Loopholes in Two Public Key Cryptosystems Using the Modular Group

Rainer Steinwandt

Institut für Algorithmen und Kognitive Systeme,
Arbeitsgruppen Computeralgebra & Systemsicherheit, Prof. Dr. Th. Beth,
Universität Karlsruhe, Am Fasanengarten 5, 76 131 Karlsruhe, Germany,
steinwan@ira.uka.de

Abstract. We demonstrate that the public key cryptosystems using the modular group suggested in [4,5] are vulnerable to very simple ciphertext-only attacks. Consequently, in the present form both of these systems cannot be considered as sufficiently secure for cryptographic purposes.

1 Introduction

At PKC'98 A. Yamamura proposed a public key cryptosystem using the so-called modular group (see [4]). In this cryptosystem the ciphertext consists of a 2×2 matrix with entries from $\mathbb{C}[X]$. At ACISP'99 he proposed another public key cryptosystem using the modular group where the ciphertext is represented by a single complex number.

In this paper we show that with the given specifications both of these cryptosystems are vulnerable to ciphertext-only attacks. We give several examples which illustrate that it is often possible to decrypt a ciphertext by means of the public data alone, i. e., without requiring the private key. The essential idea is to exploit the message expansion occurring in both of these cryptosystems.

More detailed, the paper is organized as follows: in the next section we shortly recall the set-up of the cryptosystem suggested in [4] to the extent needed for our attack. In Section 3 we describe our attack and demonstrate its practicability through some examples. Thereafter we show how a modification of our attack can be applied successfully to the public key cryptosystem suggested in [5].

2 A Public Key Cryptosystem Using $\mathbf{SL_2}(\mathbb{Z})$

In this section we shortly recall the ingredients of the public key cryptosystem suggested in [4] to the extent necessary for describing our attack—for a complete description we refer to the original work [4].
As usual, for an integral domain R we denote by

$$\mathrm{SL}_2(R) := \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in R^{2\times 2} : \ ad - bc = 1 \right\}$$

the group of all 2×2 matrices over R with determinant 1; $\mathrm{SL}_2(\mathbb{Z})$ is also known as the *modular group*. To derive a public key we first need generators A, B of $\mathrm{SL}_2(\mathbb{Z})$ subject to the relations

$$A^6 = B^4 = A^3B^{-2} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

As is shown in [4] one can derive such generators by choosing a matrix $N \in \mathrm{SL}_2(\mathbb{Z})$ arbitrarily and setting

$$A := N^{-1} \cdot \begin{pmatrix} 0 & -1 \\ 1 & 1 \end{pmatrix} \cdot N, \quad B := N^{-1} \cdot \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \cdot N.$$

Next, one has to choose two products $V_1, V_2 \in \{A, B\}^*$ subject to certain requirements described in [4]. According to Yamamura's paper a concrete instance satisfying these requirements can be obtained by setting

$$V_1 := (BA)^i, \quad V_2 := (BA^2)^j \quad \text{with } i, j \in \mathbb{N} \text{ positive integers.}$$

Finally, for constructing a public key we also need a non-singular 2×2 matrix M with complex entries, i. e., $M \in \mathrm{GL}_2(\mathbb{C})$ and two 2×2 matrices $F_1(X), F_2(X)$ whose entries are taken from the polynomial ring over the complex numbers $\mathbb{C}[X]$. The matrices $F_1(X), F_2(X)$ are to be chosen in such a way that for some $a \in \mathbb{C}$ we have $F_1(a) = V_1$ and $F_2(a) = V_2$. In other words, evaluating the entries of $F_1(X), F_2(X)$ at $X = a$ yields the matrices V_1, V_2—here $V_1, V_2 \in \{A, B\}^*$ are identified with the 2×2 matrix obtained by "multiplying the letters A, B".

With these conventions the private key is the pair $(M, a) \in \mathrm{GL}_2(\mathbb{C}) \times \mathbb{C}$, and the public key consists of two matrices $W_1(X), W_2(X)$ which are constructed as follows:

$$(W_1(X), W_2(X)) := (M^{-1} \cdot F_1(X) \cdot M, M^{-1} \cdot F_2(X) \cdot M)$$

In order to encrypt the bitstring $b_1 \dots b_n \in \{0, 1\}^*$ with the public key $(W_1(X), W_2(X))$ one has to compute the matrix product

$$C(X) := W_2(X) \cdot \prod_{i=1}^{n} (W_1(X)^{b_i+1} W_2(X)). \tag{1}$$

From the ciphertext $C(X)$ and the private key (M, a) the original bitstring $b_1 \dots b_n$ can be recovered by means of a procedure described in [4].

3 Attacking the System

Denote by $C(X)$ the ciphertext obtained by encrypting the bitstring $b_1 \dots b_n \in \{0, 1\}^*$ according to the rule (1). Then the entries of the matrix

$$D(X) := W_2(X)^{-1} \cdot C(X) = \prod_{i=1}^{n} (W_1(X)^{b_i+1} W_2(X))$$

are contained in $\mathbb{C}[X]$, and by construction also the entries of the matrix $\left(W_1(X)^{b_1+1} \cdot W_2(X)\right)^{-1} \cdot D(X)$ are polynomials with complex coefficients. So if at least one of the entries of

$$\left(W_1(X)^2 \cdot W_2(X)\right)^{-1} \cdot D(X)$$

involves a non-constant denominator then we can conclude $b_1 = 0$. This observation motivates the following naïve procedure which either after n iterations of the while-loop yields the correct plaintext or does not terminate:

Procedure 1

In: *Public key* $(W_1(X), W_2(X))$
Ciphertext $C(X) = W_2(X) \cdot \prod_{i=1}^{n}(W_1(X)^{b_i+1}W_2(X))$
Out: $\perp$ *or the plaintext* $b_1 \dots b_n$

```
begin
  $D(X) \leftarrow W_2(X)^{-1} \cdot C(X)$                # remove superfluous factor
  $l \leftarrow 1$                    # number of plaintext bit to be processed next
  while $D(X)$ is not the identity do              # decryption incomplete
    $D'(X) \leftarrow \left(W_1(X)^2 \cdot W_2(X)\right)^{-1} \cdot D(X)$        # Should left-most bit
                                                             # be set?
    if $D'(X)$ contains a non-polynomial entry
      then $b_l \leftarrow 0$              # no $\rightarrow$ strip off $(W_1(X) \cdot W_2(X))^{-1}$
      else $b_l \leftarrow 1$              # yes $\rightarrow$ strip off $(W_1(X)^2 \cdot W_2(X))^{-1}$
    fi
    $D(X) \leftarrow \left(W_1(X)^{b_l+1} \cdot W_2(X)\right)^{-1} \cdot D(X)$
    $l \leftarrow l + 1$                       # proceed with next bit of plaintext
  od
  return $b_1 \dots b_{l-1}$
end
```

Of course, one may think of elaborating the approach taken in Procedure 1 by overriding the decision for certain plaintext bits in the while-loop—e. g., based on the part of the plaintext which has been recovered already. However, the following examples illustrate that already this simple version works quite well; as always in the sequel for the computations we use the computer algebra system MAGMA V2.5-1 (see [1]) on a Linux platform with 800 MHz:

Example 1. This example is based on matrices $F_1(X), F_2(X)$ taken from [4]: setting

$$V_1 := \begin{pmatrix} -1 & 0 \\ 1 & -1 \end{pmatrix}$$
$$V_2 := \begin{pmatrix} -1 & 1 \\ 0 & -1 \end{pmatrix}$$

$$\begin{aligned} F_1(X) &:= \begin{pmatrix} -1 & (X-\sqrt{3})(X-2\sqrt{3}) \\ \frac{1}{3}(X-\sqrt{3})^2 & -1 \end{pmatrix} \\ F_2(X) &:= \begin{pmatrix} -1 & \frac{1}{3}(X-\sqrt{3})^2 \\ (X-\sqrt{3})(X-2\sqrt{3}) & -1 \end{pmatrix} \end{aligned} \tag{2}$$

we obtain $F_1(2\sqrt{3}) = V_1$ and $F_2(2\sqrt{3}) = V_2$. Moreover, we choose the matrix M as

$$M := \begin{pmatrix} -5 & \frac{1}{2} \\ 3 & -\frac{1}{4} \end{pmatrix}.$$

Finally, we define $W_1(X) := M^{-1} \cdot F_1(X) \cdot M$, $W_2(X) := M^{-1} \cdot F_2(X) \cdot M$ and use these parameters to encrypt the bitstring corresponding to the ASCII representation of the text "A small example." (128 bit) with the encryption rule (1): we obtain a matrix $C(X)$ whose entries are (dense) polynomials of degree 626. As already each of the constant terms of the diagonal entries is an integer with 225 decimal digits, we do not write down the matrix $C(X)$ explicitly here (in particular this example supports the hypothesis from [4] that the public key cryptosystem under consideration is only of limited practical value). Nevertheless, applying Procedure 1 to $C(X)$ yields the correct plaintext within a few minutes.

Example 2. Setting

$$\begin{aligned} V_1 &:= \begin{pmatrix} -1 & 0 \\ -35 & -1 \end{pmatrix} \\ V_2 &:= \begin{pmatrix} -1 & -61 \\ 0 & -1 \end{pmatrix} \end{aligned}$$

$$\begin{aligned} F_1(X) &:= \begin{pmatrix} X^2-86X+1847 & X^3-126X^2+5297X-74298 \\ 5X^4-840X^3+52920X^2-1481767X+15558739 & X^2-88X+1931 \end{pmatrix} \\ F_2(X) &:= \begin{pmatrix} X^3-126X^2+5297X-74299 & X^4-168X^3+10588X^2-296688X+3118691 \\ X^2-86X+1848 & X^3-126X^2+5288X-73921 \end{pmatrix} \end{aligned}$$

we obtain $F_1(42) = V_1$ and $F_2(42) = V_2$. Moreover, we choose the matrix M as

$$M := \begin{pmatrix} -1 & 2 \\ 1 & -3 \end{pmatrix}.$$

Finally, we define $W_1(X) := M^{-1} \cdot F_1(X) \cdot M$, $W_2(X) := M^{-1} \cdot F_2(X) \cdot M$ and use these parameters to encrypt the bitstring corresponding to the ASCII representation of the text "a secret message" (128 bit) with the encryption rule (1): we obtain a matrix $C(X)$ whose entries are (dense) polynomials of degree 1150. Again, applying Procedure 1 to $C(X)$ yields the correct plaintext within a few minutes.

In [4] the possibility is mentioned to replace the univariate polynomial ring $\mathbb{C}[X]$ with a multivariate polynomial ring $\mathbb{C}[X_1, \dots, X_r]$. Obviously, this modification does not vitiate the above attack, and the question arises which public keys are not vulnerable to the approach of Procedure 1. An obvious way to prevent this kind of attack is to impose an appropriate restriction on the choice of $W_1(X), W_2(X)$: if both $W_1(X), W_2(X)$ and $W_1(X)^{-1}, W_2(X)^{-1}$ are contained in $\mathrm{SL}_2(\mathbb{C}[X])$ already, then Procedure 1 does not terminate, as no non-constant denominators can occur when multiplying $D(X)$ with $\big(W_1(X)^2 \cdot W_2(X)\big)^{-1}$.

Unfortunately, this condition is not sufficient to guarantee the security of the cryptosystem either, as the above attack can be adapted easily: again, the basic idea is to strip off the plaintext from $C(X)$ bit by bit. The loophole we can exploit to do this is the message expansion occurring during encryption: in the cryptosystem under consideration one can expect that a short plaintext encrypts to a matrix with "simple" polynomials and a long plaintext encrypts to a matrix with "complicated" polynomials. Taking the number of terms in a polynomial for a measure of its complexity this idea motivates the following variant of Procedure 1 (for a matrix A we denote by $A_{i,j}$ the entry of A in row i and column j):

Procedure 2

In: *Public key* $(W_1(X), W_2(X))$
Ciphertext $C(X) = W_2(X) \cdot \prod_{i=1}^{n} (W_1(X)^{b_i+1} W_2(X))$
Out: $\perp$ *or the plaintext* $b_1 \dots b_n$

```
begin
  $D(X) \leftarrow W_2(X)^{-1} \cdot C(X)$                  # remove superfluous factor
  $l \leftarrow 1$                         # number of plaintext bit to be processed next
  while $D(X)$ is not the identity do            # decryption still incomplete
    $D'(X) \leftarrow (W_1(X) \cdot W_2(X))^{-1} \cdot D(X)$
    $n_0 \leftarrow \sum_{1 \le i,j \le 2} |\mathit{Terms}(D'(X)_{i,j})|$          # number of terms if left-
                                                  # most bit is assumed to be 0
    $D'(X) \leftarrow \big(W_1(X)^2 \cdot W_2(X)\big)^{-1} \cdot D(X)$
    $n_1 \leftarrow \sum_{1 \le i,j \le 2} |\mathit{Terms}(D'(X)_{i,j})|$          # number of terms if left-
                                                  # most bit is assumed to be 1
    if $n_0 < n_1$                               # Should left-most bit be reset?
      then $b_l \leftarrow 0$                # yes $\rightarrow$ strip off $(W_1(X) \cdot W_2(X))^{-1}$
      else $b_l \leftarrow 1$                # no $\rightarrow$ strip off $(W_1(X)^2 \cdot W_2(X))^{-1}$
    fi
    $D(X) \leftarrow \big(W_1(X)^{b_l+1} \cdot W_2(X)\big)^{-1} \cdot D(X)$
    $l \leftarrow l + 1$                          # proceed with next bit of plaintext
  od
  return $b_1 \dots b_{l-1}$
end
```

The next example demonstrates that this simple procedure can indeed be applied successfully to the cryptosystem under consideration:

Example 3. Setting

$$V_1 := \begin{pmatrix} -1 & -3 \\ 0 & -1 \end{pmatrix}$$
$$V_2 := \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}$$

$$F_1 := \begin{pmatrix} -X^2+2X+3 & -X^4+4X^3+X^2-10X-7 \\ -X^2+2X+4 & -X^4+4X^3+2X^2-12X-9 \end{pmatrix}$$
$$F_2 := \begin{pmatrix} 5X^2-10X-19 & 5X^2-10X-20 \\ 5X^2-10X-18 & 5X^2-10X-19 \end{pmatrix}$$
$$M := \begin{pmatrix} -5 & 2 \\ 1 & 4 \end{pmatrix}$$

we obtain $F_1(\sqrt{5}+1) = V_1$, $F_2(\sqrt{5}+1) = V_2$, and the public key computes to

$$W_1(X) := \frac{1}{11} \cdot \begin{pmatrix} X^4-4X^3-5X^2+18X+15 & 4X^4-16X^3+2X^2+28X+16 \\ -3X^4+12X^3+\frac{41}{2}X^2-65X-\frac{167}{2} & -12X^4+48X^3+16X^2-128X-81 \end{pmatrix}$$
$$W_2(X) := \frac{1}{11} \cdot \begin{pmatrix} 20X^2-40X-79 & -30X^2+60X+124 \\ -60X^2+120X+215 & 90X^2-180X-339 \end{pmatrix}.$$

Encrypting the bitstring corresponding to the ASCII representation of the text "This is the plaintext." (176 bit) with this public key according to the encryption rule (1) yields a matrix $C(X)$ whose entries are (dense) polynomials of degree 1366. By means of Procedure 2 we can recover the plaintext from $C(X)$ within a few minutes—without requiring the private key $(M, \sqrt{5}+1)$.

In summary, we conclude that the public key cryptosystem suggested in [4] in the present form is not secure, as no possibility for constructing public keys which are immune to the described attacks is provided.

4 Attacking Another Cryptosystem Using the Modular Group

At ACISP'99 A. Yamamura suggested another public key cryptosystem using the modular group which has some similarity with the system considered above. For a full description of this system we refer to the original paper [5]. Here we only recall the aspects of the system which are relevant for the attack described below: as in the cryptosystem of [4], for constructing a public key one starts by choosing appropriate matrices $V_1, V_2 \in \mathrm{SL}_2(\mathbb{Z})$. Moreover, a suitable complex number $p \in \mathbb{C}$ and some non-singular matrix $M \in \mathrm{GL}_2(\mathbb{R})$ have to be chosen. Then the two matrices

$$W_1 := M^{-1} \cdot V_1 \cdot M \in \mathrm{SL}_2(\mathbb{R}), \quad W_2 := M^{-1} \cdot V_2 \cdot M_2 \in \mathrm{SL}_2(\mathbb{R})$$

and the complex number p are made public. To encrypt the bitstring $b_1 \ldots b_n \in \{0,1\}^*$ with the public key one starts by computing the matrix product $C := \prod_{i=1}^{n}(W_{b_i+1}) \in \mathrm{SL}_2(\mathbb{R})$. Then the ciphertext c is given by the complex number

$$c := \frac{C_{1,1} \cdot p + C_{1,2}}{C_{2,1} \cdot p + C_{2,2}}.$$

Equivalently, we can also start by computing

$$\begin{aligned} c_n &:= \frac{W_{b_n+1_{1,1}} \cdot p + W_{b_n+1_{1,2}}}{W_{b_n+1_{2,1}} \cdot p + W_{b_n+1_{2,2}}} \\ c_{n-1} &:= \frac{W_{b_{n-1}+1_{1,1}} \cdot c_n + W_{b_{n-1}+1_{1,2}}}{W_{b_{n-1}+1_{2,1}} \cdot c_n + W_{b_{n-1}+1_{2,2}}} \\ &\vdots \end{aligned}$$

and continue in this way until we finally obtain the ciphertext

$$c = c_1 := \frac{W_{b_1+1_{1,1}} \cdot c_2 + W_{b_1+1_{1,2}}}{W_{b_1+1_{2,1}} \cdot c_2 + W_{b_1+1_{2,2}}}.$$

Now the question arises whether we can recover the plaintext $b_1 \ldots b_n \in \{0,1\}^*$ efficiently from the ciphertext c and the public data alone. As all the matrices involved are contained in $\mathrm{SL}_2(\mathbb{R})$ it seems worthwhile to have a look at Procedure 2 again. The essential idea in this procedure was to exploit the message expansion occurring during encryption. To measure this expansion we used the number of terms occurring in the matrix. In the cryptosystem from [5] there is a similar phenomenon: for a long plaintext we expect the matrix C resp. the resulting ciphertext c to be "more complicated" than for a short plaintext.

In order to define a suitable measure of complexity (in analogy to the number of terms used above) it is helpful to have some information about the possible coefficients which can occur—for computational reasons it is not practical to consider coefficients which are arbitrary real or complex numbers. Motivated by the discussion in the last section of [5] here we will restrict our attention to the case that all the real numbers occurring are contained in some number field $\mathbb{Q}(\alpha) = \mathbb{Q}[\alpha]$ already. Hence, we can express each element $\eta \in \mathbb{Q}[\alpha]$ uniquely in the form $\eta = \sum_{i=0}^{r} a_i \cdot \alpha^i$ where $a_0, \ldots, a_r \in \mathbb{Q}$ and $\alpha^0, \ldots, \alpha^r$ is a vector space basis of $\mathbb{Q}[\alpha]$ over $\mathbb{Q}$.

Using this representation we can regard the number of binary digits required for expressing the absolute value of the occurring numerators resp. denominators as a measure for the "complexity" of the number η. This motivates the following procedure:

Procedure 3

In: *Public data* (W_1, W_2, p) *with* $W_1, W_2 \in \mathrm{SL}_2(\mathbb{Q}[\alpha])$ *and* $p \in \mathbb{Q}[\alpha]$
Ciphertext $c = \frac{C_{1,1}\cdot p + C_{1,2}}{C_{2,1}\cdot p + C_{2,2}}$ *where* $C = \prod_{i=1}^{n} W_{b_i+1}$
Out: $\perp$ *or the plaintext* $b_1 \dots b_n$

```
begin
  $d \leftarrow c$                                         # partially decrypted ciphertext
  $l \leftarrow 1$                                         # number of plaintext bit to be processed next
  while $d \neq p$ do                                      # decryption still incomplete
    $D \leftarrow {W_1}^{-1}$                              # "complexity" if left-most bit is assumed to be 0
    $a_0 + a_1 \cdot \alpha + \dots + a_r \cdot \alpha^r \leftarrow \frac{D_{1,1}\cdot d + D_{1,2}}{D_{2,1}\cdot d + D_{2,2}}$
    $n_0 \leftarrow \sum_{a_i \neq 0} \log_2 |\,Numerator(a_i) \cdot Denominator(a_i)|$
    $D \leftarrow {W_2}^{-1}$                              # "complexity" if left-most bit is assumed to be 1
    $a_0 + a_1 \cdot \alpha + \dots + a_r \cdot \alpha^r \leftarrow \frac{D_{1,1}\cdot d + D_{1,2}}{D_{2,1}\cdot d + D_{2,2}}$
    $n_1 \leftarrow \sum_{a_i \neq 0} \log_2 |\,Numerator(a_i) \cdot Denominator(a_i)|$
    if $n_0 < n_1$                                         # Should left-most bit be reset?
      then $(b_l, D) \leftarrow (0, W_1^{-1})$             # yes $\rightarrow$ strip off $W_1^{-1}$
      else $(b_l, D) \leftarrow (1, W_2^{-1})$             # no $\rightarrow$ strip off $W_2^{-1}$
    fi
    $d \leftarrow \frac{D_{1,1}\cdot d + D_{1,2}}{D_{2,1}\cdot d + D_{2,2}}$
    $l \leftarrow l + 1$                                   # proceed with next bit of plaintext
  od
  return $b_1 \dots b_{l-1}$
end
```

To check the relevance of Procedure 3 we apply it to some examples:

Example 4. Setting

$$V_1 := \begin{pmatrix} -1 & 0 \\ 123 & -1 \end{pmatrix}$$

$$V_2 := \begin{pmatrix} -1 & 321 \\ 0 & -1 \end{pmatrix}$$

$$M := \begin{pmatrix} 1 & 3 \\ 2 & -1 \end{pmatrix}$$

$$p := -\frac{9}{5} \cdot \left(\zeta_7^5 + \zeta_7^4 + \zeta_7^3 + \zeta_7^2 + \zeta_7 + \frac{13}{9} \right)$$

(where ζ_7 is a primitive 7-th root of unity) the public matrices W_1, W_2 compute to

$$\begin{aligned} W_1 &= \frac{1}{7} \cdot \begin{pmatrix} 362 & 1107 \\ -123 & -376 \end{pmatrix} \\ W_2 &= \frac{1}{7} \cdot \begin{pmatrix} 635 & -321 \\ 1284 & -649 \end{pmatrix}. \end{aligned}$$

Encrypting the bitstring corresponding to the ASCII representation of "Yet another plaintext ..." (200 bit) with these parameters yields a ciphertext c, and by applying Procedure 3 to c we obtain the correct plaintext within a few minutes.

Example 5. Setting

$$\begin{aligned} V_1 &:= \begin{pmatrix} 1 & 0 \\ -32 & 1 \end{pmatrix} \\ V_2 &:= \begin{pmatrix} -1 & 11 \\ 0 & -1 \end{pmatrix} \\ M &:= \begin{pmatrix} 3 & 2 \\ -\sqrt{-11} & 1 \end{pmatrix} \\ p &:= -\frac{1}{10} \cdot \sqrt{-11} + \frac{12}{5} \end{aligned}$$

the public matrices W_1, W_2 compute to

$$\begin{aligned} W_1 &= \frac{1}{53} \cdot \begin{pmatrix} -384 \cdot \sqrt{-11} + 629 & -256 \cdot \sqrt{-11} + 384 \\ 576 \cdot \sqrt{-11} - 864 & 384 \cdot \sqrt{-11} - 523 \end{pmatrix} \\ W_2 &= \frac{1}{53} \cdot \begin{pmatrix} -33 \cdot \sqrt{-11} - 295 & -22 \cdot \sqrt{-11} + 33 \\ -242 \cdot \sqrt{-11} + 363 & 33 \cdot \sqrt{-11} + 189 \end{pmatrix}. \end{aligned}$$

Encrypting the bitstring corresponding to the ASCII representation of "Unfortunately, it is not necessary to know the private key for reading this." (608 bit) with these parameters yields a ciphertext c. Applying Procedure 3 to c yields the correct plaintext within a few seconds.

Analogously as in the previous section, one may think of elaborating the approach taken in Procedure 3 by overriding the decision for certain plaintext bits in the while-loop—e. g., based on the part of the plaintext which has been recovered already. However, the above examples illustrate that already this crude variant works quite well.

5 Related Work and Conclusions

It is worth mentioning that the attacks described in this paper are somewhat reminiscent of a property of the $\mathrm{SL}_2(\mathbb{F}_{2^n})$ hashing scheme of J.-P. Tillich and

G. Zémor [3]. In the latter the hash value of a bitstring is given by a matrix in $\mathrm{SL}_2(\mathbb{F}_{2^n})$, and similarly as above for very short bitstrings it is possible to recover the original bitstring from its hash value "bit by bit" (cf. [2, Proposition 2]).

The reason for the vulnerability of the cryptosystems in [4,5] to this kind of attack is the possibility to exploit the message expansion occurring in both of these cryptosystems. Consequently, as the given specifications do not rule out such an attack we conclude that in the present form the public key cryptosystems described in [4,5] must be considered as insecure.

Acknowledgements

The author is indebted to Willi Geiselmann and Markus Grassl for various fruitful discussions and helpful comments.

References

1. W. Bosma, J. Cannon, and C. Playoust, *The Magma Algebra System I: The User Language*, Journal of Symbolic Computation, 24 (1997), pp. 235–265.
2. R. Steinwandt, M. Grassl, W. Geiselmann, and T. Beth, *Weaknesses in the* $\mathrm{SL}_2(\mathbb{F}_{2^n})$ *Hashing Scheme*, in Advances in Cryptology – CRYPTO 2000 Proceedings, M. Bellare, ed., vol. 1880 of Lecture Notes in Computer Science, Springer, 2000, pp. 287–299.
3. J.-P. Tillich and G. Zémor, *Hashing with* SL_2, in Advances in Cryptology – CRYPTO '94, Y. Desmedt, ed., vol. 839 of Lecture Notes in Computer Science, 1994, pp. 40–49.
4. A. Yamamura, *Public-Key Cryptosystems Using the Modular Group*, in Public Key Cryptography; First International Workshop on Practice and Theory in Public Key Cryptography, PKC '98, H. Imai and Y. Zheng, eds., vol. 1431 of Lecture Notes in Computer Science, Berlin; Heidelberg, 1998, Springer, pp. 203–216.
5. ———, *A Functional Cryptosystem Using a Group Action*, in Information Security and Privacy; 4th Australasian Conference, ACISP'99, J. Pieprzyk, R. Safavi-Naini, and J. Seberry, eds., vol. 1587 of Lecture Notes in Computer Science, Berlin; Heidelberg, 1999, Springer, pp. 314–325.