SECURITY EVALUATION OF MUGI

JOVAN GOLIĆ

JULY 31, 2002

Contents

1. Introduction 3

2. Description of MUGI 4

2.1 Keystream Generation 4

2.2 Initialization 6

3. Analysis of Buffer 6

3.1 Linear Recurrences 7

3.2 Generating Functions 7

3.3 Solution 8

3.4 Properties 10

4. Elimination of Buffer 11

5. Linear Cryptanalysis 12

5.1 Linear Approximations for F 12

5.2 LFSM Approximations for MUGI 13

5.2.1 Basic Equation 14

5.2.2 Initial State Reconstruction 16

5.2.3 Linear Statistical Weakness 18

5.3 LFSM Approximations for Simplified MUGI 19

6. On Low-Diffusion Statistical Distinguishers 21

7. On Using Overdefined Equations for S-boxes 23

8. Summary of Weaknesses and Strengths 24

9. References 25

1. Introduction

MUGI is a specific keystream generator for stream cipher applications proposed in [SpM]. Due to its design rationale, it is mainly suitable for software implementations. Efficient hardware implementations are also possible, but are more complex than the usual designs based on linear feedback shift registers (LFSRs), nonlinear combining functions, and irregular clocking.

It is interesting to note that in mathematical terms, the structure of MUGI is essentially one of a combiner with memory, which is a well-known type of keystream generators (see [G96a]). Specific features are the following:

- the nonlinear combining function has a large internal memory size and is based on a round function of the block cipher AES [AES]
- the driving linear finite-state machine (LFSM) providing input to the combining function is not an LFSR with a primitive connection polynomial
- the LFSM receives feeback from a part of the internal memory of the combining function
- the output at a given time is a binary word taken from the internal memory of the combining function.

A security analysis of MUGI is presented in [EvM]. The main claims from [EvM] are essentially that MUGI is not vulnerable to common attacks on block ciphers and also to some attacks on stream ciphers. However, some general methods for analyzing stream ciphers based on combiners with memory, most notably the so-called linear cryptanalysis of stream ciphers [G92, G94, G96a, G96b], are not addressed in [EvM] at all. Since MUGI can essentially be regarded as a combiner with memory, such methods are in principle also applicable to MUGI. Also, the underlying LFSM of MUGI, the so-called buffer, is not analyzed in [EvM].

Linear cryptanalysis of stream ciphers is essentially different from linear cryptanalysis of block ciphers because of the underlying iterative structure in which the initial state is unknown. It essentially consists in finding linear relations among the unknown internal variables, possibly conditioned on the known output sequence, which hold with probabilities different from one half. It has two main objectives:

- to reconstruct the secret key, in particular, the initial state of the keystream generator
- to derive a linear statistical distinguisher which can distinguish the output sequence from a purely random sequence.

Surprisingly, the recently introduced, so-called cryptanalysis of stream ciphers with linear masking [CHJ02] is not original and is just a special case of the linear cryptanalysis of stream ciphers mentioned above. However, [CHJ02] also contains a conceptually new, so-called low-diffision statistical distinguisher for certain types of stream ciphers.

A new method od cryptanalyzing block ciphers, potentially applicable to AES, is recently proposed in [CP02]. It is essentially based on the multiply-and-linearize method applied to an overdefined system of relatively sparse quadratic binary equations that can be associated with S-boxes of AES. Since the same S-boxes are used in MUGI, this new method also deserves some attention.

This report is organized as follows. Section 2 contains a brief description of MUGI. Analysis of the LFSM of MUGI is presented in Section 3, a related transformation of the underlying system of nonlinear recurrences is given in Section 4, and the linear cryptanalysis of MUGI is developed in Section 5. Section 6 is devoted to low-diffusion statistical distinguishers and

Section 7 to using overdefined systems of sparse equations associated with S-boxes. Finally, Section 8 contains a summary of weaknesses and strengths of MUGI.

2. Description of MUGI

A concise description of MUGI is specified here in as much detail as needed for the analysis. More details can be found in [SpM].

2.1 Keystream Generation

The keystream generator is a finite-state machine (FSM) whose internal state has two components:

- a linearly updated component, called buffer, $b = b_0 b_1 \cdots b_{15}$, where each b_i is a 64-bit word; the size of this component is 2^{10} bits
- a nonlinearly updated component, called state, $a = a_0 a_1 a_2$, where each a_i is a 64-bit word; the size of this component is 192 bits.

The next-state or update function is invertible and has two components, $\varphi = (\rho, \lambda)$, where ρ updates a and λ updates b, that is,

$$(a^{(t+1)}, b^{(t+1)}) = \varphi(a^{(t)}, b^{(t)}) = (\rho(a^{(t)}, b^{(t)}), \lambda(a^{(t)}, b^{(t)})).$$

The ρ component is an invertible nonlinear function defined in terms of an invertible 64×64-bit function F by a kind of Feistel structure depicted in Fig. 1. The function F is shown in Fig. 2. It is derived from the round function of AES and as such consists of 8 8×8-bit S-boxes and two linear 32×32-bit, MixColumn, transformations from AES (see [AES]). C_1 and C_2 are 64-bit constants.

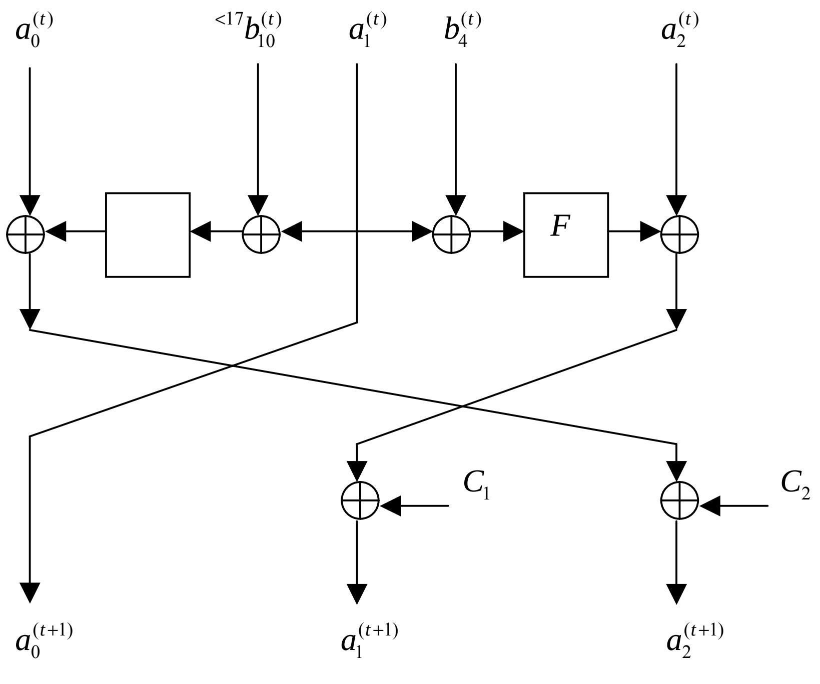

Figure 1: The ρ update function.

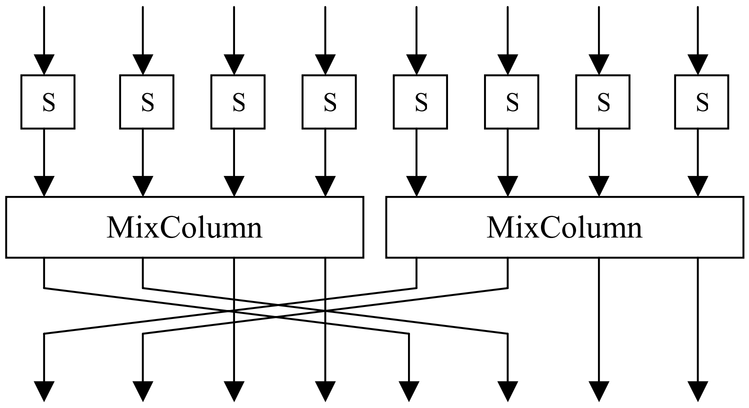

Figure 2: The F function.

The corresponding equations for ρ are given below:

$$\begin{aligned}
a_0^{(t+1)} &= a_1^{(t)} \\
a_1^{(t+1)} &= a_2^{(t)} \oplus F(a_1^{(t)} \oplus b_4^{(t)}) \oplus C_1 \\
a_2^{(t+1)} &= a_0^{(t)} \oplus F(a_1^{(t)} \oplus {}^{<17} b_{10}^{(t)}) \oplus C_2 .
\end{aligned} \qquad (1)$$

The λ component is an invertible linear function defined by the following equations:

$$\begin{aligned}
b_i^{(t+1)} &= b_{i-1}^{(t)}, \qquad i \neq 0, 4, 10 \\
b_0^{(t+1)} &= b_{15}^{(t)} \oplus a_0^{(t)} \\
b_4^{(t+1)} &= b_3^{(t)} \oplus b_7^{(t)} \\
b_{10}^{(t+1)} &= b_9^{(t)} \oplus {}^{<32} b_{13}^{(t)} .
\end{aligned}$$

(For a 64-bit word x, ${}^{<i}x$ and ${}^{>i}x$ denote the rotations of x by i bits to the left and right, respectively.)

The 64-bit output of the keystream generator at time t is defined as $a_2^{(t)}$.

The described structure is essentially a specific combiner with memory, which is a well-known type of keystream generators (see [G96a]), in which:

- the nonlinear combining function has a large internal memory size and is based on a round function of block ciphers
- the driving LFSM providing input to the combining function is not necessarily an LFSR with a primitive connection polynomial; in fact, it will be shown that the design of LFSM for MUGI is not good
- the LFSM may be non-autonomous, that is, may have an input taken from a part of the internal memory of the combining function
- the output at a given time is not a single bit, but a binary word taken from the internal memory of the combining function.

2.2 Initialization

The initial internal state $(a^{(0)}, b^{(0)})$ of the keystream generator is produced from the 128-bit secret key $K = K_0K_1$ and the 128-bit initialization vector $IV = IV_0IV_1$ in the following three stages, by using the keystream generator itself. In the first two stages only the ρ function is used.

Firstly, the state a is defined in terms of K and a 64-bit constant C_0 by:

$$\begin{aligned} a_0 &= K_0 \\ a_1 &= K_1 \\ a_2 &= {}^{<7}K_0 \oplus {}^{>7}K_1 \oplus C_0 . \end{aligned}$$

The buffer $b(K)$ is then defined by iterating ρ as follows:

$$b(K)_{15-i} = (\rho^{i+1}(a,0))_0, \qquad 0 \le i \le 15 .$$

Secondly, the last produced state $a(K) = \rho^{16}(a,0)$ and IV are linearly combined together into $a(K, IV)$ by:

$$\begin{aligned} a(K,IV)_0 &= a(K)_0 \oplus IV_0 \\ a(K,IV)_1 &= a(K)_1 \oplus IV_1 \\ a(K,IV)_2 &= a(K)_2 \oplus {}^{<7}IV_0 \oplus {}^{>7}IV_1 \oplus C_0 . \end{aligned}$$

The ρ function is again iterated 16 times to produce $\rho^{16}(a(K, IV),0)$.

Thirdly, the keystream generator is initialized by $\rho^{16}(a(K, IV),0)$ and $b(K)$ and then iterated 15 times (both ρ and λ) without producing output. The keystream generation starts from the 16th iteration on. Thus, effectively, the initial contents of both a and b, at the time when the first output is produced, depend on both K and IV. However, it is important to note that the content of b at the beginning of the third stage depends on K only.

3. Analysis of Buffer

In this section, the buffer is analyzed as a non-autonomous LFSM with one input sequence, namely, $a_0 = \left(a_0^{(t)}\right)_{t=0}^{\infty}$. The input sequence and all the internal sequences in the buffer are 64-bit sequences. Our objective is to derive expressions for the internal sequences in the buffer in terms of the input sequence a_0 and the initial state of the buffer, $b^{(0)} = b_0^{(0)} b_1^{(1)} \cdots b_{15}^{(15)}$.

In view of the λ update function, the 16 internal sequences in the buffer can be divided in three groups, in each group the sequences being phase shifts of each other (see Fig. 3).

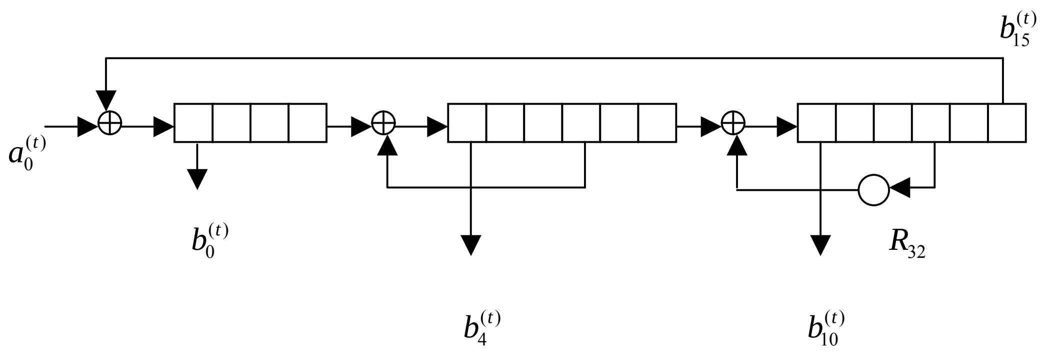

Figure 3: The buffer as a LFSM.

3.1 Linear Recurrences

From the λ update function, we directly obtain the following linear recurrences, all holding for $t \geq 1$:

$$\begin{aligned} b_4^{(t)} &= b_4^{(t-4)} \oplus b_0^{(t-4)} \\ b_{10}^{(t)} &= R_{32} b_{10}^{(t-4)} \oplus b_4^{(t-6)} \\ b_0^{(t)} &= a_0^{(t-1)} \oplus b_{10}^{(t-6)} \end{aligned}$$

where R_j denotes the rotation by j bits to the left, which is a linear transformation of a 64-bit word. In vectorial notation where vectors are represented as one-column matrices, R_{32} is represented as a matrix. The initial state of the buffer can now be represented as $\left(b_0^{(t)}\right)_{t=-3}^{0} \left(b_4^{(t)}\right)_{t=-5}^{0} \left(b_{10}^{(t)}\right)_{t=-5}^{0}$. Then, by eliminating b_0, we obtain

$$\begin{aligned} b_4^{(t)} &= b_4^{(t-4)} \oplus b_{10}^{(t-10)} \oplus a_0^{(t-5)}, \qquad t \geq 5 \\ b_{10}^{(t)} &= b_4^{(t-6)} \oplus R_{32} b_{10}^{(t-4)}, \qquad t \geq 1. \end{aligned} \tag{2}$$

This is a system of two 64-bit linear recurrences (that is, 128 binary linear recurrences) in terms of 64-bit sequences b_4 and b_{10}.

3.2 Generating Functions

The system can be solved by using the generating function technique dealing with the z-transforms of 64-bit sequences. In vectorial notation, the z-transforms or generating functions of b_4, b_{10}, and a_0 are defined as formal power series

$$B_4 = \sum_{t=0}^{\infty} b_4^{(t)} z^t, \quad B_{10} = \sum_{t=0}^{\infty} b_{10}^{(t)} z^t, \quad \text{and} \quad A_0 = \sum_{t=0}^{\infty} a_0^{(t)} z^t .$$

Firstly, we obtain

$$\sum_{t=5}^{\infty} b_4^{(t)} z^t = z^4 \sum_{t=5}^{\infty} b_4^{(t-4)} z^{t-4} \oplus z^{10} \sum_{t=5}^{\infty} b_{10}^{(t-10)} z^{t-10} \oplus z^5 \sum_{t=5}^{\infty} a_0^{(t-5)} z^{t-5}$$
$$\sum_{t=1}^{\infty} b_{10}^{(t)} z^t = z^6 \sum_{t=1}^{\infty} b_4^{(t-6)} z^{t-6} \oplus z^4 R_{32} \sum_{t=1}^{\infty} b_{10}^{(t-4)} z^{t-4} .$$

Then, we get

$$\begin{aligned}(1 \oplus z^4) B_4 \oplus z^{10} B_{10} &= z^5 A_0 \oplus \sum_{t=0}^{4} b_4^{(t)} z^t \oplus z^4 b_4^{(0)} \oplus \sum_{t=5}^{9} b_{10}^{(t-10)} z^t \\ &= z^5 A_0 \oplus (1 \oplus z^4) b_4^{(0)} \oplus \sum_{t=1}^{4} (b_4^{(t-4)} \oplus b_0^{(t-4)}) z^t \oplus \sum_{t=5}^{9} b_{10}^{(t-10)} z^t \\ &= z^5 A_0 \oplus b_4^{(0)} \oplus z^4 b_0^{(0)} \oplus \sum_{t=1}^{3} (b_4^{(t-4)} \oplus b_0^{(t-4)}) z^t \oplus \sum_{t=5}^{9} b_{10}^{(t-10)} z^t \\ (I \oplus z^4 R_{32}) B_{10} \oplus z^6 B_4 &= \sum_{t=1}^{5} b_4^{(t-6)} z^t \oplus b_{10}^{(0)} \oplus R_{32} \sum_{t=1}^{3} b_{10}^{(t-4)} z^t\end{aligned}$$

where I denotes the 64×64 identity matrix. In a simplified notation, we thus have

$$\begin{aligned}&(1 \oplus z^4) B_4 \oplus z^{10} B_{10} = z^5 A_0 \oplus \Delta_1 \\ &z^6 B_4 \oplus (I \oplus z^4 R_{32}) B_{10} = \Delta_2\end{aligned} \qquad (3)$$

where

$$\Delta_1 = b_4^{(0)} \oplus z^4 b_0^{(0)} \oplus \sum_{t=1}^{3} (b_4^{(t-4)} \oplus b_0^{(t-4)}) z^t \oplus \sum_{t=5}^{9} b_{10}^{(t-10)} z^t$$
$$\Delta_2 = \sum_{t=1}^{5} b_4^{(t-6)} z^t \oplus b_{10}^{(0)} \oplus R_{32} \sum_{t=1}^{3} b_{10}^{(t-4)} z^t$$

are 64-dimensional vectors (64×1 matrices) whose elements are polynomials in z defined by the initial state of the buffer and whose degrees are at most 9 and 5, respectively. Essentially, this is a system of 128 linear equations with coefficients being polynomials in z and with unknowns being 128 generating functions of 64 binary sequences in b_4 and 64 binary sequences in b_{10}.

3.3 Solution

The system has a unique solution which can be found in the following way. First, by elimination we obtain

$$\left(I \oplus z^4 (I \oplus R_{32}) \oplus z^8 R_{32} \oplus z^{16} I\right) B_4 = z^5 (I \oplus z^4 R_{32}) A_0 \oplus (I \oplus z^4 R_{32}) \Delta_1 \oplus z^{10} \Delta_2$$

$$\left(I \oplus z^4 (I \oplus R_{32}) \oplus z^8 R_{32} \oplus z^{16} I\right) B_{10} = z^{11} A_0 \oplus z^6 \Delta_1 \oplus (1 \oplus z^4) \Delta_2 .$$

Let

$$F(z) = I \oplus z^4 (I \oplus R_{32}) \oplus z^8 R_{32} \oplus z^{16} I \qquad (4)$$

denote the 64×64 matrix whose coefficients are polynomials in z of degree at most 16. The system can then be written as

$$
\begin{aligned}
&F(z)B_4 = z^5(I \oplus z^4 R_{32})A_0 \oplus (I \oplus z^4 R_{32})\Delta_1 \oplus z^{10}\Delta_2 \\
&F(z)B_{10} = z^{11}A_0 \oplus z^6\Delta_1 \oplus (1 \oplus z^4)\Delta_2 .
\end{aligned}
$$

When regarded over a field of rational functions in z, $F(z)$ is invertible as is seen from the following equation:

$$
\begin{aligned}
F(z)F(z) = F(z)^2 &= \left(I \oplus z^4(I \oplus R_{32}) \oplus z^8 R_{32} \oplus z^{16} I\right)^2 \\
&= I \oplus z^8(I \oplus R_{32}^2) \oplus z^{16} R_{32}^2 \oplus z^{32} I \\
&= I \oplus z^8(I \oplus I) \oplus z^{16} I \oplus z^{32} I = (1 \oplus z \oplus z^2)^{16} I
\end{aligned}
$$

because of $R_{32}^2 = I$. Thus we get that $\dfrac{1}{f(z)}F(z)$ is the inverse of $F(z)$, where

$$
f(z) = 1 \oplus z^{16} \oplus z^{32} = (1 \oplus z \oplus z^2)^{16}.
$$

Accordingly, we obtain the solution for the generating functions B_4 and B_{10} in the form of

$$
\begin{aligned}
B_4 &= \frac{1}{f(z)} F(z) \left(z^5(I \oplus z^4 R_{32})A_0 \oplus (I \oplus z^4 R_{32})\Delta_1 \oplus z^{10}\Delta_2\right) \\
B_{10} &= \frac{1}{f(z)} F(z) \left(z^{11}A_0 \oplus z^6\Delta_1 \oplus (1 \oplus z^4)\Delta_2\right).
\end{aligned}
$$

Unfortunately, it turns out that the polynomial $f(z)$ has a very small exponent (period), equal to 48, because of

$$
f(z) = \frac{1 \oplus z^{48}}{1 \oplus z^{16}} .
$$

So, the solution can also be put into the form of

$$
\begin{aligned}
B_4 &= \frac{1 \oplus z^{16}}{1 \oplus z^{48}} F(z) \left(z^5(I \oplus z^4 R_{32})A_0 \oplus (I \oplus z^4 R_{32})\Delta_1 \oplus z^{10}\Delta_2\right) \\
B_{10} &= \frac{1 \oplus z^{16}}{1 \oplus z^{48}} F(z) \left(z^{11}A_0 \oplus z^6\Delta_1 \oplus (1 \oplus z^4)\Delta_2\right).
\end{aligned}
$$

Equivalently, we have

$$
\begin{aligned}
B_4 &= \frac{1}{f(z)} z^5 G(z) A_0 \oplus \frac{1}{f(z)} \Delta_1' \\
&= \frac{1}{1 \oplus z^{48}} z^5 (1 \oplus z^{16}) G(z) A_0 \oplus \frac{1}{1 \oplus z^{48}} (1 \oplus z^{16}) \Delta_1'
\end{aligned}
\tag{5}
$$

$$B_{10} = \frac{1}{f(z)} z^{11} F(z) A_0 \oplus \frac{1}{f(z)} \Delta_2' \\ = \frac{1}{1 \oplus z^{48}} z^{11} (1 \oplus z^{16}) F(z) A_0 \oplus \frac{1}{1 \oplus z^{48}} (1 \oplus z^{16}) \Delta_2' \tag{6}$$

where

$$G(z) = F(z)(I \oplus z^4 R_{32}) = (1 \oplus z \oplus z^2 \oplus z^3 \oplus z^4)^4 I \oplus z^{20} R_{32} \tag{7}$$

denotes the 64×64 matrix whose coefficients are polynomials in z of degree at most 20, and

$$\Delta_1' = G(z)\Delta_1 \oplus z^{10} F(z) \Delta_2 \\ \Delta_2' = F(z)\left(z^6 \Delta_1 \oplus (1 \oplus z^4)\Delta_2\right)$$

are 64-dimensional vectors (64×1 matrices) whose elements are polynomials in z defined by the initial state of the buffer and whose degrees are at most 31.

3.4 Properties

Consequently, both b_4 and b_{10} have two components, one being a linear transform of the input sequence a_0 and the other being a linear transform of the initial conditions contained in Δ_1 and Δ_2. For both b_4 and b_{10}, the other, intrinsic component consists of 64 binary linear recurring subsequences produced by the LFSR with the feedback polynomial $f(z)$, or equivalently, by the cycling LFSR with the feedback polynomial $1 \oplus z^{48}$. Therefore, the period of each of these binary subsequences is equal to 48 or divides 48.

- *This period is unacceptably small for cryptographic applications, especially in view of the fact that these subsequences depend only on the secret key and not on the initialization vector.*
- *In common designs of keystream generators, the linear component, with the feedback from the nonlinear component disconnected, normally ensures a large period of the corresponding internal state sequence which itself very likely provides a lower bound on the period of the keystream sequence. This criterion is not satisfied here.*
- *Another weakness is that the degree of $f(z)$ is only 32, and with an appropriate design it could have been as large as $16 \times 64 = 2^{10}$, which is the size of the internal state of the buffer.*
- *The polynomial $f(z)$ also defines a sequential linear transform of the buffer sequence that is equal to a sequential linear transform of the input sequence coming from the nonlinear component. Its low degree and small period facilitate the initial state reconstruction and finding statistical distinguishers for the keystream sequence (see Sections 4-6).*

Yet another interesting property to analyze is the dependence of the intrinsic binary linear recurring subsequences upon the initial conditions. A careful analysis reveals (details are omitted) that for both b_4 and b_{10} each such subsequence depends on only 32 bits of the initial state of the buffer. More precisely, for both b_4 and b_{10} and for any $1 \leq j \leq 64$, the j-th binary subsequence depends on the j-th and the $(j+32)_{64}$-th binary subsequences of $b^{(0)}$, that is, on $\left(b_{i,j}^{(0)}\right)_{i=0}^{15}$ $\left(b_{i,(j+32)_{64}}^{(0)}\right)_{i=0}^{15}$. (Here the subscript 64 denotes the residue mod 64.)

- *This means that mixing between different binary subsequences in the buffer, provided by the linear transform R_{32}, is not good.*

In addition, for both b_4 and b_{10} and for any $1 \le j \le 64$, the j-th binary subsequence depends on the j-th and the $(j+32)_{64}$-th binary subsequences of the input 64-bit sequence a_0.

4. Elimination of Buffer

The obtained expressions (5) and (6) for the generating functions of the 64-bit buffer sequences b_4 and b_{10} can be transformed into the time domain and then appropriately substituted in the recurrences (1) for the update function ρ. In this way, we can derive the recurrences involving only the state sequences a_1 and a_2, where the output sequence a_2 is assumed to be known, in the known-plaintext scenario. Namely, from (1) we first eliminate a_0 and use the fact that F is invertible to get for $t \ge 0$

$$a_1^{(t)} \oplus b_4^{(t)} = F^{-1}(a_1^{(t+1)} \oplus a_2^{(t)} \oplus C_1) \tag{8}$$
$$a_1^{(t)} \oplus^{<17} b_{10}^{(t)} = F^{-1}(a_1^{(t-1)} \oplus a_2^{(t+1)} \oplus C_2) \tag{9}$$

where F^{-1} is the inverse of F and formally $a_1^{(-1)} = a_0^{(0)}$. Now by converting (5) and (6) into the time domain we get the following linear recurrences holding for $t \ge 48$:

$$\begin{aligned} b_4^{(t)} \oplus b_4^{(t-48)} = {} & a_0^{(t-5)} \oplus a_0^{(t-9)} \oplus a_0^{(t-13)} \oplus a_0^{(t-17)} \oplus a_0^{(t-25)} \oplus a_0^{(t-29)} \oplus a_0^{(t-33)} \oplus a_0^{(t-37)} \\ & \oplus^{<32} a_0^{(t-25)} \oplus^{<32} a_0^{(t-41)} \end{aligned} \tag{10}$$

$$\begin{aligned} b_{10}^{(t)} \oplus b_{10}^{(t-48)} = {} & a_0^{(t-11)} \oplus a_0^{(t-15)} \oplus a_0^{(t-31)} \oplus a_0^{(t-43)} \\ & \oplus^{<32} a_0^{(t-15)} \oplus^{<32} a_0^{(t-19)} \oplus^{<32} a_0^{(t-31)} \oplus^{<32} a_0^{(t-35)} \end{aligned}. \tag{11}$$

Finally, by combining (8) with (10) and (9) with (11), we get the following recurrences involving a_1 and a_2 only, which hold for $t \ge 48$:

$$\begin{aligned} & a_1^{(t)} \oplus a_1^{(t-48)} \oplus F^{-1}(a_1^{(t+1)} \oplus a_2^{(t)} \oplus C_1) \oplus F^{-1}(a_1^{(t-47)} \oplus a_2^{(t-48)} \oplus C_1) = \\ & \quad a_1^{(t-6)} \oplus a_1^{(t-10)} \oplus a_1^{(t-14)} \oplus a_1^{(t-18)} \oplus a_1^{(t-26)} \oplus a_1^{(t-30)} \oplus a_1^{(t-34)} \oplus a_1^{(t-38)} \oplus^{<32} a_1^{(t-26)} \oplus^{<32} a_1^{(t-42)} \end{aligned} \tag{12}$$

$$\begin{aligned} & a_1^{(t)} \oplus a_1^{(t-48)} \oplus F^{-1}(a_1^{(t-1)} \oplus a_2^{(t+1)} \oplus C_2) \oplus F^{-1}(a_1^{(t-49)} \oplus a_2^{(t-47)} \oplus C_2) = \\ & \quad {}^{<17} a_1^{(t-12)} \oplus^{<17} a_1^{(t-16)} \oplus^{<17} a_1^{(t-32)} \oplus^{<17} a_1^{(t-44)} \oplus^{<49} a_1^{(t-16)} \oplus^{<49} a_1^{(t-20)} \oplus^{<49} a_1^{(t-32)} \oplus^{<49} a_1^{(t-36)} \end{aligned}. \tag{13}$$

The recurrences are nonlinear because of nonlinear F^{-1}. As the first 16 outputs $\left(a_2^{(t)}\right)_{t=0}^{15}$ are not known, it is interesting to consider (12) and (13) only for $t \ge 64$.

One can also obtain different recurrences by using the polynomial $f(z)$ of degree 32 instead of the polynomial $1 \oplus z^{48}$. They will hold for $t \geq 32$, but each will involve F^{-1} three times which makes them less useful.

In principle, there are two general ways of using (12) and (13).

- One is to try to eliminate a_1 from these two recurrences, possibly with certain approximation probabilities, thus yielding a recurrence in a_2 holding with a certain probability which will represent a statistical distinguisher between the keystream sequence and a purely random sequence (i.e., a sequence of mutually independent uniformly distributed random variables).
- The other is to assume that a_2 is known, in the known-plaintext scenario, and try to solve the corresponding nonlinear equations (or their approximations) for $a_1^{(t)}$ for particular t, e.g., for $t = 15$. This might open the door for a further attack targeting the secret key.

Both ways are essentially addressed in the following sections.

5. Linear Cryptanalysis

Linear cryptanalysis of stream ciphers is essentially different from linear cryptanalysis of block ciphers because of the underlying iterative structure in which the initial state is unknown, whereas the output sequence is assumed to be known in the known-plaintext scenario. A general way of conducting the linear cryptanalysis of stream ciphers is to linearize the next-state and output functions, with certain approximation probabilities, and to analyze the LFSM resulting from these linear approximations (see [G92, G94, G96a, G96b]). The obtained LFSM is in fact a LFSM approximation of the keystream generator, which itself is a nonlinear FSM. It can be analyzed with respect to the following two general objectives [G94]:

- to reconstruct the secret key, in particular, the initial state of the keystream generator
- to derive a linear statistical distinguisher which can distinguish the keystream sequence from a purely random sequence.

Linearizing the next-state function of MUGI reduces to linearizing the nonlinear function F or its inverse F^{-1}. More precisely, we will linearize equations (8) and (9), where the sequences b_4 and b_{10} are determined by (5) and (6). In turn, linearizing F^{-1} reduces to linearizing the S-boxes of AES. The effectiveness of the linear cryptanalysis depends on the way this linearization is performed and on the underlying approximation probabilities.

5.1 Linear Approximations for F

Our objective in this section is to derive linear approximations to F or F^{-1}. In particular, especially effective are the linear approximations involving only one active S-box, as the underlying approximation probability is then most different from one half.

First note that the 64×64-bit F can be divided into two separate 32×32-bit functions G, where G is a composition of S-boxes and the linear MixColumn transformation. An S-box is a composition of the multiplicative inversion in GF(256), with 0 mapped to 0, and an invertible affine transformation. Linearizing G then consists of linearizing the S-boxes and of linearly transforming the obtained linear approximations by the MixColumn transformation. If we want only one S-box to be active, then we have to find a linear approximation to any linear combination of 8 input bits for any chosen S-box and then to express the found linear combination of 8 output bits of this S-box in terms of 32 output bits of G by using the linear MixColumn transformation. Effectively, we thus linearize the inverse function G^{-1}.

What remains to be examined is how to find linear approximations for individual S-boxes. The correlation between a linear input function α and a linear output function β of an S-box can be measured by the correlation coefficient

$$c(\alpha,\beta) = \Pr(\alpha = \beta) - \Pr(\alpha \neq \beta).$$

It is well known that the maximal correlation coefficient magnitude is $1/8$. We examined the linear approximations by computer simulations, that is, we computed the correlation coefficients for every $(\alpha,\beta) \neq (0,0)$. Table 1 displays the number of β correlated to any given α with a given correlation coefficient magnitude. The same table is valid for the inverse S-box.

	5	16	36	24	34	40	36	48	16
	8/64	7/64	6/64	5/64	4/64	3/64	2/64	1/64	0

Table 1: Distribution of correlation coefficient magnitudes for an S-box.

We also considered linear approximations for the whole G^{-1} by taking into account the MixColumn transformation. For each α involving only the 8 input bits to any individual S-box, we thus determined all the corresponding β, involving the corresponding 32 output bits of the MixColumn transformation. An interesting conclusion is that each pair (α,β) such that $|c(\alpha,\beta)| =$ 8/64 or 7/64 involves at least 10 input and output bits. In addition, it is interesting to note that for each α involving the 16 input bits to any pair of S-boxes, each pair (α,β) such that $|c(\alpha,\beta)|$ is close to being maximal, $(8/64)^2$, involves at least 5 input and output bits. However, observe that the correlation coefficient reduced because of two S-boxes being active.

5.2 LFSM Approximations for MUGI

We will use equations (8) and (9) in which, for convenience, $t-1$ is substituted for t. A basic way of linearization is to find linear approximations to 64 component Boolean functions of F^{-1}. In this case linearization is performed by substituting a 64×64-bit vectorial linear function for F^{-1} in (8) and (9). A more general way is to find linear approximations to some 64 linearly independent linear combinations of 64 component Boolean functions of F^{-1}. In this case linearization is performed by applying an invertible matrix L_1 to the left-hand sides of (8) and (9) and by substituting a matrix L_2 for F^{-1} on the right-hand sides of (8) and (9). Namely, we thus get for $t \geq 1$

$$L_1 a_1^{(t-1)} \oplus L_1 b_4^{(t-1)} = L_2 a_1^{(t)} \oplus L_2 a_2^{(t-1)} \oplus L_2 C_1 \oplus e_1^{(t)} \tag{14}$$

$$L_1 a_1^{(t-1)} \oplus L_1{}^{<17} b_{10}^{(t-1)} = L_2 a_1^{(t-2)} \oplus L_2 a_2^{(t)} \oplus L_2 C_2 \oplus e_2^{(t)} \tag{15}$$

where e_1 and e_2 are the 64-bit approximation-error sequences whose binary component subsequences are expressed as nonbalanced Boolean functions of the corresponding inputs to F^{-1}. The more nonbalanced these functions, the better the underlying linear approximations to L_1F^{-1}. However, we will see that the effectiveness of the linear cryptanalysis does not depend only on that. The linear approximation L_2 to L_1F^{-1} and the corresponding correlation coefficients are obtained by using the linear approximations to S-boxes as explained in Section 5.1.

5.2.1 Basic Equation

Equations (14) and (15) in fact define an LFSM with input sequences e_1 and e_2. The LFSM can be solved for a_1 and a_2 by using the generating function method already applied in Section 3. Let

$$A_1 = \sum_{t=0}^{\infty} a_1^{(t)} z^t\ ,\ A_2 = \sum_{t=0}^{\infty} a_2^{(t)} z^t\ ,\ E_1 = \sum_{t=1}^{\infty} e_1^{(t)} z^t\ \text{, and}\ E_2 = \sum_{t=1}^{\infty} e_2^{(t)} z^t$$

denote the generating functions of a_1, a_2, e_1, and e_2, respectively. Then we get

$$zL_1\sum_{t=1}^{\infty} a_1^{(t-1)} z^{t-1} \oplus zL_1\sum_{t=1}^{\infty} b_4^{(t-1)} z^{t-1} = L_2\sum_{t=1}^{\infty} a_1^{(t)} z^t \oplus zL_2\sum_{t=1}^{\infty} a_2^{(t-1)} z^{t-1} \oplus \frac{z}{1\oplus z} L_2C_1 \oplus \sum_{t=1}^{\infty} e_1^{(t)} z^t$$
$$zL_1\sum_{t=1}^{\infty} a_1^{(t-1)} z^{t-1} \oplus zL_1R_{17}\sum_{t=1}^{\infty} b_{10}^{(t-1)} z^{t-1} = z^2L_2\sum_{t=1}^{\infty} a_1^{(t-2)} z^{t-2} \oplus L_2\sum_{t=1}^{\infty} a_2^{(t)} z^t \oplus \frac{z}{1\oplus z} L_2C_2 \oplus \sum_{t=1}^{\infty} e_2^{(t)} z^t$$

and accordingly

$$zL_1A_1 \oplus zL_1B_4 = L_2A_1 \oplus zL_2A_2 \oplus E_1 \oplus \frac{z}{1\oplus z} L_2C_1 \oplus L_2a_1^{(0)}$$
$$zL_1A_1 \oplus zL_1R_{17}B_{10} = z^2L_2A_1 \oplus L_2A_2 \oplus E_2 \oplus \frac{z}{1\oplus z} L_2C_2 \oplus zL_2a_0^{(0)} \oplus L_2a_2^{(0)}$$

in view of $a_1^{(-1)} = a_0^{(0)}$. By rearranging the terms we obtain

$$(zL_1 \oplus L_2)A_1 \oplus zL_2A_2 = zL_1B_4 \oplus E_1 \oplus \frac{z}{1\oplus z} L_2C_1 \oplus L_2a_1^{(0)}$$
$$z(L_1 \oplus zL_2)A_1 \oplus L_2A_2 = zL_1R_{17}B_{10} \oplus E_2 \oplus \frac{z}{1\oplus z} L_2C_2 \oplus zL_2a_0^{(0)} \oplus L_2a_2^{(0)}$$

where B_4 and B_{10} are determined in terms of A_0 by (5) and (6), respectively, and

$$A_0 = zA_1 \oplus a_0^{(0)}\ .$$

For simplicity, let C_1' and C_2' denote the generating functions of the constant sequences, of period equal to 1, corresponding to the constants C_1 and C_2, respectively. Altogether, we finally obtain

$$\begin{aligned}&\left(f(z)(zL_1 \oplus L_2) \oplus z^7 L_1 G(z)\right) A_1 = \\ &\qquad zf(z)L_2A_2 \oplus f(z)E_1 \oplus zL_1\Delta_1' \oplus z^6 L_1 G(z) a_0^{(0)} \oplus f(z)(L_2 a_1^{(0)} \oplus C_1')\end{aligned} \tag{16}$$

$$\begin{aligned}&z\left(f(z)(L_1 \oplus zL_2) \oplus z^{12} L_1 R_{17} F(z)\right) A_1 = \\ &\qquad f(z)L_2A_2 \oplus f(z)E_2 \oplus zL_1R_{17}\Delta_2' \oplus z(z^{11}L_1R_{17}F(z) \oplus f(z)L_2)a_0^{(0)} \oplus f(z)(L_2a_2^{(0)} \oplus C_2')\end{aligned}\,. \tag{17}$$

Here E_1 and E_2 are not known. However, E_1 and E_2 are generating functions of nonbalanced sequences and as such in fact make (16) and (17) a system of binary linear recurrences each holding with a probability different from one half. Therefore, it is desirable that $f(z)$ has as few terms as possible. This is why it is better to use its polynomial multiple $1 \oplus z^{48}$ with only two terms. Consequently, (16) and (17) can be written in a different form where $f(z)$ is replaced by $1 \oplus z^{48}$ and all the other terms are multiplied by $1 \oplus z^{16}$, because $1 \oplus z^{48} = f(z)(1 \oplus z^{16})$. We thus get

$$F_1(z)\, A_1 = z(1 \oplus z^{48})L_2A_2 \oplus (1 \oplus z^{48})E_1 \oplus \Delta_1'' \oplus (1 \oplus z^{48})C_1' \tag{18}$$

$$F_2(z)\, A_1 = (1 \oplus z^{48})L_2A_2 \oplus (1 \oplus z^{48})E_2 \oplus \Delta_2'' \oplus (1 \oplus z^{48})C_2' \tag{19}$$

where we introduced the following abbreviated notation:

$$\begin{aligned}F_1(z) &= (1 \oplus z^{48})(zL_1 \oplus L_2) \oplus z^7(1 \oplus z^{16})L_1G(z) \\ &= (1 \oplus z^{16})\left(f(z)(zL_1 \oplus L_2) \oplus z^7 L_1 G(z)\right)\end{aligned}$$

$$\begin{aligned}F_2(z) &= z\left((1 \oplus z^{48})(L_1 \oplus zL_2) \oplus z^{12}(1 \oplus z^{16})L_1R_{17}F(z)\right) \\ &= z(1 \oplus z^{16})\left(f(z)(L_1 \oplus zL_2) \oplus z^{12}L_1R_{17}F(z)\right)\end{aligned}$$

$$\Delta_1'' = (1 \oplus z^{16})(zL_1\Delta_1' \oplus z^6 L_1 G(z) a_0^{(0)}) \oplus (1 \oplus z^{48})L_2 a_1^{(0)}$$

$$\Delta_2'' = z(1 \oplus z^{16})L_1R_{17}\Delta_2' \oplus z\left(z^{11}(1 \oplus z^{16})L_1R_{17}F(z) \oplus (1 \oplus z^{48})L_2\right)a_0^{(0)} \oplus (1 \oplus z^{48})L_2 a_2^{(0)}\,.$$

The matrices $F_1(z)$ and $F_2(z)$ depend on the performed linearization and and it is very likely that at least one of them is invertible, because of L_1 being invertible. Note that Δ_1'' and Δ_2'' are 64-dimensional vectors whose elements are polynomials in z defined by the initial state of the whole keystream generator ($b^{(0)}$ and $a_0^{(0)}a_1^{(0)}a_2^{(0)}$) and whose degrees are at most 48.

The objective is to eliminate unknown A_1 from (18) and (19). If $F_1(z)$ is invertible, then we have $F_1(z)^{-1} = \det F_1(z) \cdot F_1(z)^*$, where $F_1(z)^*$ is the adjunct matrix of $F_1(z)$, and accordingly obtain

$$\begin{aligned}&F_2(z)\, F_1(z)^*\Big(z(1 \oplus z^{48})L_2A_2 \oplus (1 \oplus z^{48})E_1 \oplus \Delta_1'' \oplus (1 \oplus z^{48})C_1'\Big) = \\ &\qquad \det F_1(z)\Big((1 \oplus z^{48})L_2A_2 \oplus (1 \oplus z^{48})E_2 \oplus \Delta_2'' \oplus (1 \oplus z^{48})C_2'\Big)\end{aligned}\,.$$

By rearranging the terms we finally get the basic equation:

$$(1\oplus z^{48})\left(zF_2(z)\,F_1(z)^*\oplus\det F_1(z)\,I\right)L_2A_2\oplus(1\oplus z^{48})\left(F_2(z)\,F_1(z)^*C_1'\oplus\det F_1(z)\,C_2'\right)=$$
$$(1\oplus z^{48})\left(F_2(z)\,F_1(z)^*E_1\oplus\det F_1(z)\,E_2\right)\oplus F_2(z)\,F_1(z)^*\Delta_1''\oplus\det F_1(z)\,\Delta_2''\ . \quad (20)$$

If, as above, $t=0$ is taken as the initial time, then the first 16 elements of the output sequence, $\left(a_2^{(t)}\right)_{t=0}^{15}$, are unknown, but the initial state of the buffer, $b^{(0)}$, represented by Δ_1 and Δ_2, depends on the secret key only. Alternatively, if $t=16$ is taken as the initial time, then the output sequence, represented by A_2, is known, but the initial state of the buffer (i.e., $b^{(16)}$) depends on the initialization vector as well. The choice can influence the analysis whose objective is to reconstruct the secret key from the known output sequence for a number of initialization vectors.

5.2.2 Initial State Reconstruction

Let us put the basic equation (20) into the following form:

$$\frac{F_2(z)\,F_1(z)^*\Delta_1''\oplus\det F_1(z)\,\Delta_2''}{1\oplus z^{48}}=F_2(z)\,F_1(z)^*E_1\oplus\det F_1(z)\,E_2\oplus$$
$$\left(zF_2(z)\,F_1(z)^*\oplus\det F_1(z)\,I\right)L_2A_2\oplus\left(F_2(z)\,F_1(z)^*C_1'\oplus\det F_1(z)\,C_2'\right)$$

(21)

and let us take $t=16$ as the initial time. Then Δ_1'' and Δ_2'' are in fact determined by the initial conditions $b^{(16)}$ and $a_0^{(16)}a_1^{(16)}a_2^{(16)}$ where $a_2^{(16)}$ is known. The effective number of binary unknowns is thus $18\cdot 64=1152$.

It is important to note that if we are able to reconstruct the state $b^{(16)}$ *and* $a_0^{(16)}a_1^{(16)}a_2^{(16)}$ *at time* $t=16$*, then we can reconstruct the initial state* $b^{(0)}$ *and* $a_0^{(0)}a_1^{(0)}a_2^{(0)}$ *simply by reversing the equations for the next-state function of* MUGI *even if the output sequence is unknown (as is the case in this situation).* More precisely, the next-state function can be reversed in the following way:

$$\begin{aligned}
&b_i^{(t)}=b_{i+1}^{(t+1)}, \qquad i\neq 3,9,15\\
&a_1^{(t)}=a_0^{(t+1)}\\
&a_0^{(t)}=a_2^{(t+1)}\oplus F(a_1^{(t)}\oplus^{<17}b_{10}^{(t)})\oplus C_2\\
&a_2^{(t)}=a_1^{(t+1)}\oplus F(a_1^{(t)}\oplus b_4^{(t)})\oplus C_1\\
&b_3^{(t)}=b_4^{(t+1)}\oplus b_7^{(t)}\\
&b_9^{(t)}=b_{10}^{(t+1)}\oplus^{<32}b_{13}^{(t)}\\
&b_{15}^{(t)}=b_0^{(t+1)}\oplus a_0^{(t)}\,.
\end{aligned}$$

Also, the 128-bit secret key can be directly obtained from $b_0^{(0)}b_1^{(0)}$ by reversing the update equations for ρ . Accordingly, reconstructing the secret key is not more difficult than reconstructing $b_0^{(0)}b_1^{(0)}$ and, altogether, than reconstructing the internal state $b^{(16)}$ and *$a_0^{(16)}a_1^{(16)}$. In fact, this should be regarded as a weakness of the initialization algorithm.*

In the time domain, the left-hand side of (21) is a 64-bit sequence, x, denoted as X in the generating function domain, which depends on the initial conditions, and the last two terms on the right-hand side of (21) are linear transforms of the known sequence a_2 and of the constant sequences corresponding to C_1' and C_2' , respectively. The first term on the right-hand side of (21) is the noise term depending on the performed linear approximations. Consequently, (21) means that the linear recurring sequence x depending on the initial conditions which is ultimately periodic with period of only 48 is termwise correlated to a sequential linear transform of a_2 and of the constant sequences corresponding to C_1' and C_2' . More precisely, this is the case for each of the 64 constituent binary subsequences. The effectiveness of the correlation equation (21) is determined by how much the probabilities for the 64 underlying binary noise subsequences deviate from one half, and the corresponding correlation coefficients can be positive or negative.

Accordingly, the periodic part of the 64-bit linear recurring sequence x , that is, the corresponding 48 64-bit words can in principle be reconstructed by a sort of a fast correlation attack. These 48 64-bit words in fact define a system of $48\cdot 64$ binary equations among the unknown $17\cdot 64$ initial state bits, $b^{(16)}$ and $a_0^{(16)}$, which can thus be obtained by solving the system. This is because $a_1^{(16)}$ does not affect the periodic part of the sequence x, due to

$$X = \frac{(1\oplus z^{16})F_2(z)F_1(z)^*\left(zL_1\Delta_1' \oplus z^6 L_1 G(z) a_0^{(0)}\right)}{1\oplus z^{48}} \oplus F_2(z)F_1(z)^* L_2 a_1^{(0)} \oplus$$

$$\frac{\det F_1(z)\left(z(1\oplus z^{16})L_1R_{17}\Delta_2' \oplus z^{12}(1\oplus z^{16})L_1R_{17}F(z)\, a_0^{(0)}\right)}{1\oplus z^{48}} \oplus \det F_1(z)\,(zL_2a_0^{(0)} \oplus L_2a_2^{(0)})$$

where the periodic part depends only on $a_0^{(0)}$, and not on $a_1^{(0)}$. To get the whole initial state, the 64-bit part $a_1^{(16)}$ has to be guessed and this can be achieved in 2^{64} steps.

What facilitates the attack is that the period of the sequence x is only 48, so that the required low-weight parity checks are easily obtained by considering the sequence at times being the integer multiples of 48. More precisely, to reconstruct a bit of a binary constituent subsequence of x , we need $O(48\,c^{-2})$ bits of the corresponding binary output subsequence and the complexity is $O(c^{-2})$. The reconstructed bit value is simply obtained by the majority count. Here $c = 1-2p$ denotes the (positive or negative) correlation coefficient of the corresponding binary noise subsequence where p is the probability that the noise bit is equal to 1. *In order to estimate c we will use a well-known fact that the correlation coefficient of a binary sum of mutually independent binary random variables is equal to the product of their individual correlation coefficients.*

So, the feasibility of the attack depends on the correlation coefficients c for the underlying 64 binary noise subsequences. Recall that the generating function of the 64-bit noise sequence is given by the linear transform

$$E = F_2(z)\, F_1(z)^* E_1 \oplus \det F_1(z)\, E_2 . \qquad (22)$$

The correlation coefficients of the binary noise subsequences of e_1 and e_2 depend on the linearization of the S-boxes and their magnitudes are equal to 2^{-3} or are close to this value (see Section 5.1). The underlying probabilistic assumption is that the noise subsequences are mutually independent sequences of mutually independent and uniformly distributed binary random variables. The correlation coefficient magnitude of the i -th constituent binary subsequence of the resulting noise sequence e, defined by (22), is then given as $|c| = 2^{-3m}$ where m (depending on i) denotes the total number of binary terms from e_1 and e_2 present in this subsequence or, equivalently, the total number of nonzero binary coefficients of the (involved) polynomials in the i -th row of the matrix $F_2(z)\, F_1(z)^*$ and in the polynomial $\det F_1(z)$. In turn, this depends on the linearization of F^{-1}, that is, on the properties of the S-boxes and the linear MixColumn transformation.

In theory, the attack would be effective only if the total complexity is faster than the exhaustive search over the initial states, that is, if $18 \cdot 64 \cdot 2^{6m} \leq 18 \cdot 2^{18 \cdot 64}$, that is, if $m \leq 191$. Note that the exhaustive search requires 18 64-bit output values to be produced. To be on the conservative side, it is here assumed that one round of MUGI (i.e., producing one 64-bit output value) has the same complexity as one elementary operation in the described attack. *It seems that such linearizations of F^{-1} are likely to exist, but the problem may be to find them.* In practice, since the secret key of MUGI has only 128 bits, the attack would be effective if $18 \cdot 64 \cdot 2^{6m} \leq 18 \cdot 2^{128}$, that is, if $m \leq 20$. *Such linearizations of F^{-1} are very unlikely to exist.* This may be related to the diffusion properties of the linear MixColumn transformation and is an interesting topic for further investigations.

5.2.3 Linear Statistical Weakness

The basic equation (20) can also be put into the form

$$\begin{aligned} L(z) A_2 = {} & (1 \oplus z^{48}) E \oplus \\ & (1 \oplus z^{48}) \left(F_2(z)\, F_1(z)^* C_1' \oplus \det F_1(z)\, C_2' \right) \oplus F_2(z)\, F_1(z)^* \Delta_1'' \oplus \det F_1(z)\, \Delta_2'' \end{aligned} \qquad (23)$$

where the matrix

$$L(z) = (1 \oplus z^{48}) \left(z F_2(z)\, F_1(z)^* \oplus \det F_1(z)\, I \right)$$

defines a sequential linear transform of the output sequence a_2. This equation specifies a linear statistical distinguisher between the output sequence and a purely random sequence. Namely, all the terms on the right-hand side of (23) except the noise term are polynomials in z and as such vanish in the time domain after a sufficiently large t depending on the degrees of polynomials in $F_2(z)\, F_1(z)^*$ and $\det F_1(z)$. So, (23) means that a linear transform of the output sequence is termwise correlated to the all-zero 64-bit sequence where the approximation/correlation noise is defined by $(1 \oplus z^{48})E$. Equivalently, the 64 constituent binary subsequences, obtained as linear transforms of the output sequence, are bitwise correlated to the all-zero binary sequence, where the corresponding correlation coefficients can be approximated as squares of the correlation coefficients of the corresponding binary noise subsequences of e. If c^2 is such a correlation coefficient, then the output sequence length required for detecting the weakness in the corresponding binary subsequence is $O(c^{-4})$. The output sequence length required to detect the weakness by using all the 64 subsequences is then $O(c^{-4})/64$.

The correlation coefficient of the i-th constituent binary noise subsequence is now approximately given as $c^2 = 2^{-6m}$ with the same notation as in Section 5.2.2. Accordingly, in theory, the statistical distinguisher would be effective if the total required output sequence length (proportional to the complexity) is smaller than the expected period for the size of the internal state, that is, if $2^{12m} \leq 64 \cdot 2^{18 \cdot 64}$, i.e., if $m \leq 96$. *It seems that such linearizations of F^{-1} may exist.* In practice, since the secret key of MUGI has only 128 bits, the attack would be effective if $2^{12m} \leq 64 \cdot 2^{128}$, that is, if $m \leq 11$. *Such linearizations of F^{-1} are extremely unlikely to exist.*

5.3 LFSM Approximations for Simplified MUGI

We will now conduct linear cryptanalysis of a simplified MUGI in which the feedback from the nonlinear component (i.e., the 64-bit sequence a_0) to the linear component (i.e., the buffer) is disconnected. The resulting keystream generator then becomes weaker and our objective is to examine its resistance to linear cryptanalysis. It is reasonable to regard the obtained complexity results for the simplified MUGI as lower bounds to the complexity of attacks on the full version of MUGI.

In this case, the buffer sequences b_4 and b_{10} depend only on the initial state of the buffer and are thus determined by (5) and (6) where the first terms corresponding to the input sequence a_0 are removed. They are both periodic with a very small period of 48. It is of separate interest to notice that the nonlinear recurrences (12) and (13) obtained by eliminating b_4 and b_{10} then have a simple form of

$$a_1^{(t)} \oplus a_1^{(t-48)} \oplus F^{-1}(a_1^{(t+1)} \oplus a_2^{(t)} \oplus C_1) \oplus F^{-1}(a_1^{(t-47)} \oplus a_2^{(t-48)} \oplus C_1) = 0$$
$$a_1^{(t)} \oplus a_1^{(t-48)} \oplus F^{-1}(a_1^{(t-1)} \oplus a_2^{(t+1)} \oplus C_2) \oplus F^{-1}(a_1^{(t-49)} \oplus a_2^{(t-47)} \oplus C_2) = 0.$$

The linearization method from Section 5.2 is then essentially the same as linearizing F^{-1} in these expressions. More precisely, in the generating function domain, (16) and (17) become

$$f(z)(zL_1 \oplus L_2)A_1 = zf(z)L_2A_2 \oplus f(z)E_1 \oplus zL_1\Delta_1' \oplus f(z)(L_2a_1^{(0)} \oplus C_1')$$
$$zf(z)(L_1 \oplus zL_2)A_1 = f(z)L_2A_2 \oplus f(z)E_2 \oplus zL_1R_{17}\Delta_2' \oplus zf(z)L_2a_0^{(0)} \oplus f(z)(L_2a_2^{(0)} \oplus C_2').$$

Equivalently, (18) and (19) remain to be true, but with simplified expressions:

$$F_1(z) = (1 \oplus z^{48})(zL_1 \oplus L_2) = (1 \oplus z^{16})f(z)(zL_1 \oplus L_2)$$
$$F_2(z) = z(1 \oplus z^{48})(L_1 \oplus zL_2) = z(1 \oplus z^{16})f(z)(L_1 \oplus zL_2)$$

$$\Delta_1'' = z(1 \oplus z^{16}) L_1 \Delta_1' \oplus (1 \oplus z^{48}) L_2 a_1^{(0)}$$
$$\Delta_2'' = z(1 \oplus z^{16}) L_1 R_{17} \Delta_2' \oplus z(1 \oplus z^{48}) L_2 a_0^{(0)} \oplus (1 \oplus z^{48}) L_2 a_2^{(0)} .$$

Because of L_1 being invertible, both matrices $F_1(z)$ and $F_2(z)$ are invertible. For example, the inverse matrix of $F_1(z)$ is given as

$$F_1(z)^{-1} = \frac{1}{(1 \oplus z^{48}) \det(zI \oplus L_1^{-1} L_2)} (zI \oplus L_1^{-1} L_2)^* L_1^{-1} .$$

Note that $\det(zI \oplus L_1^{-1} L_2)$ is by definition the characteristic polynomial of the matrix $L_1^{-1} L_2$. So, by eliminating unknown A_1 from (18) and (19) we get the same basic equation (2), but with simplified expressions for the involved component terms. The conclusions from Sections 5.2.2 and 5.2.3 remain to be true, but in this case the number, m, of binary terms in the equivalent noise sequence is expected to be smaller, as the coefficients of the matrices $F_1(z)$ and $F_2(z)$ are then polynomials with a smaller number of terms.

It is interesting to analyze a special case when the matrices L_1 and L_2 commute, that is, when $L_1 L_2 = L_2 L_1$. The basic equation (20) then takes a simplified form, which, in fact, can be directly obtained by eliminating A_1 from (18) and (19). Namely, it then follows that the matrices $zL_1 \oplus L_2$ and $L_1 \oplus zL_2$ commute so that we have

$$\begin{aligned} & z(L_1 \oplus zL_2)\left(z(1 \oplus z^{48}) L_2 A_2 \oplus (1 \oplus z^{48}) E_1 \oplus \Delta_1'' \oplus (1 \oplus z^{48}) C_1' \right) = \\ & \qquad (zL_1 \oplus L_2)\left((1 \oplus z^{48}) L_2 A_2 \oplus (1 \oplus z^{48}) E_2 \oplus \Delta_2'' \oplus (1 \oplus z^{48}) C_2' \right) \end{aligned}$$

which by rearranging the terms becomes

$$\begin{aligned} & (1 \oplus z^{48})\left(z^2 (L_1 \oplus zL_2) \oplus (zL_1 \oplus L_2)\right) L_2 A_2 \oplus (1 \oplus z^{48})\left(z(L_1 \oplus zL_2) C_1' \oplus (zL_1 \oplus L_2) C_2' \right) = \\ & \qquad (1 \oplus z^{48})\left(z(L_1 \oplus zL_2) E_1 \oplus (zL_1 \oplus L_2) E_2\right) \oplus z(L_1 \oplus zL_2)\Delta_1'' \oplus (zL_1 \oplus L_2)\Delta_2'' . \end{aligned} \tag{24}$$

The generating function of the 64-bit correlation noise sequence is then given as

$$E = z(L_1 \oplus zL_2) E_1 \oplus (zL_1 \oplus L_2) E_2$$

so that the number, m, of terms in each row of E is explicitly determined by the number of binary terms in each row of L_1 and the number of terms in each row of L_2. The number of terms in a row of L_1 is the number of binary terms in the corresponding linear combination of output bits of F^{-1} and the number of terms in a row of L_2 is the number of binary terms in the corresponding linear combination of input bits of F^{-1}. What is relevant is the sum of the two numbers. The results obtained by computer simulations, reported in Section 5.1, show that this sum can be as small as 10. The total number of terms in the corresponding row of E is then 20. The resulting correlation coefficient magnitude is then $|c| = 2^{-60}$. If linearizations with two active S-boxes are considered, then the total number of terms can reduce to 10, but the overall correlation coefficient magnitude remains the same.

It follows that the initial state reconstruction attack from Section 5.2.2 is then effective in theory and on the borderline to be effective in practice, whereas the linear statistical weakness from Section 5.2.3 is detectable in theory, but not in practice.

6. On Low-Diffusion Statistical Distinguishers

The concept of low-diffusion statistical distinguishers for certain types of stream ciphers is introduced in [CHJ02]. Here we present a more general and more precise treatment of the subject. Consider a general type of keystream generator with the binary internal state vector $(a^{(t)},b^{(t)})$ consisting of two components one of which, $b^{(t)}$, is updated linearly. More precisely, let the component next-state functions and the output function have the following form, respectively:

$$a^{(t+1)} = \rho(a^{(t)} \oplus \mu b^{(t)}) \tag{25}$$

$$b^{(t+1)} = \lambda_a a^{(t)} \oplus \lambda_b b^{(t)} \tag{26}$$

$$z^{(t)} = \eta_a a^{(t)} \oplus \eta_b b^{(t)} \tag{27}$$

where $\mu, \lambda_a, \lambda_b, \eta_a$, and η_b are all linear functions, and ρ is a nonlinear function. In general, the linearly updated component can be solved to yield

$$\Lambda_b b^{(t+1)} = \Lambda_a \lambda_a a^{(t)} \tag{28}$$

where Λ_b and Λ_a are sequential linear transforms and Λ_b is binary, that is, in the generating function domain it is represented by the matrix $p(z)I$ where $p(z)$, $p(0)=1$, is a binary polynomial and I is the identity matrix of appropriate dimension. In this regard, see (10) and (11) for MUGI. As such, Λ_b is invertible and commutes with any sequential linear transform (i.e., $\Lambda_b L = L\Lambda_b$ for every sequential linear transform L). If ρ is invertible, then this can be used to eliminate the linear component from the update equations, namely,

$$\Lambda_b a^{(t)} \oplus \Lambda_b \rho^{-1} a^{(t+1)} = \mu \Lambda_a \lambda_a a^{(t-1)}$$

(see Section 4 for MUGI). This is a nonlinear recurrence for the internal state component $a^{(t)}$, but as $a^{(t)}$ is generally not obserbavle, it does not yield a statistical distinguisher. What is observable in the known-plaintext scenario is the output value $z^{(t)}$. However, (27) generally does not allow $a^{(t)}$ to be expressed in terms of $z^{(t)}$, unless η_a is invertible.

Suppose now that η_a is invertible. In this case, we first get

$$a^{(t)} = \eta_a^{-1} z^{(t)} \oplus \eta_a^{-1} \eta_b b^{(t)}$$

and then in view of (28)

$$\Lambda_b b^{(t+1)} \oplus \Lambda_a \lambda_a \eta_a^{-1} \eta_b b^{(t)} = \Lambda_a \lambda_a \eta_a^{-1} z^{(t)} .$$

This again can be solved to yield

$$\Lambda_b' b^{(t+1)} = \Lambda_a' \Lambda_a \lambda_a \eta_a^{-1} z^{(t)} \tag{29}$$

and then

$$\Lambda_b' a^{(t)} = \Lambda_b' \eta_a^{-1} z^{(t)} \oplus \eta_a^{-1} \eta_b \Lambda_a' \Lambda_a \lambda_a \eta_a^{-1} z^{(t-1)} \tag{30}$$

where Λ_b' and Λ_a' are sequential linear transforms and Λ_b' is binary (its dimension is adjusted to the dimension of the sequence it is applied to). Note that (29) and (30) are linear recurrences, but cannot be solved to directly yield the sequences a and b because of unknown initial conditions. If that was possible, then (25) would directly define a very effective statistical distinguisher. However, in principle (29), (30), and (25) represent a basis for finding statistical distinguishers.

For example, for the so-called low-diffusion statistical distinguisher [CHJ02] we just have to note that we can compute the same sequential linear transform, namely Λ_b', of both the 64-bit input and output sequences to ρ in (25). More precisely, we have

$$\Lambda_b' a^{(t+1)} = \Lambda_b' \eta_a^{-1} z^{(t+1)} \oplus \eta_a^{-1} \eta_b \Lambda_a' \Lambda_a \lambda_a \eta_a^{-1} z^{(t)}$$

$$\Lambda_b' (a^{(t)} \oplus \mu b^{(t+1)}) = (\mu \Lambda_a' \Lambda_a \lambda_a \oplus \Lambda_b') \eta_a^{-1} z^{(t)} \oplus \eta_a^{-1} \eta_b \Lambda_a' \Lambda_a \lambda_a \eta_a^{-1} z^{(t-1)}$$

which means that we can compute, in terms of the known output sequence z, the paired sequence

$$\left(\Lambda_b' u^{(t)},\ \Lambda_b' \rho u^{(t)}\right), \quad \text{where} \quad u^{(t)} = a^{(t)} \oplus \mu b^{(t+1)} \quad \text{and} \quad \rho u^{(t)} = a^{(t+1)} . \tag{31}$$

As Λ_b' is binary, when computed for a given value of t, each pair is in fact a bitwise sum of the current and some previous inputs and outputs to ρ, that is,

$$\left(u^{(t)} \oplus u^{(t-t_1)} \oplus \cdots \oplus u^{(t-t_m)},\ \rho u^{(t)} \oplus \rho u^{(t-t_1)} \oplus \cdots \oplus \rho u^{(t-t_m)}\right). \tag{32}$$

A set of sufficiently many such pairs can in principle be statistically distinguished from a purely random sequence. The complexity predominantly depends on the number of nonzero terms in Λ_b', but also on the bit-size of the input/output to ρ. To this end, it may be desirable to reduce this bit-size by considering linear functions of the input and output to ρ. Namely, instead of $(u^{(t)}, \rho u^{(t)})$ we can consider $(l_{in} u^{(t)}, \rho' l_{in} u^{(t)})$ where $l_{out} \rho u^{(t)} = \rho' l_{in} u^{(t)}$. In a special case of 4 terms only ($m = 3$), an approximation to the probability distribution of the pairs is given in [CHJ02] for a probabilistic model in which the function ρ is assumed to be random. In general, the larger the number of terms, the larger the complexity of the statistical distinguisher.

In the case when there is no feedback to the linearly updated component, we have $\lambda_a = 0$ and $\Lambda_b' = \Lambda_b$, so that the linear transform Λ_b' may in principle have a smaller number of terms than in the case with feedback.

The main point of the described low-diffusion statistical distinguisher is that $a^{(t)}$ can be expressed in terms of $z^{(t)}$, i.e., that η_a is invertible. In the case of MUGI, this assumption is not satisfied and, in fact, a large portion of the 192-bit $a^{(t)}$ remains to be unknown, that is, the 64-bit $a_1^{(t)}$. Accordingly, this statistical distinguisher is not applicable to MUGI. What one can obtain by eliminating the linearly updated component are the nonlinear recurrences (12) and (13) in the unknown 64-bit sequence a_1. However, it has to be noted that the number of terms in Λ_b for MUGI is only 2, because of a bad design of the buffer.

7. On Using Overdefined Equations for S-boxes

The nonlinear part of an S-box in AES realizes the multiplicative inversion in GF(256) and maps zero to zero. As such it can be described by the equation $xy = 1$ for $x \neq 0$. On the basis of this, it is pointed out in [CP02] that one can associate with an S-box 39 linearly independent and relatively sparse quadratic binary equations in 8 input and 8 output binary variables. They hold with probability 1, so they are not approximations. Among them, there are 23 bi-affine equations which do not include products of input or output variables solely.

For AES, one can thus write down a large system of quadratic binary equations obtained by associating 39 or 23 equations with each S-box and by taking into account the key-scheduling algorithm too. The system can be treated by the multiply-and-linearize method to obtain a large overdefined system of linear equations in new variables. The method essentially consists in multiplying the equations by (appropriate) products of binary variables and by replacing the obtained products by new binary variables. The complexity of the method is determined by the number of variables in the resulting system of linear equations. It is shown in [CP02] that AES with 128-bit key is not vulnerable to this method. It is also shown that AES with 256-bit key is on the borderline to be vulnerable, but the complexity estimate is based on the questionable assumption that the resulting linear equations are linearly independent.

Since MUGI is a keystream generator, the system of nonlinear equations is produced in a different way from that of a block cipher like AES. For example, one can consider the system in 18 64-bit initial state variables obtained by assuming that 19 consecutive 64-bit outputs are known. More precisely, the initial state variables are $b^{(0)}$ and $a_0^{(0)}a_1^{(0)}$, whereas $a_2^{(0)}$ and $\left(a_2^{(t)}\right)_{t=1}^{18}$ are assumed to be known. The 128-bit secret key can then easily be obtained from the reconstructed initial state as explained in Section 5.2.2. The ρ update function of MUGI is not the same as the round function of AES, but the two are similar. To get the system, one then has to iterate ρ 18 times, which is much bigger then 10, which is the number of rounds in AES with 128-bit key. The complexity of the multiply-and-linearize method would then very likely be lower than $2^{18\cdot 64}$ but much higher than 2^{128}.

Another way of getting the system would be to write down directly the system in 128 binary secret key variables from the first two or three known 64-bit outputs. One then has to take into account the initialization algorithm which altogether takes 48 iterations of ρ in order to produce the first 64-bit output. Again, in light of [CP02], the complexity of the multiply-and-linearize method would very likely be (much) higher than 2^{128}.

One can of course ask the question if the method from [CP02] can be improved. For example, is it possible to derive sequential variants of the multiply-and-linearize method that would employ some sort of chaining the variables, which may reduce the total number of variables and hence the complexity? This does not appear to be likely mainly because of an interesting property of the quadratic equations associated with an S-box that each of them includes at least one mutual product of an input and an output variable. Because of this, it is difficult to perform chaining (composition) of these equations in iterative structures like in AES or MUGI.

8. Summary of Weaknesses and Strengths

Our main finding is that the linearly updated component of MUGI, the so-called buffer, is not designed properly. We proved that if the feedback from the nonlinearly updated component is disconnected, then the binary subsequences of the buffer are linear recurring sequences with the linear complexity of only 32 and with the period of only 48. This is what can be called the intrinsic response of the buffer. Accordingly, the buffer does not provide a large lower bound on the period of output sequences of MUGI which is normally the case with many designs of keystream generators. Furthermore, as each such subsequence depends on only 32 bits of the initial state of the buffer, the mixing between the $16 \cdot 64$ bits of the initial state of the buffer is not good.

As a consequence of this small period, it is shown that the buffer sequence can easily be eliminated from the update equations for the nonlinearly updated component of MUGI, the so-called state, thus yielding the nonlinear recurrences involving only the output sequence and a part of the state sequence. It is then pointed out that this may facilitate the cryptanalysis of MUGI such as the linear cryptanalysis as well as finding the statistical distinguishers for MUGI.

However, the period of MUGI is not expected to be small, primarily because of a large bit-size of the internal state and because of the 1-1 next-state function, but not because of a good design of the linearly updated component. A theoretical analysis of the period is thus very likely to be intractable.

The linear cryptanalysis of MUGI performed by linearizing the next-state function and by solving the resulting linear finite-state machine shows that MUGI may in theory be vulnerable to the initial state reconstruction attack faster than the exhaustive search over the 1152-bit initial states (the whole internal state has 1216 bits, but 64 bits are taken to the output). This is again a consequence of the bad design of the buffer, but the attack is very unlikely to be faster than the exhaustive search over the 128-bit secret key. This is due to good linear correlation properties of the S-boxes and good diffusion properties of the linear MixColumn transformation, both from AES. Moreover, it is also argued that the linear statistical distinguishers may exist in theory, but not in practice, as their complexity is expected to be much higher than the exhaustive search over the 128-bit secret key. It is also shown that the simplified version of MUGI obtained by removing the feedback to the buffer is more vulnerable to the linear cryptanalysis. So, this feedback is also one of the strengths of MUGI.

It is pointed out that the 128-bit secret key can easily be recovered from the reconstructed internal state of MUGI at any time. This is a weakness of the initialization algorithm, which is itself relatively complex, but does not have the property that it should be infeasible to recover the secret key from the initial state.

An in-depth general analysis of low-diffusion statistical distinguishers is conducted and it is argued that they are not applicable to MUGI, primarily because of the fact that only a part of the whole state vector is taken to the output, so that a large portion of it remains to be unknown.

In addition, it is also argued that MUGI is very unlikely to be vulnerable to the multiply-and-linearize attack based on overdefined systems of quadratic binary equations associated with S-boxes of AES. This is related to the large bit-size of the internal state and a relatively complicated initialization algorithm.

9. References

[CHJ02] D. Coppersmith, S. Halevi, and C. Jutla, "Cryptanalysis of stream ciphers with linear masking," *Cryptology ePrint Archive*, IACR, 2002/020.

[CP02] N. Courtois and J. Pieprzyk, "Cryptanalysis of block ciphers with overdefined systems of equations," *Cryptology ePrint Archive*, IACR, 2002/044.

[AES] J. Daemen and V. Rijmen, "AES Proposal: Rijndael," 1999, available at http://www.nist.gov/aes/.

[G92] J. Golić, "Correlation via linear sequential circuit approximation of combiners with memory," Advances in Cryptology – EUROCRYPT '92, *Lecture Notes in Computer Science*, vol. 658, pp. 124-137, 1993.

[G94] J. Golić, "Linear cryptanalysis of stream ciphers," Fast Software Encryption – FSE '94, *Lecture Notes in Computer Science*, vol. 1008, pp. 154-169, 1995.

[G96a] J. Golić, "Correlation properties of a general combiner with memory," *Journal of Cryptology*, vol. 9, pp. 111-126, 1996.

[G96b] J. Golić, "Linear models for keystream generators," *IEEE Transactions on Computers*, vol. 45, pp. 41-49, 1996.

[SpM] D. Watanabe, S. Furuya, H. Yoshida, and K. Takaragi, MUGI Pseudorandom number generator, Specification, Ver. 1.2, 2001, available at http://www.sdl.hitachi.co.jp/crypto/mugi/index-e.html.

[EvM] D. Watanabe, S. Furuya, H. Yoshida, and K. Takaragi, MUGI Pseudorandom number generator, Self-evaluation report, Ver. 1.1, 2001, available at http://www.sdl.hitachi.co.jp/crypto/mugi/index-e.html.