A Multi-Signature Scheme based on Coding Theory

Mohammed Meziani and Pierre-Louis Cayrel
CASED–Center for Advanced Security Research Darmstadt
Mornwegstrasse 32, 64293 Darmstadt, Germany
Email: {mohammed.meziani, pierre-louis.cayrel@cased.de}

Abstract—In this paper we propose two first non-generic constructions of multisignature scheme based on coding theory. The first system make use of the CFS signature scheme and is secure in random oracle while the second scheme is based on the KKS construction and is a few times. The security of our construction relies on a difficult problems in coding theory: The Syndrome Decoding problem which has been proved NP-complete [4].

Keywords—Post-quantum cryptography, Coding-based cryptography, Digital signature, Multisignature scheme.

I. INTRODUCTION

Digital signature schemes, similar to handwritten signatures, are a fundamental cryptographic primitive used in practice for authenticity and non-repudiation of messages. Several signature schemes exist, but most of them are based on the computational difficulty of solving number theoretic problems such factoring problem, discrete logarithm problem in the multiplicative group of a prime field or in the group of points of an elliptic curve over a finite field. But, in the event of quantum computers all these schemes could be broken due to Shor’s algorithm [29] proposed in 1997. Indeed, the Shor’s algorithm can solve both the factoring problem and the discrete logarithm problem in finite fields and on elliptic curves in polynomial time. Therefore, the cryptographic community has to investigate other mathematical problems that are believed to be hard to solve by quantum algorithms. Among these there are problems in coding theory using error correcting codes. The problem of decoding general codes is such a problem, which has been proven to be NP-complete by Berlekamp, McEliece and Van Tilborg [4].

In 1978, McEliece [22] first proposed an asymmetric cryptosystem which is based on the coding theory and derives its security from the general decoding problem. No efficient attack on this schemes has been found up to date, though numerous computationally intensive attacks have been published in the literature [5], [12]. The idea behind this scheme is to first select a particular (linear) code for which an efficient decoding algorithm is known, and then to use a trapdoor function to disguise the code as a general linear code.

The encryption in the McEliece cryptosystem is not invertible, and therefore it cannot be used for authentication or signature schemes, this is indeed why very few signature schemes based on coding theory have been proposed. This problem was open until 2001 in when Courtois et.al [9] showed how to achieve a code-based signature scheme whose security is based on the syndrome decoding problem. While this problem is NP-complete, their construction is still inefficient for large numbers of errors. Recently, a few code-based signature schemes with additional properties have been published and most of them make use the construction proposed in [9].

ID-based cryptography. The motivation behind the identity based cryptography, proposed by Shamir in 1984 [28], was to simplify the PKI requirements. Instead of using the public key, a user can use his identity (e.g. e-mail address or IP-address) while the associated secret key can be issued by a trusted key generation center (KGC) thanks to a master secret key that only the KGC knows. And thereby some of the costs associated to PKI and certificates can be avoided. Despite this, the identity-based cryptography suffers from a major drawback since a complete trust must be placed on the KGC. This problem is known as the key escrow problem. To overcome this problem, a solution has been proposed in [6] which consists in employing multiple KGCs to jointly produce the master secret key.

Multisignature schemes. A multisignature scheme (MSS) is a normal signature scheme that enables a group of users to cooperatively sign the same document and can be verified by any user. Multisignature schemes have many practical uses such as signing legal electronic documents (e.g. contracts, cheque, etc) by multiple managers in a company. Based on the nature of the application scenarios, the multisignature schemes are divided into categories depending on the signing manner: serial and parallel signing. In the first case, the resulting multisignature is equal to the signature generated by the last signer. More precisely, a signer produces his own signature on a document then broadcasts it to the next signer which after verifying it signs the received components and so on. Here the signing order property should be taken into account. That is, the resulting multisignature depends on the signing order. In the second case the multisignature is produced by a designated signer, called a clerk, which has to collect individual signatures generated by each signer and then combine them into a single signature.

Multisignature schemes have been first introduced in [16]. However, these schemes have an efficiency issue because the generation and the verification cost of the multisignature increases linearly with the number of signers. Since then, various multisignature schemes have been realized. For example, multisignature schemes that are based on RSA assumption [15], [14], [26], constructed form bilinear maps...
II. CODING THEORY BACKGROUND

This section will first provide a brief introduction to coding theory, then give the basic definitions and list some hard problems we use throughout this paper.

A. Coding theory

The term coding theory refers to a broad branch of mathematics concerned with transmitting data across noisy channels and recovering the message. It provides secure transmission of messages, in the sense that any errors which are introduced during the transmission can be corrected.

B. Notations and Definitions

Let $\mathbb{F}_q$ denote the finite field with $q$ elements.

a) Codes: An $(n, k)$-code over $\mathbb{F}_q$ is a linear subspace $C$ of the linear space $\mathbb{F}_q^n$. Elements of $\mathbb{F}_q^n$ are called words and elements of $C$ are codewords. We call $n$ the length, and $k$ the dimension of the code. If $q = 2$, the code is called binary, and is denoted by $[n, k]$.

b) Hamming distance, Hamming weight: The Hamming distance $d(x, y)$ between two words $x, y \in \mathbb{F}_q^n$ counts the number of positions in which $x$ and $y$ differ. More formally, denote $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$. Then $d(x, y) = \left| \{ i : x_i \neq y_i \} \right|$. Here, we use $|S|$ to denote the number of elements of a set $S$. The Hamming weight (or just weight) of a word $x \in \mathbb{F}_q^n$ is denoted by $wt(x)$ and represents the number of non-zero entries of this word, i.e., $wt(x) = d(0, x)$, where $0$ is the vector containing $n$ 0's.

c) Minimum distance: The minimum distance $d$ of an $(n, k)$-code $C$ is the minimum Hamming distance between two codewords, i.e., $d = \min_{x, y \in C, x \neq y} d(x, y)$.

d) Generator matrix, systematic codes: A generator matrix of an $(n, k)$-linear code $C$ is a $k \times n$ matrix $G$ whose rows form a basis for the vector subspace $C$, i.e., $C = \{ xG : x \in \mathbb{F}_q^k \}$. Notice that $C$ is not unique for a code $C$. We call a code systematic if it can be characterized by a generator matrix $C$ of the form $C = (I_k \times k | A_{k \times (n-k)})$, where $I_k \times k$ is the $k \times k$ identity matrix and $A$ an $k \times (n-k)$ matrix.

e) Parity-check matrix, dual code: A parity-check matrix of an $(n, k)$-linear code $C$ is an $(n-k) \times n$ matrix $H$ whose rows form a basis of the orthogonal complement of the vector subspace $C$, i.e., it holds that, $C = \{ c \in \mathbb{F}_q^n : Hc^T = 0 \}$. Note that $H$ can be viewed as the generator matrix of an $(n, n-k)$ linear code $C^\perp$ containing codewords $c$ such that for all codewords $c \in C$, it holds that $c^Tc = 0$. The $C^\perp$ is generally referred to as the dual of $C$.

f) Syndrome: Let $H$ be a parity check matrix of the code $C$. The syndrome of a word $x \in \mathbb{F}_q^n$ is a vector $s \in \mathbb{F}_q^{n-k}$ defined by $s = Hx^T$.

C. Hard problems

In what follows, we recall some hard problems. The security of most code-based cryptosystems is related to hardness of solving these problems.
1) Syndrome Decoding problem (SD):
- Input: An \( r \times n \) matrix \( H \) over \( \mathbb{F}_q \), a target vector \( s \in \mathbb{F}_q^r \) and an integer \( t > 0 \).
- Question: Is there a vector \( x \in \mathbb{F}_q^n \) of weight \( \leq t \), such that \( s = Hx^T \)?

This problem has been proved to be NP-complete by Berlekamp, McEliece, and van Tilborg [4] in 1978 for the general class of binary linear codes. In 1994, Barg [1] extended this result over linear codes defined over \( \mathbb{F}_q \). NP-completeness ensures that this problem can not be solved in polynomial time in the worse case, meaning that there are some hard instances, not that every instance is hard.

To end this section, we state another hard problem, Goppa Parametrized Bounded Decoding (GPBD), which is a variation of SD problem and have been proved to be NP-complete by Finiasz [11] in 2004.

2) Goppa Parametrized Bounded Decoding (GPBD):
- Input: An \((n-k) \times n\) matrix \( H \) over \( \mathbb{F}_2 \) and a syndrome \( s \in \mathbb{F}_2^{n-k} \).
- Question: A word \( x \in \mathbb{F}_2^n \) of weight \( \leq \frac{n-k}{\log_2(n)} \), such that \( Hx^T = s \) ?

The Niederreiter PKC

The Niederreiter PKC is a code-based public key cryptosystem: signing a document requires to hash it into a syndrome and then to try to decode this syndrome. However, for a \( t \)-error correcting Goppa code of length \( n = 2^m \), only a fraction of approximately \( 1/t! \) of the syndromes are decodable. Thus, a counter is appended to the message and the signer tries successive counter values until the hash value is decodable. The signature consists of both the error pattern of weight \( t \) corresponding to the syndrome, and the value of the counter giving this syndrome.

Algorithm 2 The CFS signature

Key Generation:
- Pick a random parity check matrix \( H \) of a \((n,k)\)-binary Goppa code correcting up to \( t \) errors and having a decoding algorithm \( \gamma \).
- Construct the matrices \( Q, \tilde{H} \) and \( P \) as in Algorithm 1.

Signature: To sign a message \( m \)
1) \( i \to i + 1 \)
2) \( x' = \gamma(Q^{-1}h(m\|i)) \)
3) if no \( x' \) was found go to 1
- Output \( (i, x'P) \)

Verification:
- Compute \( s' = Hx'^T \) and \( s = h(m\|i) \).
- The signature is valid if \( s \) and \( s' \) are equals.

2) Security: In [12], the authors present an attack against the CFS scheme due to Daniel Bleichenbacher. Due to this attack, the values of \( m \) and \( t \) used in the CFS scheme have to change. The authors of [12] propose \( m = 21 \) and \( t = 10 \), or \( m = 19 \) and \( t = 11 \), or \( m = 15 \) and \( t = 12 \), as new parameters for a security of more than \( 2^{80} \) binary operations. Due to this attack, the values of \( m \) and \( t \) used in the CFS scheme have to change. The authors of [12] propose \( m = 21 \) and \( t = 10 \), or \( m = 19 \) and \( t = 11 \), or \( m = 15 \) and \( t = 12 \), as new parameters for a security of more than \( 2^{80} \) binary operations.

3) Security proof in the random oracle model: In [10], the author proposes to choose this counter randomly in \( \{1, \ldots, 2^{n-k}\} \), and then obtain a proof of security in the random oracle model.

B. KKS signature scheme

Kabatianskii et al. [17] proposed a signature scheme based on arbitrary linear error-correcting codes. Actually, they proposed to use a linear application \( f \). Three versions are given which are presented in the sequel but all have one point in common: for any \( m \in \mathbb{F}_q^k \), the signature \( f(m) \) is a codeword
of a linear code \( \mathcal{U} \). Each version of KKS proposes different linear codes in order to improve the scheme. We now give a full description of their scheme.

1) \textbf{Description:} Firstly, we suppose that \( C \) is defined by a random parity check matrix \( H \). We also assume that we have a very good estimate \( d \) of its minimum distance. Next, we consider a linear code \( \mathcal{U} \) of length \( n' \leq n \) and dimension \( k \) defined by a generator matrix \( G = [g_{ij}] \). We suppose that there exist two integers \( t_1 \) and \( t_2 \) such that \( t_1 \leq w(u) \leq t_2 \) for any non-zero codeword \( u \in \mathcal{U} \).

Let \( J \) be a subset of \( \{1, \ldots, n\} \) of cardinality \( n' \). \( H(J) \) be the submatrix of \( H \) consisting of the columns \( h_i \) where \( i \in J \) and define an \( r \times n' \) matrix \( F \) such that \( f \in \mathbb{F}_q^k \rightarrow \mathbb{M}_n \) is then defined by \( f(m) = mG^T \) for any \( m \in \mathbb{F}_q^k \) where \( G^* = [g_{i,j}^*] \) is the \( k \times n \) matrix with \( g_{i,j}^* = g_{i,j} \) if \( j \in J \) and \( g_{i,j}^* = 0 \) otherwise. The public application \( \chi \) is then \( \chi(m) = Fm^T \) because \( HG^T = H(J)G^T \). The main difference with Niederreiter signatures resides in the verification step where the receiver checks that:

\[
t_1 \leq w(z) \leq t_2 \quad \text{and} \quad F \cdot m^T = H \cdot z^T.
\]

\textbf{Algorithm 3 The KKS signature}

\textbf{Key Generation:}
- Select two positive integers \( t_1 \) and \( t_2 \) s.t. \( t_1 \leq t_2 \).
- Pick a random parity check matrix \( H = [I_r | D] \) of an \( (n, n-r) \)-code.
- Construct the matrices \( Q, H \) and \( P \) as in Algorithm 1.

\textbf{Signature:}
- To sign a message \( m \):
  - (1) \( i \leftarrow i + 1 \)
  - (2) \( x' = \gamma (Q^{-1}h(m)|i)) \)
  - (3) if no \( x' \) was found go to 1
- Output \((i, x'P)\)

\textbf{Verification:}
- Compute \( s' = Hx^T \) and \( s = h(m)|i) \).
- The signature is valid if \( s \) and \( s' \) are equal.

It has been proved in [8], that this scheme is few times.

IV. OUR PROPOSED SERIAL MULTISIGNATURE SCHEMES

Before presenting our constructions, we give first the formal definition of a multisignature scheme. We denote by \( \mathcal{S} = \{S_1, \ldots, S_N\} \) the set of \( N \) signers intended to sign the message \( M \).

A. Definition

A multisignature scheme \( \mathcal{MS} \) consists of three algorithms: the key generation \( \mathcal{MK} \), the multisignature generation \( \mathcal{MS} \) and the multisignature verification \( \mathcal{MV} \) that are defined as follows:

- \( \mathcal{MK} \) takes a security parameter and returns a public/secret key pair \( (pk_i, sk_i) \) for a signer \( S_i \).
- \( \mathcal{MS} \) takes the set of secret keys \( (sk_i) \) and a message \( M \) and outputs a common a multisignature \( \sigma \).
- \( \mathcal{MV} \) takes the set of public keys \( (pk_i) \) (or only one public key), a multisignature \( \sigma \) and the message \( M \) and outputs 1 (accepts) or 0 (rejects).

The proposed serial multisignature schemes here follow the model of [23] which requires a priori knowledge of an ordered signers set \( \{S_1, \ldots, S_N\} \). The basic idea of our multisignature schemes is that a signer \( S_i \) first generates a signature \( \sigma_i \) on a message \( M \) and broadcasts it to the next signer \( S_{i+1} \) for further processing. After verifying \( \sigma_i \), \( S_{i+1} \) produces a valid signature on the received components. The generation of the multisignature will be complete when the last signer \( S_N \) signs the message.

B. CFS-based serial multisignature

1) \textbf{Description:} Our scheme can be regarded as the extended version of the modified CFS algorithm [10]. In this scheme a signer \( S_i \) makes use of the CFS signature decoding algorithm to generate its signature based on the previous signature produced by the signer \( S_{i-1} \). Before the signing step, all signers first collaborate to produce a public random vector \( r \) in a serial manner which will be signed together with the message \( M \). In order to check the validity of the resulting multisignature, only the public key of the last signer in the queue will be needed. The CFS multisignature scheme is illustrated in Algorithm 4.

2) \textbf{Performance Analysis:} Using an \( (2^n, 2^n - m) \) Goppa code, each public key \( H_i \) is a binary matrix of size \( mt \times 2^n \) bits which takes about 99 Mbytes for \( t = 9 \) and \( m = 22 \), the multisignature generation consists in producing of \( N \) successive CFS signatures of each signer, each of them requires \( t^2 m \)\(! \) binary operations, where \( N \) is the number of signers. Verification requires one matrix-vector multiplication and \( N \) hash computing. A matrix-vector multiplication can be performed in approximately \( t^2 m \) binary operations using the mailman algorithm [21]. The CFS-multisignature is composed of a vector of \( F_2^m \) of weight less than \( t \), \( N \) indexes from \( \{1, \ldots, 2^m\} \) and a vector of \( F_2^{2^m} \). Thus the size of CFS-multisignature is bounded by \( \log_2 (|2^n|) + N \log_2 (|t|) + tm \).

We can easily see that the performance of the proposed multisignature depends mainly on the choice of parameters \( m \) and \( t \). If we want to get a reasonable signature cost, we will need a \( t \) not greater than 10, for example \( (m, t) = (22, 9) \) that give a security level of \( 2^{81.7} \) according [12]. But if we want to minimize the public key size as well as the signature length, we take \( (m, t) = (15, 12) \) for a security level of \( 2^{81.3} \) [12]. In this case, the signature length amounts to \( 377 + 18.47N \) bits.

3) \textbf{Security Analysis:} Since the modified version of CFS signature scheme is secure in the random oracle model [10], We can prove the security of our scheme. The details of our analysis will appear in a full version of the this paper, but we can give some arguments about the security of our scheme. Our scheme satisfies the non-repudiation and the non-forgeability. Indeed, when \( N = 1 \), our signature scheme degenerates into mCFS signature scheme which satisfies these two properties. If \( N > 1 \), an attacker who does not belong to the signer set, can not forge the multisignature because
the signer set has been already known in advance and if he generates a couple \((s_A, t_A)\) as own signature, this signature will be invalid after checking it by the next signer.

### C. KKS-based serial multisignature scheme

1) **Description:** Our scheme extends the regular KKS-signature into a multi-signer one. In this scheme each signer applies the KKS-signature algorithm to produce his own signature on received components before he forwards it to the next signer for consecutive handling. Before the beginning of signing process, all signers first collaborate to create a public a vector \(r \in \mathbb{F}_q^{n-k}\) in a serial way which will be concatenated with the original message \(M\). During the signing step, a signer \(S_i\) has first to verify the previous signature \(\sigma_{i-1}\) generated by previous signer and then to produce his own signature \(\sigma_i\) as follows: The signer hashes the bitwise addition of \(M\) linked with \(r\) and the preceding KKS-signature \(\sigma_{i-1}\) generated by \(S_{i-1}\) and then he applies the KKS-algorithm on the result. After that, he replaces the resulting signature by substituting the quantity \((\sigma_{i-1}, G_i)\) from it. The last operation is designed in order that the new signature can be verified by the succeeding signer in the same manner as the standard KKS-signature. The KKS-multisignature \(\sigma_{kks}\) consists finally of the KKS-signature produced by the last signer in the queue (say \(s_N\)) and the vector \(r\) constructed hence. To test whether this multisignature is valid, the verifier has to apply the KKS-verification algorithm. The Algorithm 5 explains in more detail our scheme.

### Algorithm 5 KKS-based multisignature

**Key Generation:** Given a hash-function of range \(\{0, 1\}^{n-k}\), each signer \(S_i\) has to:
- select \(n, k, t_1\) and \(t_2\) as security parameter.
- select a random matrix \(H_i\) as a parity check matrix of a random \((n, k)\) code \(C_i\).
- Choose secretly and randomly:
  - a generator matrix \(G_i\) of a linear code \(U_i\) of length \(n' \leq n\) and dimension \(k\) s.t. \(t_1 \leq w(u) \leq t_2\) for all \(u \in U_i\).
  - a subset \(J_i\) of \(\{1, \ldots, n\}\) of cardinality \(n'\).
- Build the sub matrix \(H_i(J_i)\) of \(H_i\) consisting of the columns \(h_j\) where \(j \in J_i\).
- Define the matrix \(F_i = H_i(J_i)G_i^T\).
- The public key: \((F_i, H_i, t_1, t_2)\)
- The private key: \((J_i, G_i)\).

**Signature:**

1- Generation of a random vector \(r \in \mathbb{F}_q^{n-k}\)
   - \(S_i\) selects a random vector \(r_i \in \mathbb{F}_q^{n-k}\).
   - For \(i := 2\) to \(N\) do
     - \(S_i\) broadcasts \(r_{i-1}\) to \(S_{i-1}\).
     - \(S_i\) selects a random vector \(r_i \in \mathbb{F}_q^{n-k}\) and assigns \((r_{i-1} + r_i)\) to \(r_i\).
   - \(S_N\) broadcasts \(r_{N-1}\) to \(S_{N-1}\).
   - \(S_N\) selects a random vector \(r_N \in \mathbb{F}_q^{n-k}\).

2- **Multisignature Generation**
   - \(S_i\) calculates \(\sigma_i = h(M|r) \cdot G_i\) and produces \(\sigma_i\) s.t.
     - \(\sigma_{i,j} = \begin{cases} \sigma_{i,j}^* & \text{if } j \in J_i, \\ 0 & \text{if } j \notin J_i, \end{cases}\)
   - For \(i := 2\) to \(N\) do
     - \(S_i\) sends \(r_{i-1}\) to \(S_{i-1}\).
     - \(S_i\) checks the validity of \(\sigma_{i-1}\) by
       - \(t_1 \leq w(\sigma_{i-1}) \leq t_2\) and
       - \(F_{i-1} \cdot (h(M|r))^T = H_{i-1} \cdot \sigma_{i-1}^T\).
     - \(S_i\) calculates \(\sigma_i = (h(M|r) + \sigma_{i-1}) \cdot G_i\) and produces \(\sigma_i\) s.t.
       - \(\sigma_{i,j} = \begin{cases} \sigma_{i,j}^* & \text{if } j \in J_i, \\ 0 & \text{if } j \notin J_i, \end{cases}\)

**Verification:** Given a tuple \((z, r)\), the multisignature is valid if:
   - \(t_1 \leq w(z) \leq t_2\)
   - \(F_N \cdot (h(M|r))^T = H_N \cdot z^T\).

### Algorithm 4 CFS-based multisignature

**Key Generation:** Each signer \(S_i\) has to:
- generate his public/private key as in the CFS algorithm i.e.
  \(H_i = Q_iH_iP_i\) (\(Q_i, H_i, P_i, \gamma_i\))

**Signature:**

1- Generation of a random vector \(r \in \mathbb{F}_q^{n-k}\)
   - \(S_i\) selects randomly \(k_i \in \mathbb{F}_q^n\) of weight up to \(t\) and computes \(r_i = H_i \cdot k_i^T\).
   - From \(i := 2\) to \(N\) do
     - \(S_{i-1}\) broadcasts \(r_{i-1}\) to \(S_i\).
     - \(S_i\) selects randomly \(k_i \in \mathbb{F}_q^n\) of weight up to \(t\) and computes \(r_i = r_{i-1} + H_i \cdot k_i^T\).
   - Set \(r = r_N\).

2- **Multisignature Generation**
   - \(S_i\) computes a \(n\)-bit vector \(s_i\) of weight up to \(t\) and an index \(i_1\) s.t.
     \(H_i \cdot s_i^T = h((M + r)|i_1)\)
   - For \(i := 2\) to \(N\) do
     - \(S_{i-1}\) sends \((s_{i-1}, i_{j-1})\) to \(S_j\).
     - \(S_j\) checks the validity of \(s_j\) by
       - \(H_j \cdot s_j^T = h((M + r)|i_{j-1})\) and \(w(s_j) \leq t\)
     - \(S_j\) computes a \(n\)-bit vector \(s_j\) of weight up to \(t\) and an index \(i_j\) s.t.
       - \(H_j \cdot s_j^T = h((H_{i-1} \cdot s_{i-1}^T + h(M + r))|i_j)\)
   - Set \(s = s_N\).
   - \(\sigma_{i,j} = (s, i_1, \ldots, i_N, r)\) is the multisignature.

**Verification:** Given a tuple \((s, t_1, t_2)\)
   - Check that \(w(s) \leq t\)
   - Compute \(x = H_N \cdot s^T\).
   - Compute iteratively the sequence \((z_i)_{i=1,\ldots,N}\)
     defined by:
     - \(z_1 = h((M + r)|i_1)\)
     - For \(j := 2\) to \(N\) : \(z_j = h((z_{j-1} + h(M + r))|i_j)\).
   - The multisignature \(\sigma\) is valid if \(x\) and \(z_N\) are equals.
of this code is huge for any practical applications, the KKS-1 is still impracticable. Therefore, [17] replaced the equidistant code by another code whose non-zero codewords have a weight between two different values $t_1$ and $t_2$ and proposed two improvements of KKS-1, KKS-2 and KKS-3. The KKS-2 is based on the dual of a BCH code while the KKS-3 is fully random construction and uses a random linear code. In this section we restrict our analysis to the KKS-3 signature scheme.

In KKS-3, each signer choose a random $k \times n'$ generator matrix $G_i$ given in the systematic form $[I_k|B_i]$. The public key is composed of $F_i$ and $H_i = [I_r|D_i]$ where $D_i$ is a random $r \times (n - k)$ binary matrix. The secret key consists of the set $J_i$ and the matrix $B_i$. Thus, to store each public key, we need in total $\tau(n + r) \times k$ bits. For each secret key, we have to store $r b_2 \frac{1}{2} + (n' - k) \times \log_2(3)$ bits, where $b_2(x) = -x \log_2((1-x) \log_2(1-x))$. The multisignature consists of a vector of length $n$ and a weight up to $t_2$ and a random vector of $\{0,1\}^{n-k}$. Thus, the total length of our multisignature is about $[t_2 b_2(\frac{1}{2}) \times n] + (n - k) \times \log_2(3)$ bits which does not depends on the number of signers. The essential part in generating the multisignature is the second step in which each user has to produce his own KKS-like signature while the generating the multisignature is the first step in which each user has to produce its own KKS-like signature while the first phase for producing a common random vector can be performed off-line. Thus, to generate a multisignature, each signer first have to verify the preceding signature and then to produce his KKS-signature. Therefore, the overall cost of our multisignature is approximated to $N\tau k' + (N - 1)\tau (n + k)$ binary operations. After receiving a multisignature, any user can check its validity by comparing the results of two vector-matrix multiplications that require about $\tau (n + k)$ binary operations.

3) Security Analysis: In [17] the authors claimed that their constructions are secure as Niederreiter scheme if the public parameters do not provide any information. Unfortunately [8] showed that a generated KKS-signature discloses a lot of information about the secret set $J$ leading to find the secret matrix $G$ with high probability. Furthermore, [8] proved that just a few intercepted signatures damages the KKS-system. For instance, an attacker needs at most 20 signatures to break the original KKS-3 scheme with an approximate amount of $2^{77}$ binary operations. Regarding the security of our multisignature, since our construction is based on the KKS-signature, we can assume that our scheme is a few times. Following [8], we propose the same parameters for our multisignature scheme to achieve a security level more than $2^{80}$ binary operations. However, the first system suffers from slow signature cost and large key sizes while the second scheme is only few times and very fast but also requires big key sizes. Recently, two works are published for reducing the key sizes (see [3], [24]) and further progress on this topic should increase significantly the performance of our schemes.

REFERENCES


International Scholarly and Scientific Research & Innovation 4(3) 2010 410 scholar.waset.org/1307-6892/5409


