum_{i=1}^{k} |\sigma_i|$ to $|\Sigma|$ bits, which results in a compression rate of $\tau(k)$. A value of $\tau = 0$ corresponds to a signature Σ which is as long as the concatenation of all the individual signatures (i.e. no compression at all). A value of $\tau = 1 - \frac{1}{k}$ expresses the state that the aggregate signature Σ has the size of an individual signature, which corresponds to an optimal aggregate signature scheme.

3 The HFEv- Signature Scheme

In this section we review the HFEv- signature scheme, which is the basis of our construction. Before we give a detailed description of the scheme itself, we start with a short overview of the basic concepts of multivariate cryptography.

3.1 Multivariate Cryptography

The basic objects of multivariate cryptography are systems of multivariate quadratic polynomials. (see equation (2)).

$$\begin{aligned} p^{(1)}(x_1, \ldots, x_n) &= \sum_{i=1}^{n} \sum_{j=i}^{n} p_{ij}^{(1)} \cdot x_i x_j + \sum_{i=1}^{n} p_i^{(1)} \cdot x_i + p_0^{(1)} \\ p^{(2)}(x_1, \ldots, x_n) &= \sum_{i=1}^{n} \sum_{j=i}^{n} p_{ij}^{(2)} \cdot x_i x_j + \sum_{i=1}^{n} p_i^{(2)} \cdot x_i + p_0^{(2)} \\ &\vdots \\ p^{(m)}(x_1, \ldots, x_n) &= \sum_{i=1}^{n} \sum_{j=i}^{n} p_{ij}^{(m)} \cdot x_i x_j + \sum_{i=1}^{n} p_i^{(m)} \cdot x_i + p_0^{(m)} \end{aligned} \tag{2}$$

The security of multivariate schemes is based on the MQ problem.

Definition 1 (MQ Problem). *Given m multivariate quadratic polynomials $p^{(1)}(\mathbf{x}), \ldots, p^{(m)}(\mathbf{x})$ in n variables $x_1, \ldots, x_n$ as shown in equation (2), find a vector $\bar{\mathbf{x}} = (\bar{x}_1, \ldots, \bar{x}_n)$ such that $p^{(1)}(\bar{\mathbf{x}}) = \ldots = p^{(m)}(\bar{\mathbf{x}}) = 0$.*

The MQ problem (for $m \approx n$) is proven to be NP-hard even for quadratic polynomials over the field GF(2) [12].

To build a public key cryptosystem based on the MQ problem, one starts with an easily invertible quadratic map $\mathcal{F} : \mathbb{F}^n \to \mathbb{F}^m$ (central map). To hide the structure of $\mathcal{F}$ in the public key, one composes it with two invertible affine (or linear) maps $\mathcal{S} : \mathbb{F}^m \to \mathbb{F}^m$ and $\mathcal{T} : \mathbb{F}^n \to \mathbb{F}^n$. The *public key* is therefore given by $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$. The *private key* consists of $\mathcal{S}$, $\mathcal{F}$ and $\mathcal{T}$ and therefore allows to invert the public key. We note that, due to the above construction, the security of multivariate schemes is not only based on the MQ-Problem but also on the EIP-Problem ("Extended Isomorphism of Polynomials") of finding the composition of $\mathcal{P}$.

In this paper we focus on multivariate signature schemes of the BigField family. For this type of multivariate schemes, the map $\mathcal{F}$ is a specially chosen and easily invertible map over a degree n extension field $\mathbb{E}$ of $\mathbb{F}$. One uses an isomorphism $\Phi : \mathbb{F}^n \to \mathbb{E}$ to transform $\mathcal{F}$ into a quadratic map

$$\bar{\mathcal{F}} = \Phi^{-1} \circ \mathcal{F} \circ \Phi \tag{3}$$

from $\mathbb{F}^n$ to itself. The public key of the scheme is therefore given by

$$\mathcal{P} = \mathcal{S} \circ \bar{\mathcal{F}} \circ \mathcal{T} = \mathcal{S} \circ \Phi^{-1} \circ \mathcal{F} \circ \Phi \circ \mathcal{T} : \mathbb{F}^n \to \mathbb{F}^n. \tag{4}$$

with two invertible affine maps $\mathcal{S}, \mathcal{T} : \mathbb{F}^n \to \mathbb{F}^n$.
The standard signature generation and verification process of a multivariate BigField scheme works as shown in Figure 2.

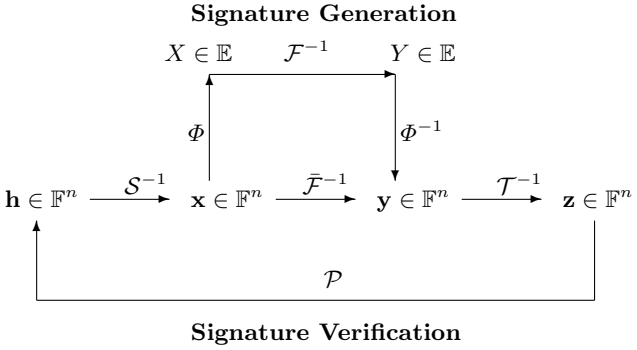

Fig. 2. General workflow of multivariate BigField signature schemes

Signature generation: To generate a signature for a message $\mathbf{h} \in \mathbb{F}^n$, one computes recursively $\mathbf{x} = \mathcal{S}^{-1}(\mathbf{h}) \in \mathbb{F}^n$, $X = \Phi(\mathbf{x}) \in \mathbb{E}$, $Y = \mathcal{F}^{-1}(X) \in \mathbb{E}$, $\mathbf{y} = \Phi^{-1}(Y) \in \mathbb{F}^n$ and $\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y})$. The signature of the message $\mathbf{h}$ is given by $\mathbf{z} \in \mathbb{F}^n$.

Verification: To check the authenticity of a signature $\mathbf{z} \in \mathbb{F}^n$, one simply computes $\mathbf{h}' = \mathcal{P}(\mathbf{z}) \in \mathbb{F}^n$. If $\mathbf{h}' = \mathbf{h}$ holds, the signature is accepted, otherwise rejected.

Two widely used variations of multivariate BigField signature schemes are the Minus variation and the use of additional (Vinegar) variables.
Minus-Variation: The idea of this variation is to remove a small number of equations from the public key. The Minus-Variation was first used in schemes like

SFLASH [22] to prevent Patarin's Linearization Equations attack [23] against the Matsumoto-Imai cryptosystem [18].
Vinegar-Variation: In the vinegar variation one parametrizes the central map $\mathcal{F}$ by adding (a small set of) additional (Vinegar) variables. In the context of multivariate BigField signature schemes, the Vinegar variation can be used to increase the security of the scheme against direct and rank attacks.

A good overview on existing multivariate schemes can be found in [6].

3.2 HFEv-

The HFEv- signature scheme [21] can be described as follows. Let $\mathbb{F} = \mathbb{F}_q$ be a finite field with q elements and $\mathbb{E}$ be a degree n extension field of $\mathbb{F}$. Furthermore, we choose integers D, a and v. Let Φ be the canonical isomorphism between $\mathbb{F}^n$ and $\mathbb{E}$, i.e.

$$\Phi(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i \cdot X^{i-1}. \tag{5}$$

The central map $\mathcal{F}$ of the HFEv- scheme is a map from $\mathbb{E} \times \mathbb{F}^v$ to $\mathbb{E}$ of the form

$$\begin{aligned} \mathcal{F}(X) = & \sum_{0 \le i \le j}^{q^i + q^j \le D} \alpha_{ij} \cdot X^{q^i + q^j} \\ & + \sum_{i=0}^{q^i \le D} \beta_i(v_1, \ldots, v_v) \cdot X^{q^i} \\ & + \gamma(v_1, \ldots, v_v), \end{aligned} \tag{6}$$

with $\alpha_{ij} \in \mathbb{E}$, $\beta_i : \mathbb{F}^v \to \mathbb{E}$ being linear and $\gamma : \mathbb{F}^v \to \mathbb{E}$ being a quadratic function.
Due to the special form of $\mathcal{F}$, the map $\bar{\mathcal{F}} = \Phi^{-1} \circ \mathcal{F} \circ \Phi$ is a quadratic polynomial map from $\mathbb{F}^{n+v}$ to $\mathbb{F}^n$. To hide the structure of $\bar{\mathcal{F}}$ in the public key, one composes it with two affine (or linear) maps $\mathcal{S} : \mathbb{F}^n \to \mathbb{F}^{n-a}$ and $\mathcal{T} : \mathbb{F}^{n+v} \to \mathbb{F}^{n+v}$ of maximal rank.

The *public key* of the scheme is the composed map $\mathcal{P} = \mathcal{S} \circ \bar{\mathcal{F}} \circ \mathcal{T} : \mathbb{F}^{n+v} \to \mathbb{F}^{n-a}$, the *private key* consists of $\mathcal{S}$, $\mathcal{F}$ and $\mathcal{T}$.

Signature generation: To generate a signature for a message $\mathbf{h} \in \mathbb{F}^{n-a}$, the signer performs the following three steps to compute HFEv$-^{-1}(\mathbf{h})$.

1. Compute a pre-image $\mathbf{x} \in \mathbb{F}^n$ of $\mathbf{h}$ under the affine map $\mathcal{S}$.
2. Lift $\mathbf{x}$ to the extension field $\mathbb{E}$ (using the isomorphism Φ). Denote the result by X.
   Choose random values for the vinegar variables $v_1, \ldots, v_v \in \mathbb{F}$ and compute $\mathcal{F}_V = \mathcal{F}(v_1, \ldots, v_v)$.

Solve the univariate polynomial equation $\mathcal{F}_V(Y) = X$ by Berlekamp's algorithm and compute $\mathbf{y}' = \Phi^{-1}(Y) \in \mathbb{F}^n$.
Set $\mathbf{y} = (\mathbf{y}'||v_1||\ldots||v_v)$.

3. Compute the signature $\mathbf{z} \in \mathbb{F}^{n+v}$ by $\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y})$.

Signature verification: To check the authenticity of a signature $\mathbf{z} \in \mathbb{F}^{n+v}$, one simply computes $\mathbf{h}' = \mathcal{P}(\mathbf{z}) \in \mathbb{F}^{n-a}$. If $\mathbf{h}' = \mathbf{h}$ holds, the signature is accepted, otherwise rejected.

3.3 Gui

Recently, Petzoldt et al. proposed the multivariate signature scheme Gui [24], which is based on the concept of HFEv-. In fact, the private and public keys of Gui are just HFEv- keys over the field GF(2) with specially chosen parameters n, D, a and v. Since the number of equations in the public key and therefore the input size of Gui is only 90 bits, it would be possible for an attacker to find two messages m_1 and m_2 whose hash values collide in these first 90 bits. To overcome this problem, the authors of [24] developed a specially designed signature generation process. For this, they used a special parameter l (denoted as repetition factor). The signature generation process of Gui works as shown in Algorithm 1.

Algorithm 1 Signature Generation Process of Gui

Input: HFEv- private key ($\mathcal{S}$, $\mathcal{F}$, $\mathcal{T}$) message $\mathbf{d}$, repetition factor l
Output: signature $\sigma \in \mathrm{GF}(2)^{(n-a)+l\cdot(a+v)}$

1: $\mathbf{h} \leftarrow$ SHA-256($\mathbf{d}$)
2: $S_0 \leftarrow \mathbf{0} \in \mathrm{GF}(2)^{n-a}$
3: **for** $i = 1$ to l **do**
4: $\quad D_i \leftarrow$ first $n-a$ bits of $\mathbf{h}$
5: $\quad (S_i, X_i) \leftarrow \mathrm{HFEv}-^{-1}(D_i \oplus S_{i-1})$
6: $\quad \mathbf{h} \leftarrow$ SHA-256($\mathbf{h}$)
7: **end for**
8: $\sigma \leftarrow (S_l||X_l||\ldots||X_1)$
9: **return** σ

Roughly spoken, one computes HFEv- signatures for l different hash values of the message $\mathbf{d}$ and combines them to a single signature of size $(n-a)+l\cdot(a+v)$. Similarly, the verification algorithm (see Algorithm 2) evaluates the public key l times.

Algorithm 2 Signature Verification Process of Gui

Input: Gui public key $\mathcal{P}$, message $\mathbf{d}$, repetition factor l, signature $\sigma \in \mathrm{GF}(2)^{(n-a)+l(a+v)}$

Output: **TRUE** or **FALSE**

$\mathbf{h} \leftarrow$ SHA-256($\mathbf{d}$)
$(S_l, X_l, \ldots, X_1) \leftarrow \sigma$
for $i = 1$ to l **do**
    $D_i \leftarrow$ first $n - a$ bits of $\mathbf{h}$
    $\mathbf{h} \leftarrow$ SHA-256($\mathbf{h}$)
end for
for $i = l - 1$ to 0 **do**
    $S_i \leftarrow \mathcal{P}(S_{i+1}||X_{i+1}) \oplus D_{i+1}$
end for
if $S_0 = \mathbf{0}$ **then**
    return TRUE
else
    return FALSE
end if

4 Our Sequential Aggregate Signature Scheme

In the design of our multivariate sequential aggregate signature scheme we apply the HFEv- / Gui [24] trapdoor functions, which results in a scheme resembling these basic components. However, while in Gui all the partial signatures are computed using the same private key, we use here for every l-th partial signature another key.

4.1 Key Generation

Let $\mathbb{F} = \mathbb{F}_q$ be a finite field with q elements, $n, D, a, v, l \in \mathbb{N}$ be public parameters and $U = \{u_1, \ldots, u_k\}$ be a set of users. Every user $u_i \in U$ generates an HFEv- key pair $((\mathcal{S}_i, \mathcal{F}_i, \mathcal{T}_i), \mathcal{P}_i)$ according to the given parameter set. Additionally, he computes a public key identity $\mathsf{id}_i = \mathcal{H}(\mathcal{P}_i)$ using a hash function $\mathcal{H}$ modeled as a random oracle. He publishes his public key $\mathcal{P}_i$ and his public key identity id_i while keeping $\mathcal{S}_i, \mathcal{F}_i$ and $\mathcal{T}_i$ secret.

The reason for introducing public key identities in our scheme is the fact, that the public keys serve as input to the hash function multiple times. For large public keys it is therefore more efficient to utilize public identities instead of the public keys itself and thus reducing the input size of the hash functions. This results in faster signature generation engines.

4.2 Signature Generation

Assume that each user u_i has a message m_i to be signed. To generate an aggregate signature Σ for the messages $m_1, \ldots, m_k$, every signer $u_1, \ldots, u_k$ performs

Algorithm 3. The final aggregate signature Σ is given by the output of the signer u_k.

The first signer u_1 just computes a standard HFEv- (Gui) signature for the message m_1 (using Algorithm 1) and returns it as the first aggregate signature Σ_1.

In addition to his own private key and message, each signer u_i ($i \in \{2, \ldots, k\}$) requires as input lists of the public keys $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_{i-1}\}$, the public key identities $\{\mathsf{id}_1, \ldots, \mathsf{id}_{i-1}\}$, and the messages $m_1, \ldots, m_{i-1}$ of the previous signers $u_1, \ldots, u_{i-1}$ and the $(i-1)$-th sequential aggregate signature Σ_{i-1}. Before computing his own signature σ_i and combining it with Σ_{i-1}, u_i checks the correctness of Σ_{i-1} via the verification algorithm (see Algorithm 4).

In order to generate the aggregate signature Σ_i, the signer u_i ($i = 2, \ldots, k$) splits up the input aggregate signature Σ_{i-1} into two blocks $\tilde{S}, \tilde{X}$, where $\tilde{S}$ has a length of $n-a$ bits. He uses his own private key sk_i to compute a standard Gui signature $\sigma_i = (S_l, X_l, \ldots, X_1)$ for the hash value $\mathbf{h} = \mathcal{H}(m_1, \ldots, m_i, \mathsf{id}_1, \ldots, \mathsf{id}_i)$ by running Algorithm 1. After that, he combines his signature $(S_l, X_l, \ldots, X_1)$ with the previous aggregate signature Σ_{i-1}) to generate the new aggregate signature Σ_i. Algorithm 3 shows this process in algorithmic form.

Algorithm 3 Signature Generation Process of MQSAS for each user $i \in 1, \cdots, k$

Input: private key sk_i, message m_i , public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_{i-1}$, public key identities $\mathsf{id}_1, \ldots, \mathsf{id}_{i-1}$, messages $m_1, \cdots, m_{i-1}$, aggregate signature Σ_{i-1}, where $\Sigma_0 = \emptyset$, repetition factor l

Output: aggregate signature Σ_i

1: **if** i = 1 **then**
2: $\quad \tilde{S} = \mathbf{0}^{n-a}$
3: $\quad \tilde{X} \leftarrow \emptyset$
4: **else if** $\mathsf{AggVerify}(i-1, \Sigma_{i-1}, \mathsf{pk}_1, \cdots, \mathsf{pk}_{i-1}, m_1, \cdots, m_{i-1})$ =**TRUE then**
5: $\quad (\tilde{S}, \tilde{X}) \leftarrow \text{split}(\Sigma_{i-1})$
6: **else**
7: $\quad print(''IncorrectSignature'')$
8: $\quad$ **return**
9: **end if**
10: $\mathbf{h} \leftarrow \mathcal{H}(m_1, \ldots, m_i, id_1, \cdots, id_i)$
11: **for** $j = 1$ to l **do**
12: $\quad D_j \leftarrow$ first $n-a$ bits of $\mathbf{h}$
13: $\quad (S_j, X_j) \leftarrow \text{HFEV}-^{-1}(D_j \oplus S_{j-1})$
14: $\quad \mathbf{h} \leftarrow \mathcal{H}(\mathbf{h})$
15: **end for**
16: $\tilde{S} \leftarrow \tilde{S} \oplus S_l$
17: $\tilde{X} \leftarrow (X_l || \ldots || X_1 || \tilde{X})$
18: $\Sigma_i \leftarrow (\tilde{S}, \tilde{X})$
19: **return** Σ_i

4.3 Signature Verification

To check the authenticity of an aggregate signature Σ_i, we parse Σ_i into the sequence of blocks $\tilde{S}, X_{i\cdot l}, \ldots, X_{(i-1)\cdot l+1}, X_{(i-1)\cdot l}, \ldots, X_1$. Here, the length of the block $\tilde{S}$ is $n-a$, while all the other blocks are of length $a+v$. After this, the verification of the aggregate signature Σ_i works very similar to the verification of a Gui signature.

In the j-th iteration, the algorithm first reconstructs the hash values $D_1, \ldots, D_l$ used during the generation of the j-th partial signature. Just as in Algorithm 2 it then evaluates the public key pk_j l times to compute the new value of $\tilde{S}$. At termination, the aggregate signature Σ_i is accepted, if and only if $\tilde{S} = 0$ holds.

Algorithm 4 Verification Process of MQSAS

Input: public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_i$, public key identities $\mathsf{id}_1, \ldots, \mathsf{id}_i$, messages $m_1, \ldots, m_i$, repetition factor l and aggregate signature Σ_i
Output: boolean value **TRUE** or **FALSE**

$(\tilde{S}, X_{i\cdot l}, \ldots, X_{(i-1)\cdot l+1}, X_{(i-1)\cdot l}, \ldots, X_1) \leftarrow \text{split}(\Sigma_i)$
for $j = i$ to 1 **do**
    $\mathbf{h} \leftarrow \mathcal{H}(m_1, \ldots, m_j, \mathsf{id}_1, \cdots, \mathsf{id}_j)$
    for $k = 1$ to l **do**
        $D_k \leftarrow$ first $n-a$ bits of $\mathbf{h}$
        $\mathbf{h} \leftarrow \mathcal{H}(\mathbf{h})$
    end for
    for $k = l-1$ to 0 **do**
        $\tilde{S} \leftarrow pk_j(\tilde{S} || X_{(j-1)\cdot l+k+1}) \oplus D_{k+1}$
    end for
end for
if $\tilde{S} = \mathbf{0}$ **then**
    return TRUE
else
    return FALSE
end if

The Algorithms 3 and 4 show how to instantiate the MQSAS signature scheme with HFEv- / Gui. However we note that our multivariate sequential aggregate signature scheme can also be instantiated on the basis of every other multivariate signature scheme such as Rainbow. Nevertheless, since HFEv- / Gui leads to optimal compression rates, we restrict here to initializing our scheme with HFEv- / Gui.

5 Security

In this section we give an overview of the security of our sequential aggregate signature scheme. In particular, we start with a description of the formal security model adopted in order to provide evidence that the security of our multivariate sequential aggregate signature scheme can be reduced from the one-wayness of HFEv-. Then, we discuss existing attacks on the HFEv- scheme itself in order to estimate the practical security of our construction. We start with some basic assumptions on HFEv-.

Assumption 1 *(One-wayness) The HFEv- function is one-way, i.e. for all PPT adversaries it is hard to find a preimage $x \in \mathbb{F}^{n+v}$ for a given value $y \in \mathbb{F}^{n-a}$ such that $y = \text{HFEv}-(x)$. The advantage of the adversary is given by*

$$\mathbf{Adv}^{ow}_{\text{HFEv}-}(\mathcal{A}) = \text{Prob}|y \xleftarrow{\$} \mathbb{F}^{n-a}, x \leftarrow \mathcal{A} \mid \text{HFEv}-(x) = y| \leq \epsilon,$$

for a negligible function ϵ.

We remark that one-wayness is a minimal required hardness assumption to build from HFEv- a secure digital signature scheme. A trapdoor function is said to be (t, ϵ)-one-way if there exists no algorithm running in time at most t, that outputs a valid preimage with probability at most ϵ.

Theorem 1. *Assuming the (t', ϵ')-one-wayness of HFEv- as per Assumption 1, the sequential aggregate signature scheme presented in Section 4 is $(t, q_H, q_S, n, \epsilon)$-secure against existential forgery under adaptive sequential aggregate chosen-message attack such that*

$$(q_S + q_H + 1) \cdot \epsilon' \geq \epsilon \text{ and } t \leq t' - (4kq_H + 4kq_S + 7k - 1)$$

for all t and ϵ.

Proof (sketch). The security model of our sequential aggregate signature scheme is mainly adopted from [16], where the forger is allowed to control the private keys of all but at least one honest signer. The adversary $\mathcal{A}$ runs in time at most t with at most q_S queries to the signing oracle and q_H hash queries and succeeds in providing a forgery with probability at most ϵ. The number of users is bounded by k. We use a forger to break the one-wayness of the HFEv- function similar to [16]. As a consequence, almost all steps of the security proof go through with only some slight modifications taking into account that we additionally apply an encoder that splits the signature into two parts, where one part is handed over to the signer.

5.1 Practical Attacks against HFEv-

As stated in [24], the relevant attacks against the HFEv- signature scheme and therefore against MQSAS are the MinRank and the direct attack. According to [24], the complexity of the MinRank attack against an HFEv- scheme is at least

$$\text{Complexity}_{\text{MinRank}} = \mathcal{O}(q^{n\cdot(r+v+a-1)} \cdot (n-a)^3). \tag{7}$$

Following the discussion in [24], this complexity is much higher than that of a direct attack against the scheme.

The complexity of a direct attack against a multivariate scheme is mainly determined by the degree of regularity, at which a Gröbner basis algorithm such as F_4 finds a solution. In the case of HFEv-, this degree is upper bounded [10] by

$$\mathrm{d}_{\text{reg}} \leq \begin{cases} \frac{(q-1)\cdot(r-1+a+v)}{2} + 2 & q \text{ even and } r+a \text{ odd} \\ \frac{(q-1)\cdot(r+a+v)}{2} + 2 & \text{otherwise} \end{cases}, \tag{8}$$

where $r = \lfloor \log_q(D-1) \rfloor + 1$. In [24] it was shown that this upper bound is relatively tight.

In [24], the authors performed numerous experiments with direct attacks on HFEv- schemes over fields of characteristic 2. We therefore follow from their results that our parameter sets over GF(2) fulfill the proposed security levels.

However, in order to reduce the repetition factor l of the Gui scheme and therefore to improve the compression rate of MQSAS, we aim at implementing the scheme over larger fields such as GF(7), too. To estimate the security of these schemes, we performed a number of experiments. We implemented the HFEv- scheme in MAGMA and solved the public systems using the F_4 [11] algorithm integrated in MAGMA. Before applying the algorithm, we fixed $(v+a)$ variables to create determined systems and added the field equations $x_i^q - x_i = 0$ $(i = 1, \ldots, n-a)$. Table 1 shows the results. For each parameter set we performed 10 experiments. The values presented in the table are the average of the single results.

HFEv-	$D=8,\ a=v=2$			$D=8,\ a=v=3$		
$n-a$	10	12	14	10	12	14
d_{reg}	8	9	10	9	10	11
time (s)	10.1	395	12,628	15.7	538	15,874
memory(MB)	25.7	220	2,837	32.8	1,057	13,728

Table 1. Direct attacks on HFEv- signature schemes over GF(7)

As the table shows, the degree of regularity at which F_4 finds a solution to our systems is quite high. In particular, with the parameter set $(D, a, v) = (8, 2, 2)$ we can reach a degree of regularity of at least 10, while $(D, a, v) = (8, 3, 3)$ leads to $d_{\text{reg}} \geq 11$.

The complexity of a direct attack against a multivariate scheme can be estimated by

$$\text{Complexity}_{F_4/F_5} = 3 \cdot T^2 \cdot \tau = 3 \cdot \binom{n-a}{d}^2 \cdot \binom{n-a}{2}. \tag{9}$$

By substituting the values of n and d_{reg} into this formula, we find that the complexity of a direct attack against the schemes listed in Table 2 will be higher than the proposed levels of security. Note that this estimation is very conservative, since we assume that the degree of regularity will not rise above 10 and 11 respectively.

6 Parameters and Comparison

In this section we propose concrete parameter sets for the MQSAS scheme and compare its compression capabilities and performance to that of other (sequential) aggregate signature schemes. In particular, we propose 5 parameter sets for 80-bit security and 2 parameter sets for 120-bit security, allowing a trade off between compression rate and performance (see Table 2). The parameters shown in the table are chosen in such a way that the complexity of a direct attack against the scheme is beyond the proposed level of security.

security level (bit)	MQSAS $(\mathbb{F}, n, D, a, v, l)$	public key size (kB)	private key size (kB)	$\lvert\sigma\rvert$ (bit) (20 signers)	compression factor $\tau(20)$
80	(GF(2),96,5,6,6,2)	57.7	2.4	570	0.75
	(GF(2),95,9,5,5,2)	55.5	2.3	490	0.78
	(GF(2),94,17,4,4,2)	53.3	2.3	410	0.81
	(GF(2),96,65,2,2,2)	55.7	2.3	254	0.88
	(GF(7),62,8,2,2,1)	47.1	2.9	420	0.89
120	(GF(2),127,9,4,6,2)	133.8	4.1	523	0.81
	(GF(7),93,8,3,3,1)	156.7	6.4	630	0.90

Table 2. Proposed Parameters and resulting key sizes for the MQSAS scheme

To avoid the above mentioned double signing, we also propose parameters for the MQSAS scheme over GF(7). We can efficiently store 14 bits in 5 GF(7)-elements, while an element of GF(7) is stored in 3 bits. By doing so, we need

60 GF(7)-elements to store a hash value of 160 bits (80-bit security), while 90 GF(7) elements are needed to store a hash value of 240 bits (120-bit security).

Note that the key sizes listed in Table 2 are those of a single signer u_i. The verifyer of the aggregate signature $\Sigma = \Sigma_k$ is faced with a public key of size k times the value listed in the table.

The size of a sequential aggregate signature of our scheme is given by

$$|\Sigma| = (n-a) + k \cdot l \cdot (a+v), \tag{10}$$

the compression rate τ is given by

$$\tau = 1 - \frac{|\Sigma|}{k \cdot |\sigma|} = 1 - \frac{1}{k} \cdot \left(1 + \frac{(k-1) \cdot l \cdot (a+v)}{(n-a) + l \cdot (a+v)}\right). \tag{11}$$

Table 3 and Figure 3 show the signature sizes and the compression factor τ for the above parameter sets and different numbers of users.

MQ-SAS	5 signers		10 signers		20 signers		50 signers		100 signers	
$(\mathbb{F}, n, D, a, v, l)$	$\lvert\Sigma\rvert$ (bit)	τ	$\lvert\Sigma\rvert$ (bit)	τ	$\lvert\Sigma\rvert$ (bit)	τ	$\lvert\Sigma\rvert$ (bit)	τ	$\lvert\Sigma\rvert$ (bit)	τ
(GF(2),96,5,6,6,2)	210	0.63	330	0.71	570	0.75	1,290	0.77	2,490	0.78
(GF(2),95,9,5,5,2)	190	0.65	290	0.74	490	0.78	1,090	0.80	2,090	0.81
(GF(2),94,17,4,4,2)	170	0.68	250	0.76	410	0.81	890	0.83	1,690	0.84
(GF(2),96,65,2,2,2)	134	0.73	174	0.83	254	0.88	494	0.90	894	0.91
(GF(7),62,8,2,2,1)	240	0.75	300	0.84	420	0.89	780	0.92	1,380	0.93
(GF(2),127,9,4,6,2)	223	0.69	323	0.77	523	0.82	1,123	0.84	2,123	0.85
(GF(7),93,8,3,3,1)	360	0.75	450	0.84	630	0.89	1,170	0.92	2,070	0.93

Table 3. Signature sizes and compression rates of the MQ-SAS scheme

6.1 Implementation

To estimate the performance of MQSAS, we created an implementation of our scheme in C. For the implementation of the underlying HFEv- scheme we thereby adapted the implementation of Gui [24] to our setting. Table 4 shows the computation time needed to generate an aggregate signature for different number of users and parameter sets. The experiments were run on a PC with a Core-i5 3750k processor (Ivy Bridge) at 2.4 GHz and with 16 GB of RAM. The timings in the table are the average values of 500 signature generation processes.

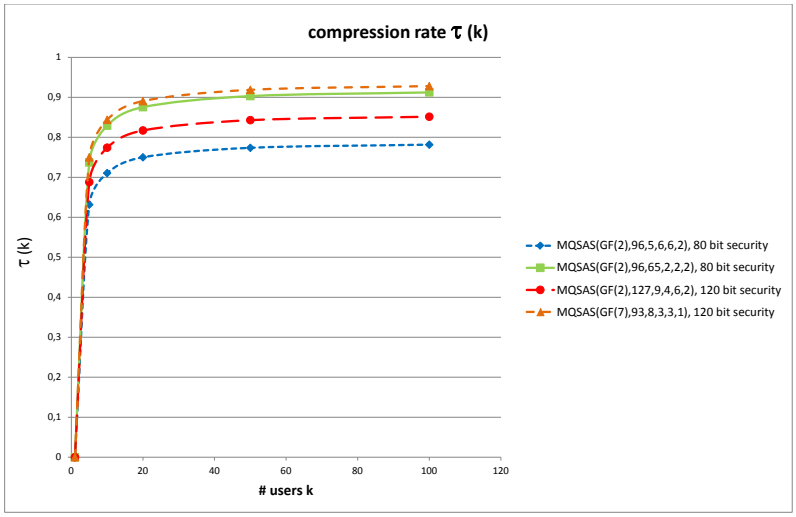

Fig. 3. Compression rate τ of the MQ-SAS scheme

MQ-SAS	signature generation time (ms)				
$(\mathbb{F}, n, D, a, v, l)$)	5 signers	10 signers	20 signers	50 signers	100 signers
(GF(2),96,5,6,6,2)	1.24	3.13	6.11	27.86	82.91
(GF(2),95,9,5,5,2)	2.94	6.75	8.52	36.31	94.25
(GF(2),94,17,4,4,2)	9.30	17.19	32.32	68.18	158.01
(GF(2),127,9,4,6,2)	3.42	5.23	11.61	45.3	141.4
(GF(2),96,65,2,2,2)	150.86	302.23	599.24	1509.69	3053.25

Table 4. Signature generation time of MQSAS

6.2 Discussion

Table 2 indicates that we achieve very short signatures and high compression rates at security levels of both 80-bit and 120-bit. For example, for 80-bit security, our scheme allows to generate an aggregate signature for 20 signers of length only 254 bits, which is less than one quarter of a single RSA signature at the same security level. Remarkably, the compression rates for 120-bit security are even higher than in the 80-bit case. On the other hand, the key sizes are considerably larger for the corresponding parameter sets.

Table 3 shows the aggregate signature sizes and associated compression rates for different parameter sets and number of users. The highest compression rates can hereby be achieved by the two schemes MQ-SAS$(\mathrm{GF}(7), 62, 8, 2, 2, 1)$ and MQ-SAS$(\mathrm{GF}(7), 93, 8, 3, 3, 1)$. The reason for this is that, for these parameter sets, we can avoid the double signing of Gui. As the table shows, these parameter sets allow compression rates of up to 93 %, which means that we need only the size of 7 individual signatures to prove the validity of an aggregate signature for 100 signers (see also Figure 3).

In Table 4 we provide the timings for the signing and verification engine of our scheme. In fact, we note, that each signing step by construction invokes the verification engine in order to check the actual aggregate in terms of validity, before the signer is able to proceed. The timings indicate that the parameter set $(\mathrm{GF}(2), 96, 65, 2, 2, 2)$, due to the high degree of the HFE polynomial in use, is much slower than the remaining ones. We therefore observe a trade off between the compression rate and the performance of the scheme.

6.3 Comparison to other aggregate signature schemes

In this subsection we compare our sequential aggregate signature scheme with other constructions. In fact, we observe that multivariate-based sequential aggregate signature schemes are more suitable for practice than their counterparts from classical and lattice-based cryptography. In terms of signature sizes and performance, HFEv- has been shown to be far more efficient than the other schemes. The size of an individual signature is only slightly more than one hundred bits, whereas the underlying signature schemes of the other sequential aggregate signature schemes occupy memory of size thousands of bits.

With regard to the performance, the timings of our scheme for signing and verification are also significantly better than that of the other schemes. We have shown that the overhead, that our sequential aggregate signature scheme entails, can be at least as low as 7 bits per signature for a reasonable level of security. Hence, almost the whole signature of a signer is concealed within the signature of his successor. Furthermore, the arithmetic operations of our scheme are mainly performed over the field $\mathbb{F}_2$, which is well studied and thus allows to carry out fast

operations such as the encoding function in order to hide the signature in a syndrome to be signed by the next signer in the chain. Lattice-based systems work over $\mathbb{Z}_q^n$, which implies to carry out more complex arithmetic operations such as reductions modulo q for at least $n \geq 256$ components. For RSA-based sequential aggregate signature schemes it is more difficult to instantiate the scheme.

In fact, it is important to agree on how to choose the hash functions beforehand, since the domains of the participating signers differ. Furthermore, the operations are more complex and hence lead to a less efficient scheme. This has also been observed in [2]. The advantages of our scheme are evident emerging MQSAS as a real post-quantum candidate for the BGPsec [19] protocol and other applications. Our scheme outperforms all current schemes in terms of performance and signature sizes.

7 Conclusion

In this paper we proposed a multivariate sequential aggregate signature scheme on the basis of the HFEv- signature scheme, which, to our knowledge, is the first multivariate signature scheme of this kind. Due to the use of HFEv- / Gui as the underlying signature scheme, the resulting signatures are very short (less than 1 kbit for 100 signers at 80 bits of security) and we achieve high compression rates (up to 93 % for $k = 100$ signers). Furthermore, due to the efficiency of arithmetic operations over GF(2), our scheme outperforms all current (sequential) aggregate signature schemes in terms of performance.

References

1. D.J. Bernstein, J. Buchmann, E. Dahmen (eds.): Post Quantum Cryptography. Springer, 2009.
2. R. El Bansarkhani, J. Buchmann: Towards Lattice Based Aggregate Signatures. AFRICACRYPT 2014, LNCS vol. 8469, pp. 336- 355. Springer, 2014.
3. A. Bogdanov, T. Eisenbarth, A. Rupp, C. Wolf. Time-area optimized public-key engines: MQ-cryptosystems as replacement for elliptic curves? CHES 2008, LNCS vol. 5154, pp. 45-61. Springer, 2008.
4. A.I.T. Chen, M.-S. Chen, T.-R. Chen, C.-M. Cheng, J. Ding, E. L.-H. Kuo, F. Y.-S. Lee, B.-Y. Yang. SSE implementation of multivariate PKCs on modern x86 cpus. CHES 2009, LNCS vol. 5747, pp. 33 - 48. Springer, 2009.
5. N.T. Courtois, M. Daum, P. Felke: On the Security of HFE, HFEv- and QUARTZ. PKC 2003, LNCS vol. 2567, pp. 337 - 350. Springer 2003.
6. J. Ding, J. E. Gower, D. S. Schmidt: Multivariate Public Key Cryptosystems. Springer, 2006.
7. J. Ding, T. Hodges: Inverting HFE Systems Is Quasi-Polynomial for All Fields. CRYPTO 2011, LNCS vol. 6841, pp. 724-742. Springer 2011.
8. J. Ding, T. Kleinjung: Degree of regularity for HFE-. IACR eprint 2011/570.
9. J. Ding, D. S. Schmidt: Rainbow, a new multivariate polynomial signature scheme. ACNS 2005, LNCS vol. 3531, pp. 164-175. Springer 2005.

10. J. Ding, B.Y. Yang: Degree of Regularity for HFEv and HFEv-. PQCrypto 2013, LNCS vol. 7932, pp. 52-66. Springer, 2013.
11. J.C. Faugère: A new efficient algorithm for computing Gröbner bases (F4). Journal of Pure and Applied Algebra 139, pp. 61-88 (1999).
12. M. R. Garey and D. S. Johnson: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Company 1979.
13. D. Kravitz: Digital Signature Algorithm. US patent 5231668 (July 1991).
14. A. Kipnis, L. Patarin, L. Goubin: Unbalanced Oil and Vinegar Schemes. EUROCRYPT 1999, LNCS vol. 1592, pp. 206–222. Springer 1999.
15. A. Kipnis, A. Shamir: Cryptanalysis of the HFE Public Key Cryptosystem. CRYPTO 99, LNCS vol. 1666, pp. 19 - 30. Springer 1999.
16. A. Lysyanskaya, S. Micali, L. Reyzin, H. Shacham: Sequential Aggregate Signatures from Trapdoor Permutations. EUROCRYPT 2004, LNCS vol. 3027, pp. 74-90. Springer 2004.
17. M. S. E. Mohamed, J. Ding and J. Buchmann: Towards Algebraic Cryptanalysis of HFE Challenge 2, ISA 2011, Communications in Computer and Information Science vol. 200, pp. 123-131. Springer 2011.
18. T. Matsumoto, H. Imai: Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. EUROCRYPT 1988. LNCS vol. 330, pp. 419-453. Springer 1988.
19. Network Working Group: A Border Gateway Protocol (BGP-4). RFC 4271, https://tools.ietf.org/html/rfc4271.
20. Patarin, J.: Hidden Field equations (HFE) and Isomorphisms of Polynomials (IP). In: Proceedings of EUROCRYPT'96, pp. 38–48, Springer, Heidelberg (1996)
21. J. Patarin, N. Courtois, L. Goubin: QUARTZ, 128-Bit Long Digital Signatures. CTRSA 2001, LNCS vol. 2020, pp. 282-297. Springer, 2001.
22. J. Patarin, N. Courtois, L. Goubin: Flash, a fast multivariate signature algorithm. CTRSA 2001, LNCS vol. 2020, pp. 298 - 307. Springer, 2001.
23. J. Patarin: Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt 88. CRYPTO 95. LNCS vol. 963, pp. 248 - 261. Springer 1995.
24. A. Petzoldt, M.S. Cheng, B.Y. Yang, C. Tao, J. Ding: Design Principles for HFEv-based Signature Schemes. Asiacrypt 2015. To appear.
25. R. L. Rivest, A. Shamir, L. Adleman: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Commun. ACM 21 (2), pp. 120-126 (1978).
26. P. Shor: Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, SIAM J. Comput. 26 (5), pp. 1484 - 1509 (1997).