Eliminating Decryption Failures from the Simple Matrix Encryption Scheme

Albrecht Petzoldt[1], Jintai Ding[2], and Lih-Chung Wang[3]

[1] Kyushu University, Fukuoka, Japan
petzoldt@imi.kyushu-u.ac.jp
[2] University of Cincinnati, Ohio, USA
jintai.ding@gmail.com
[3] National Dong Hwa University, Taiwan
lihchungwang@gmail.com

Abstract. The SimpleMatrix encryption scheme as proposed by Tao et al. [16] is one of the very few existing approaches to create a secure and efficient encryption scheme on the basis of multivariate polynomials. However, in its basic version, decryption failures occur with non-negligible probability. Although this problem has been addressed in several papers [5,17], a general solution to it is still missing.
In this paper we propose an improved version of the SimpleMatrix scheme, which eliminates decryption failures completely and therefore solves the biggest problem of the SimpleMatrix encryption scheme. Additionally, we propose a second version of the scheme, which reduces the blow-up factor between plain and ciphertext size to a value arbitrary close to 1.

Keywords: Multivariate Cryptography, SimpleMatrix Encryption Scheme, Decryption Failures, Blow-up Factor

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 problems such as factoring large integers and solving discrete logarithms. The best known schemes in this area are RSA [14], DSA [11] and ECC. However, schemes like these will become insecure as soon as large enough quantum computers arrive. The reason for this is Shor's algorithm [15], which solves number theoretic problems like integer factorization and discrete logarithms in polynomial time on a quantum computer. Therefore, one needs alternatives to those classical public key schemes which are based on mathematical problems not affected by quantum computer attacks.
Besides lattice, code and hash based cryptosystems, multivariate cryptography is one of the main candidates for this [1]. Multivariate schemes are very fast and require only modest computational resources, which makes them attractive for

the use on low cost devices like smart cards and RFID chips [2,3]. However, while there exist many practical multivariate signature schemes [7,10,13], the number of efficient and secure multivariate encryption schemes is somewhat limited.
At PQCrypto 2013, Tao et al. proposed a new MPKC for encryption called the SimpleMatrix (or ABC) encryption scheme [16], which is quite efficient and resists all known attacks against multivariate cryptosystems. However, decryption failures occur with non-negligible probability. The problem of decryption failures occurring in the SimpleMatrix scheme has been addressed in several papers, including [5,17]. However, while these approaches could reduce the probability of decryption failures occurring, a general solution to the problem is still missing. Moreover, the strategy presented in [17] increases key and ciphertext sizes of the scheme.
In this paper, we propose an improved version of the SimpleMatrix scheme, which eliminates decryption failures from the scheme completely, without increasing the key sizes or the blow-up factor between plain and ciphertext size. Therewith we solve the biggest problem of the SimpleMatrix encryption scheme, which prevented the scheme from being used in practice. Furthermore, in contrast to previous versions of the SimpleMatrix scheme, our scheme can be used over small fields, which reduces the key sizes significantly.
We achieve our results by choosing a linear map $\mathcal{T}$ of a special form. By doing so, we enable the sender of a message to check a priori if the corresponding ciphertext will be decryptable. Therefore it is ensured that only decryptable ciphertexts are sent to the receiver.
Additionally to this, we propose a second improved version of the SimpleMatrix encryption scheme, which reduces the blow-up factor between plain and ciphertext size from 2 to a value arbitrary close to 1.
The rest of this paper is organized as follows. In Section 2 we give an overview of multivariate cryptography and introduce the basic ABC encryption scheme as proposed in [16]. In Section 3 we present our technique to eliminate decryption failures from the SimpleMatrix scheme, while Section 4 reduces the blow-up factor of the scheme. We discuss the differences between the improved schemes and the basic scheme and analyze the security of our constructions. Section 5 gives practical parameter sets for our schemes and discusses their efficiency, while Section 6 considers the question, if and how our two approaches can be combined. Finally, Section 7 concludes the paper.

2 The basic ABC Encryption Scheme

In this section we introduce the basic ABC encryption scheme as proposed by Tao et al. in [16]. Before we come to the description of the scheme itself, we start with an overview of the main concepts of multivariate cryptography.

2.1 Multivariate Cryptography

The basic objects of multivariate cryptography are systems of multivariate quadratic polynomials. The security of multivariate schemes is therefore based on the

MQ Problem: Given m multivariate quadratic polynomials $p^{(1)}(\mathbf{x}), \ldots, p^{(m)}(\mathbf{x})$ in the n variables $x_1, \ldots, x_n$, 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 $m \approx n$) has been proven to be NP-hard even for quadratic polynomials over the field GF(2) [9].

To build a public key cryptosystem on the basis of 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.

Remark: Due to the upper construction, the security of multivariate public key schemes is not solely based on the MQ problem. Instead of this, an attacker can also try to find the decomposition of the public key (EIP-Problem).

In this paper we concentrate on multivariate encryption schemes. The standard encryption/decryption process works as shown in Figure 1.

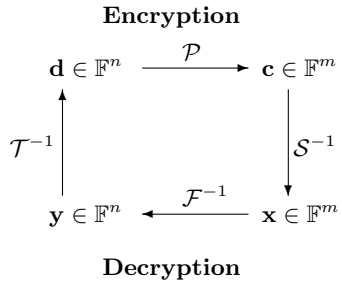

Fig. 1. General workflow of multivariate encryption schemes

Encryption: To encrypt a message $\mathbf{d} \in \mathbb{F}^n$, one simply computes $\mathbf{c} = \mathcal{P}(\mathbf{d})$. The ciphertext of the message $\mathbf{d}$ is $\mathbf{c} \in \mathbb{F}^m$.

Decryption: To decrypt the ciphertext $\mathbf{c} \in \mathbb{F}^m$, one computes recursively $\mathbf{x} = \mathcal{S}^{-1}(\mathbf{c})$, $\mathbf{y} = \mathcal{F}^{-1}(\mathbf{x})$ and $\mathbf{d} = \mathcal{T}^{-1}(\mathbf{y})$. $\mathbf{d} \in \mathbb{F}^n$ is the plaintext corresponding to the ciphertext $\mathbf{c}$.

Since, for multivariate encryption schemes, we have $m \geq n$, the pre-image of the vector $\mathbf{x}$ under the central map $\mathcal{F}$ and therefore the decrypted plaintext is

unique.

A good overview of existing multivariate schemes can be found in [4].

2.2 The ABC Encryption Scheme

The basic SimpleMatrix encryption scheme as proposed by Tao et al. in [16] can be described as follows.

Key Generation: Let $\mathbb{F}$ be a finite field. For a parameter $s \in \mathbb{N}$ we set $n = s^2$ and $m = 2 \cdot n$ and define three matrices $A,\ B$ and C of the form

$$A = \begin{pmatrix} x_1 & \dots & x_s \\ \vdots & & \vdots \\ x_{(s-1)\cdot s+1} & \dots & x_n \end{pmatrix},\ B = \begin{pmatrix} b_1 & \dots & b_s \\ \vdots & & \vdots \\ b_{(s-1)\cdot s+1} & \dots & b_n \end{pmatrix},\ C = \begin{pmatrix} c_1 & \dots & c_s \\ \vdots & & \vdots \\ c_{(s-1)\cdot s+1} & \dots & c_n \end{pmatrix}.$$

Here, $x_1, \dots, x_n$ are the linear monomials of the multivariate polynomial ring $\mathbb{F}[x_1, \dots, x_n]$, whereas $b_1, \dots, b_n$ and $c_1, \dots, c_n$ are randomly chosen linear combinations of $x_1, \dots, x_n$.
One computes $E_1 = A \cdot B$ and $E_2 = A \cdot C$. The central map $\mathcal{F}$ of the scheme consists of the m components of E_1 and E_2.

The *public key* of the scheme is the composed map $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T} : \mathbb{F}^n \to \mathbb{F}^m$ with two randomly chosen invertible linear maps $\mathcal{S} : \mathbb{F}^m \to \mathbb{F}^m$ and $\mathcal{T} : \mathbb{F}^n \to \mathbb{F}^n$, the *private key* consists of the matrices B and C and the linear maps $\mathcal{S}$ and $\mathcal{T}$.

Encryption: To encrypt a message $\mathbf{d} \in \mathbb{F}^n$, one simply computes $\mathbf{c} = \mathcal{P}(\mathbf{d}) \in \mathbb{F}^m$.

Decryption: To decrypt a ciphertext $\mathbf{c} \in \mathbb{F}^m$, one has to perform the following three steps.

1. Compute $\mathbf{x} = \mathcal{S}^{-1}(\mathbf{c})$. The elements of the vector $\mathbf{x} \in \mathbb{F}^m$ are written into matrices $\bar{E}_1$ and $\bar{E}_2$ as follows.
$$\bar{E}_1 = \begin{pmatrix} x_1 & \dots & x_s \\ \vdots & & \vdots \\ x_{(s-1)\cdot s+1} & \dots & x_n \end{pmatrix},\ \bar{E}_2 = \begin{pmatrix} x_{n+1} & \dots & x_{n+s} \\ \vdots & & \vdots \\ x_{n+(s-1)\cdot s+1} & \dots & x_m \end{pmatrix}.$$
2. In the second step one has to find a vector $\mathbf{y} = (y_1, \dots, y_n)$ such that $\mathcal{F}(\mathbf{y}) = \mathbf{x}$. To do this, one has to distinguish four cases:
   - If $\bar{E}_1$ is invertible, one considers the equation $B \cdot \bar{E}_1^{-1} \cdot \bar{E}_2 - C = 0$. Therefore one gets n linear equations in the n variables $y_1, \dots, y_n$.
   - If $\bar{E}_1$ is not invertible, but $\bar{E}_2$ is invertible, one considers the equation $C \cdot \bar{E}_2^{-1} \cdot \bar{E}_1 - B = 0$. One gets n linear equations in the n variables.
   - If none of $\bar{E}_1$ and $\bar{E}_2$ is invertible, but $\bar{A} = A(\mathbf{y})$ is invertible, one considers the relations $\bar{A}^{-1} \cdot \bar{E}_1 - B = 0$ and $\bar{A}^{-1} \cdot \bar{E}_2 - C = 0$. One interprets the elements of $\bar{A}^{-1}$ as new variables $w_1, \dots, w_n$ and therefore gets m linear equations in the m variables $w_1, \dots, w_n, y_1, \dots, y_n$.

– If none of $\bar{E}_1$, $\bar{E}_2$ and $\bar{A}$ is invertible, there occurs a decryption failure.

3. Finally, one computes the plaintext by $\mathbf{d} = \mathcal{T}^{-1}(y_1, \ldots, y_n)$.

The probability of a decryption failure occurring in the second step is about $\frac{1}{q}$, where q is the cardinality of the underlying field $\mathbb{F}$.

It might happen that the linear systems in the second step of the decryption process have multiple solutions $\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(\ell)}$. In this case one has to perform the third step for each of these solutions to get a set of possible plaintexts $\mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(\ell)}$. By encrypting these plaintexts one can test which of them corresponds to the given ciphertext $\mathbf{c}$.

3 Eliminating the Decryption Failures

In this section we present our technique to eliminate the decryption failures from the SimpleMatrix encryption scheme. These decryption failures occur in all so far existing versions of the SimpleMatrix encryption scheme [16,5,17] and therefore prevent the scheme from being used in practice. Our technique enables the sender to check a priori, if the ciphertext corresponding to his message will be decryptable. If not, he can change his message slightly until he gets a message whose corresponding ciphertext is deryptable. By doing so, he can make sure to send only decryptable ciphertexts to the receiver.
Our scheme, denoted as the TensorSimpleMatrix encryption scheme, can be described as follows.

Let $\mathbb{F} = \mathbb{F}_q$ be a finite field with q elements, $s \in \mathbb{N}$ be an integer, $n = s^2$ and $m = 2 \cdot n$.

Key Generation: The $s \times s$ matrices A, B, C, E_1 and E_2 and the maps $\mathcal{F} : \mathbb{F}^n \to \mathbb{F}^m$ and $\mathcal{S} : \mathbb{F}^m \to \mathbb{F}^m$ are defined as in the case of the basic ABC scheme (see Subsection 2.2). Additionally, we choose two invertible $s \times s$ matrices T_1 and T_2 and compute $T = T_1 \otimes T_2$, where $\otimes$ is the standard tensor product. Note that with T_1 and T_2 being invertible the $n \times n$ matrix T is invertible, too. The linear map $\mathcal{T}$ used in our scheme is represented by this matrix T.

Lemma 1. *Let $\mathbf{d} = (d_1, \ldots, d_n) \in \mathbb{F}^n$ be a vector such that $A(\mathbf{d})$ is an invertible $s \times s$ matrix. Then the $s \times s$ matrix $A(\mathcal{T}(\mathbf{d}))$ will be invertible, too.*

Proof. We get $A(\mathcal{T}(\mathbf{d})) = T_1 \cdot A(\mathbf{d}) \cdot T_2^T$. Since all three matrices T_1, $A(\mathbf{d})$ and T_2 are invertible, the same holds for $A(\mathcal{T}(\mathbf{d}))$. □

Remark: Lemma 1 states that, for a plaintext vector $\mathbf{d}$ whose associated matrix $A(\mathbf{d})$ is invertible, the matrix $\bar{A} = A(\mathcal{T}(\mathbf{d}))$ considered in the second step of the decryption process is invertible, too. By encrypting only plaintexts $\mathbf{d} \in \mathbb{F}^n$ for which the matrix $A(\mathbf{d})$ is invertible (this can be ensured easily during the encryption process) we can therefore ensure that the decryption process of the ABC scheme runs correctly.

The *public key* is the composed map $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$, the *private key* consists of the matrices T_1, T_2, B, C and the linear map $\mathcal{S}$.

Encryption: To encrypt a message $\mathbf{d} = (d_1, d_2, \dots)$ of $\mathbb{F}$-elements, we decompose the message into message blocks $D_1, D_2, \dots$ of length n. We define a matrix

$$A(D_1) = \begin{pmatrix} d_1 & \dots & d_s \\ \vdots & & \vdots \\ d_{n-s+1} & \dots & d_n \end{pmatrix} \tag{1}$$

and test, if the matrix $\hat{A} = A(D_1)$ is invertible. If this is the case, we proceed to the next step, if not, we insert a predefined element pre at a random position into the message block D_1 [4]. By doing so, we get the block ${D_1}' = (d_1, \dots, d_j, \text{pre}, d_{j+1}, \dots, d_{n-1})$ [5]. After that we test if the resulting matrix $A(D_1')$ is invertible. We repeat this step, until we find a block $D_1' \in \mathbb{F}^n$ whose associated matrix $A(D_1')$ is invertible.
After having found such a block D_1' with $A(D_1')$ being invertible we compute the ciphertext by $\mathcal{P}(D_1')$.
We continue this process with the next message block D_2 (if we inserted i elements pre into the first message block D_1, D_2 has the form $D_2 = (d_{n-i+1}, \dots, d_{2n-i})$).

Remark: By Lemma 1 we know that, with $A(D_1')$, the matrix $\bar{A} = A(\mathcal{T}(D_1')$ appearing in the second step of the decryption process will be invertible, too. Therefore we know that the decryption process will run correctly.

Decryption: The decryption process works as shown in Subsection 2.2. However, in the new setting, the matrix $\bar{A}$ used in step 2 of the decryption process will always be invertible and therefore decryption failures no longer occur.

After having found the encrypted plaintext $\mathbf{d}$, we remove all the appearances of the character pre to get the original message.

3.1 Security Analysis

Direct attacks The most straightforward way to attack a multivariate scheme such as SimpleMatrix is by trying to solve the public equation $\mathcal{P}(\mathbf{d}) = \mathbf{c}$ directly (message recovery attack). For this an attacker can use an algorithm like XL [6] or a Gröbner basis method such as F_4 or F_5 [8].
To analyze the effectiveness of direct attacks against our scheme, we performed a number of experiments with MAGMA, which contains an efficient implementation of Faugères F_4 algorithm. The experiments were performed on a server with 24 AMD Opteron processors (2.5 GHz) and 128 GB RAM. Table 1 shows the results.

[4] In our implementation (over the field $\mathbb{F}$=GF(256)) we use pre = $0x10$ which corresponds to the ASCII character ⁻.

[5] The element d_n is moved to the next message block.

		GF(2^4)			GF(2^8)		
	s	3	4	5	3	4	5
	m,n	18,9	32, 16	50, 25	18, 9	32, 16	50, 25
TensorSimpleMatrix	d_{reg}	4	5	6	4	5	6
	time (s)	0.3	2.3	6,759	0.3	2.4	13,488
	memory (MB)	12.5	37.2	7,899	12.5	37.0	8,825
for comparison:	d_{reg}	4	5	6	4	5	6
standard	time (s)	0.3	2.4	6,759	0.3	2.4	13,546
SimpleMatrix [16]	memory (MB)	12.5	37.3	7,932	12.5	37.2	8,832

Table 1. Results of our experiments with direct attacks on the TensorSimpleMatrix scheme

As the table shows, there is no major difference between the basic SimpleMatrix scheme and our improved scheme with respect to the behavior of direct attacks.

Special Attacks Compared to the case of the standard SimpleMatrix encryption scheme (see Section 2.2), the TensorSimpleMatrix encryption scheme uses a specially designed linear map $\mathcal{T}$. In particular, the $n \times n$ matrix T representing the linear map $\mathcal{T}$ is given by

$$T = T_1 \otimes T_2 \tag{2}$$

with two invertible $s \times s$ matrices T_1 and T_2. In this section we analyze whether this can be used by an attacker against the scheme.
The special structure of the matrix T reduces the number of possible choices of this matrix significantly. However, even for the smallest of the parameter sets proposed in Section 5 ($\mathbb{F} = \text{GF}(2^4), s = 8$), we still have

$$\left(\sum_{i=0}^{s-1}(q^s - q^i)\right)^2 \approx 2^{511.8} \tag{3}$$

possibilities for the choice of T.

To answer the question, if the special structure of the map $\mathcal{T}$ yields any relation between the coefficients of the public key $\mathcal{P}$, we made the following test: For simplicity, we fixed the first affine map $\mathcal{S}$ to be the identity map, i.e. $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$. In this case, the coefficient matrix MP of the public key can be computed by

$$MP = MF \cdot A, \tag{4}$$

where MF is the coefficient matrix of the central map and $A = (\alpha_{ij}^{rs})$ is an $\frac{n\cdot(n+1)}{2} \times \frac{n\cdot(n+1)}{2}$ matrix, whose elements are given by [12]

$$\alpha_{ij}^{rs} = \begin{cases} t_{ri} \cdot t_{si} & i = j \\ t_{ri} \cdot t_{sj} + t_{rj} \cdot t_{si} & i \neq j \end{cases} \tag{5}$$

Note that, in the case of $T = T_1 \otimes T_2$, the elements of the matrix A are quartic polynomials in the elements of T_1 and T_2.

We compared the distribution of the matrices A in the case of TensorSimpleMatrix scheme with that of the standard scheme and found that there is no significant difference. Therefore we believe that it will be difficult for an attacker to distinguish between public keys of the TensorSimpleMatrix and the standard ABC scheme.

4 Reducing the Blow-up Factor between Plain and Ciphertext

For all the previously proposed variants of the SimpleMatrix encryption scheme the ciphertext is at least twice as large as the corresponding plaintext. In this section we present a simple technique to reduce this blow-up factor to a value arbitrary close to 1. We achieve this by using more than three matrices in the key generation process of the scheme. Our scheme, denoted as the ABCD encryption scheme, can be described as follows:

Let $\mathbb{F} = \mathbb{F}_q$ be a finite field with q elements and s and k be integers. We set $m = k \cdot s^2$ and $n = m - s^2$.

Key Generation: We choose an $s \times s$ matrix A and k $s \times s$ matrices $B^{(1)}, \ldots, B^{(k)}$ of the form

$$A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1s} \\ a_{21} & a_{22} & \ldots & a_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ a_{s1} & a_{s2} & \ldots & a_{ss} \end{pmatrix}, \qquad B^{(i)} = \begin{pmatrix} b_{11}^{(i)} & b_{12}^{(i)} & \ldots & b_{1s}^{(i)} \\ b_{21}^{(i)} & b_{22}^{(i)} & \ldots & b_{2s}^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ b_{s1}^{(i)} & b_{s2}^{(i)} & \ldots & b_{ss}^{(i)} \end{pmatrix} \ (i = 1, \ldots, k)$$

containing randomly chosen linear combinations of the monomials $x_1, \ldots, x_n$ of the multivariate polynomial ring $\mathbb{F}[x_1, \ldots, x_n]$ [6].

Similar to the case of the original SimpleMatrix scheme (see Subsection 2.2) we define $E^{(i)} = A \cdot B^{(i)}$ $(i = 1, \ldots, k)$. The central map $\mathcal{F}$ of our scheme consists of the $m = k \cdot s^2$ components of the matrices $E^{(i)}$ $(i = 1, \ldots, k)$. We note that each of these components is a quadratic polynomial in $\mathbb{F}[x_1, \ldots, x_n]$ whose associated quadratic form has a rank close or equal to $2s$.

The *public key* $\mathcal{P}$ of the scheme is defined as

$$\mathcal{P} = \mathcal{S} \circ \mathcal{F}$$

[6] In the case of $k = 3$, we denote the matrices $B^{(1)}, B^{(2)}$ and $B^{(3)}$ by B, C and D. In analogy to the ABC scheme we call our scheme therefore the ABCD encryption scheme.

with an invertible linear map $\mathcal{S} : \mathbb{F}^m \to \mathbb{F}^m$. [7]

The *private key* consists of the linear map $\mathcal{S}$ and the matrices $B^{(i)}$ $(i = 1, \ldots, k)$.

Encryption: To encrypt a message $\mathbf{d} = (d_1, d_2, \ldots, d_n) \in \mathbb{F}^n$, one simply computes $\mathbf{c} = \mathcal{P}(\mathbf{d})$. $\mathbf{c} \in \mathbb{F}^m$ is the ciphertext corresponding to the message $\mathbf{d}$.

Decryption: To decrypt the ciphertext $\mathbf{c} = (c_1, c_2, \ldots, c_m) \in \mathbb{F}^m$, one has to perform the following two steps:

1. Compute $\mathbf{x} = (x_1, x_2, \ldots, x_m) = \mathcal{S}^{-1}(\mathbf{c})$ and set
$$\bar{E}^{(i)} = \begin{pmatrix} x_{(i-1)s^2+1} & x_{(i-1)s^2+2} & \cdots & x_{(i-1)s^2+s} \\ x_{(i-1)s^2+s+1} & x_{(i-1)s^2+s+2} & \cdots & x_{(i-1)s^2+2s} \\ \vdots & \vdots & \ddots & \vdots \\ x_{i\cdot s^2-s+1} & x_{i\cdot s^2-s+2} & \cdots & x_{i\cdot s^2} \end{pmatrix} \quad (i = 1, \ldots, k).$$
2. Let $\bar{A} = A(\mathbf{y})$, where $\mathbf{y} = (y_1, \ldots, y_n)$ is the (so far unknown) pre-image of $\mathbf{x}$ under the central map $\mathcal{F}$.
   - If $\bar{A}$ is invertible, we set $W = \bar{A}^{-1}$ and consider the equations $\bar{E}^{(i)} = \bar{A} \cdot B^{(i)}$ $(i = 1, \ldots, k)$. From these we obtain $W \cdot \bar{E}^{(i)} = W \cdot \bar{A} \cdot B^{(i)}$ and therefore $W \cdot \bar{E}^{(i)} = B^{(i)}$ $(i = 1, \ldots, k)$. We interpret the elements of W as new variables $w_1, \ldots, w_{s^2}$ and end up with $k \cdot s^2$ linear equations in $k \cdot s^2$ unknowns (the s^2 elements of W and the $n = (k-1) \cdot s^2$ plaintext variables.). We can solve the system by Gaussian Elimination.

   - In the case of the matrix $\bar{A}$ being singular, decryption remains an open problem.

   The plaintext corresponding to the ciphertext $\mathbf{c}$ is given by $\mathbf{d} = (y_1, \ldots, y_n) \in \mathbb{F}^n$.

   Remark: As in the case of the standard SimpleMatrix scheme, it may happen that the linear system in step 2 of the decryption process has multiple solutions $\mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(\ell)}$. In this case one has to encrypt the possible plaintexts to check which of them corresponds to the given ciphertext.

   Remark: In step 2 of the decryption process we obtain a system of $m = k \cdot s^2$ linear equations in m variables. Of these variables, s^2 are the unknown elements of the matrix $W = \bar{A}^{-1}$ and $n = (k-1) \cdot s^2$ are the plaintext variables, which can be recovered by Gaussian Elimination. We therefore get a blow-up factor between plain and ciphertext size of
$$\text{blow-up-factor} = \frac{m}{n} = \frac{k \cdot s^2}{(k-1) \cdot s^2} = 1 + \frac{1}{k-1} \tag{6}$$
   By using large values of k, we therefore can reduce the blow-up factor between plain and ciphertext size to values arbitrary close to 1.

[7] For the security of our scheme, we do not need the second linear map $\mathcal{T}$. So we drop it.

4.1 Security

Direct attacks To estimate the behavior of direct attacks against the ABCD encryption scheme, we performed a number of experiments with MAGMA which contains an efficient implementation of the F_4 algorithm. The results are shown in Table 2. Note that for $k = 2$ our scheme is just the standard SimpleMatrix scheme of [16].

		GF(2^8)			GF(2^{16})		
	s	3	4	5	3	4	5
$k = 2$	m,n	18,9	32,16	50,25	18,9	32,16	50,25
	d_{reg}	4	5	6	4	5	6
	time(s)	0.3	2.3	13,546	0.3	12.6	191,114
	memory (MB)	12.5	37.2	8,824	14.1	51.0	16,886
$k = 3$	m,n	27,18	48,32	75,50	27,18	48,32	75,50
	d_{reg}	4	5	5 [1]	4	5	5[1]
	time (s)	1.7	22,956	-	4.9	120,736	-
	memory (MB)	34.3	29,247	ooM [2]	39.0	33,735	ooM
$k = 4$	m,n	36,27	64,48	100,75	36,27	64,48	100, 75
	d_{reg}	4	5	5 [1]	4	5	5 [1]
	time(s)	26.4	-	-	895.6	-	-
	memory (MB)	350.7	ooM	ooM	1,537	ooM	ooM

[1] After having reached degree 5, we ran out of memory
[2] out of memory

Table 2. Results of our experiments with direct attacks against the ABCD scheme

For $k > 2$ the public systems of the ABCD scheme are much larger (and less overdetermined) than the public systems of the original SimpleMatrix scheme. One therefore could expect that they are significantly harder to solve. But, as our experiments show, increasing k has no influence on the degree of regularity of the F_4 algorithm. However, time and memory requirements increase drastically with larger k. We therefore feel secure to decrease the parameter s while increasing k. For our proposed parameter sets (see Section 5) we chose s and k in such a way that $s + k = 10$.

Other attacks Regarding further security analysis (e.g. Rank attacks, (Higher Order) Linearization Attacks) we just note that the internal structure of the public polynomials is exactly equal to the case of the standard SimpleMatrix scheme. In particular, the components of the central map $\mathcal{F}$ are quadratic polynomials of rank close to $2s$. For a discussion of the security of our scheme against the above mentioned attacks we therefore refer to [16].

5 Parameters and Efficiency

In this section we propose concrete parameter sets for our improved variants of the SimpleMatrix encryption scheme.

For all the previously proposed versions of the SimpleMatrix encryption scheme, the probability of an decryption failure occurring depends on the cardinality q of the underlying field. To reduce the probability of decryption failures occurring, the authors of [16,5,17] suggested therefore to use the scheme over large fields such as GF(2^{16}) or GF(2^{32}). However, this increases the key sizes of the scheme and reduces its performance.
In the case of the TensorSimpleMatrix encryption scheme (see Table 3), we do not have to consider decryption failures. Therefore, we can use this scheme also over small fields, e.g. GF(16) and GF(256). By doing so, we can reduce the public key size of the SimpleMatrix scheme from 1,040 kB [16] to 130 kB. The parameters 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.

Claimed security level	parameters $\mathbb{F}$, s	input size (bit)	output size (bit)	public key size (kB)	private key size (kB)
2^{80}	GF(2^4), 8	256	512	130.0	12.1
	GF(2^8), 8	512	1,024	260.0	24.1
2^{95}	GF(2^4), 9	324	648	262.7	19.3
	GF(2^8), 9	648	1,296	525.4	38.6

Table 3. Proposed parameters and key sizes for the TensorSimpleMatrix encryption scheme

In the case of the ABCD encryption scheme (Table 4), we choose GF(2^{16}) as the underlying field. By doing so, we can bound the probability of decryption failures occurring by 2^{-16}. As above, the parameters in the table are chosen in such a way that the complexity of a direct attack against the scheme is beyond the proposed security level.
Basically, our strategy would allow to reduce the blow-up factor between plain and ciphertext size to a value arbitrary close to 1. However, this leads to very large sizes of the public key. Therefore we do not think that values of $k \geq 5$ are useful in practice.

Claimed security level	parameters $\mathbb{F}$, s, k	input size (bit)	output size (bit)	public key size (kB)	private key size (kB)	blow-up factor	probability of decryption failure
2^{80}	GF(2^{16}), 7, 3	1,568	2,352	1,393	49.2	1.50	2^{-16}
	GF(2^{16}), 6, 4	1,728	2,304	1,655	50.6	1.33	2^{-16}
2^{95}	GF(2^{16}), 8, 3	2,048	3,072	3,096	84.0	1.50	2^{-16}
	GF(2^{16}), 7, 4	2,352	3,136	4,164	93.8	1.33	2^{-16}

Table 4. Proposed parameters and key sizes for the ABCD encryption scheme

The most costly step in the decryption process of our schemes is the solution of the

linear system obtained in step 2 of the decryption process, which consists in solving a linear system of m equations in m variables. In the case of the TensorSimpleMatrix scheme, the complexity of this step is about $8 \cdot s^6$, in the case of the ABCD scheme the complexity is given by $k^3 \cdot s^6$.

6 Combining the two approaches

A natural extension of our work would be to combine the two approaches presented in Sections 3 and 4. By doing so, one would obtain a version of the SimpleMatrix scheme which both reduces the blow-up factor between plain and ciphertext size and solves the problem of decryption failures. However, this can not be done straightforward.

1. To eliminate decryption failures from the scheme, the matrix A must be known to the sender of the message. Therefore, we can not choose the matrix A to contain randomly chosen linear combinations of $x_1, \ldots, x_n$ as it is done in the case of the ABCD scheme. Moreover, we can not choose the matrix A to contain only monomials as in the case of the standard and TensorSimpleMatrix scheme, since then only s^2 of the $(k-1) \cdot s^2$ monomials would appear in the matrix. One possible solution to this problem would be to choose the elements of the matrix A as
$$a_{ij} = \sum_{l=0}^{k-2} x_{l \cdot s^2 + (i-1) \cdot s + j} \tag{7}$$
2. When choosing the matrix A as shown in equation (7), we need a linear transformation $\mathcal{T} : \mathbb{F}^n \to \mathbb{F}^n$ to hide the structure of A in the public key. However, for $n = (k-1) \cdot s^2$ with $k > 2$ it is totally unclear how to find a relation similar to that of Lemma 1. Therefore we do not know how to enable the sender to check whether the ciphertext corresponding to his message will be decryptable.

We therefore leave the problem of combining our two approaches as future work.

7 Conclusion

In this paper we presented a technique to eliminate decryption failures from the SimpleMatrix scheme completely. By choosing a linear map $\mathcal{T}$ of a special form, we enable the sender to check a priori if the ciphertext corresponding to his message will be decryptable. This guarantees that only decryptable ciphertexts are sent to the receiver and therefore solves one of the biggest problems of the SimpleMatrix encryption scheme, which prevented the scheme from being used in practice. Additionally, our new scheme has significantly smaller keys than all so far published versions of the SimpleMatrix scheme.
Furthermore, in this paper, we presented a technique to reduce the blow-up factor between plain and ciphertext to a value arbitrary close to 1.
We hope that our techniques contribute to turn the SimpleMatrix scheme into a really practical multivariate encryption scheme.
Future work includes

- Combination of the two approaches (see Section 6).
- Creation of a cyclic version of the SimpleMatrix encryption scheme. This will enable us to reduce the public key size of the scheme significantly.

References

1. D.J. Bernstein, J. Buchmann, E. Dahmen (eds.): Post Quantum Cryptography. Springer, 2009.
2. 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.
3. 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.
4. J. Ding, J. E. Gower, D. S. Schmidt: Multivariate Public Key Cryptosystems. Springer, 2006.
5. J. Ding, A. Petzoldt, L.C. Wang: The Cubic SimpleMatrix Encryption Scheme. PQCrypto 2014, LNCS vol. 8772, pp. 76 - 87. Springer, 2014.
6. J. Ding, J. Buchmann, M. S. E. Mohamed, W. S. A. E. Mohamed, R.P. Weinmann: MutantXL. SCC 2008.
7. J. Ding, D. S. Schmidt: Rainbow, a new multivariate polynomial signature scheme. ACNS 2005, LNCS vol. 3531, pp. 164-175. Springer, 2005.
8. J.C. Faugère: A new efficient algorithm for computing Gröbner bases (F4). Journal of Pure and Applied Algebra 139, pp. 61-88 (1999).
9. M. R. Garey and D. S. Johnson: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, 1979.
10. A. Kipnis, L. Patarin, L. Goubin: Unbalanced Oil and Vinegar Schemes. EUROCRYPT 1999, LNCS vol. 1592, pp. 206–222 Springer, 1999.
11. D. Kravitz: Digital Signature Algorithm. US patent 5231668 (July 1991).
12. A. Petzoldt, S. Bulygin, J. Buchmann: Linear Recurring Sequences for the UOV Key Generation. PKC'11, LNCS vol. 6571, pp. 335-350, Springer, 2011.
13. J. Patarin, N. Courtois, L. Goubin: QUARTZ, 128-Bit Long Digital Signatures. CTRSA 2001, LNCS vol. 2020, pp. 282-297. Springer, 2001.
14. 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).
15. P. Shor: Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM J. Comput. 26 (5), pp. 1484–1509.
16. C. Tao, A. Diene, S. Tang, J. Ding: Simple Matrix Scheme for Encryption. PQCrypto 2013, LNCS vol. 7932, pp. 231-242. Springer, 2013.
17. C. Tao, H. Xiang, A. Petzoldt, J. Ding: SimpleMatrix - A Multivariate Public Key Cryptosystem (MPKC) for Encryption. Finite Fields and Their Applications 35, pp. 352 - 368. Elsevier, 2015.