Report on Security Evaluation of MUGI Stream Cipher

Dr. Miodrag J. Mihaljević
Research Professor at
Mathematical Institute, Serbian Academy of Sciences and Arts
Kneza Mihaila 35, Belgrade, Yugoslavia
E-mail: miodragm@turing.mi.sanu.ac.yu

August 1, 2002

Abstract

Taking into account all the previously reported results, in this report a number of novel elements relevant for security evaluation of MUGI keystream generator are given. Because the previously reported results do not contradict the statement that MUGI keystream generator is resistant against presently known general methods for cryptanalysis, and because resistance of simplified MUGI keystream generators is not enough considered yet, this report focuses its attention towards methods for cryptanalysis of MUGI-like keystream generators with an intention to yield more insight of MUGI security.

Main results of this report are the following: (i) discussion of the reported results on security evaluation of MUGI, as well as a recently proposed approach for distinguishing MUGI output from the random streams; (ii) development of frameworks for novel cryptanalysis of a simplified and complete MUGI; (iii) more insight into MUGI keystream generator via identification of certain critical issues for MUGI security; (iv) a security comparison of MUGI, PANAMA and MULTI-S01.

Although the proposed method for cryptanalysis of MUGI yields only the framework for cryptanalyis without a definitive answer on its efficiency, the existence of this method is an important alert regarding the security of MUGI.

The features of MUGI algorithm, disclosed in this report, are at least undesirable and potentially dangerous, and accordingly, further security evaluation of MUGI is recommendable.

Report on Security Evaluation of MUGI Stream Cipher

Content

Part I: Introduction and Re-Consideration of the Reported Results

1 Introduction

This report gives a security evaluation of MUGI Stream Cipher according to the following requests from the document no. 14-IPA-505:

- "(1) Investigate the security of MUGI.
  Estimate the security of MUGI by applying various cryptanalytic techniques for pseudorandom number generators. We also welcome the evaluation of simplified (or acceptable modified) versions of MUGI.
  In addition, since the following paper claims that the described attacks can be devised against MUGI, comment on the claim if you have.
  "Cryptanalysis of stream ciphers with linear masking", by D. Coppersmith, S. Halevi and C. Jutla, ePrint IACR, February 16/2002.
  (2) Comparison with MUGI, PANAMA and MULTI-S01.
  Please discuss the superiority of MUGI to the pseudorandom number generator PANAMA and the stream cipher MULTI-S01."

In order to fulfil the above requests, this report contains the following:

1. specification of stream ciphers security evaluation methods relevant for MUGI;
2. analysis of various cryptanalytic techniques relevant for MUGI mainly based on re-consideration of reported results on MUGI security evaluation;
3. proposal of a novel approach relevant for cryptanalysis of a simplified and full MUGI;
4. a security comparison of MUGI, PANAMA and MULTI-S01;
5. concluding statements on MUGI security.

Executive Summary of the Report Results

This report originates from a collection of up to now reported results relevant for security evaluation of MUGI. These results are used for re-evaluation of MUGI resistance against various cryptanalytic attacks.

MUGI is relatively recently proposed to a wider cryptographic community and there are very few publicly reported results related to its security evaluation. The most comprehensive results are these published in [2] and [3].

Taking into account all the previously reported results, in this report a number of novel elements relevant for security evaluation of MUGI keystream generator are given. Because

the previously reported results do not contradict the statement that MUGI keystream generator is resistant against presently known methods for cryptanalysis, and because resistance of simplified MUGI keystream generators is not enough considered yet, this report focuses its attention towards methods for cryptanalysis of MUGI-like keystream generators with the main intention to yield more insight of MUGI security. Following the previous statement and the above requests, security evaluation of MUGI keystream generator is considered starting from an appropriate simplified version of MUGI.

Main results of this report are the following:

- discussion of the relevant reported results on security evaluation of MUGI, as well as a recently proposed approach for distinguishing MUGI output from the random streams;
- development of frameworks for novel cryptanalysis of a simplified and complete MUGI;
- more insight into MUGI keystream generator via identification of certain critical issues for MUGI security;
- a security comparison of MUGI, PANAMA and MULTI-S01.

Although the proposed method for cryptanalysis of MUGI yields only the framework for cryptanalyis without a definitive answer on its efficiency, the existence of this method is an important alert regarding the security of MUGI.

The novel features of MUGI algorithm, disclosed in this report, are at least undesirable and potentially dangerous, and accordingly, further security evaluation of MUGI is recommendable.

Organization of the Report

The report consists of eight sections organized into the following three parts: (i) introduction and consideration of the reported results; (ii) novel approach for cryptanalysis of MUGI; (iii) concluding discussion and references. In Section 2, an overview of MUGI including its design rationale is given. Section 3 contains summary of the evaluation methods. In Section 4 various evaluation approaches of MUGI security are considered based on the reported results. Section 5 yields a novel approach for cryptanalysis of a simplified MUGI. Results of Section 5 are used in Section 6 for consideration of resistance of full MUGI on an algebraic cryptanalytic technique based on a system of overdefined low-degree nonlinear equations. A security comparison of MUGI, PANAMA and MULTI-S01 is given in Section 7. Finally, Section 8 contains concluding statements of this report.

2 MUGI Overview

For the purposes of the report completeness, this section summarize MUGI algorithm according to [1] and [3]. Before this specification, a class of stream ciphers to which MUGI belongs is pointed out.

2.1 On a Class of Stream Ciphers to which MUGI Belongs

Recall that a stream cipher (or pseudorandom generator) is an algorithm that takes a short random string, and expands it into a much longer string, that still "looks random" to adversaries with limited resources. The short input string is called the seed (or key) of the cipher, and the long output string is called the output stream (or key-stream). Stream ciphers can be used for shared-key encryption, by using the output stream as a one-time-pad. Although one could get a pseudorandom generator simply by iterating a block cipher (say, in counter mode), it is believed that one could get higher speeds by using a "special purpose" stream cipher. One approach for designing such fast ciphers, is to use some "non-linear process" that may resemble block cipher design, and to hide this process using linear masking. A plausible rationale behind this design, is that the non-linear process behaves roughly like a block cipher, so we expect its state at two "far away" points in time to be essentially uncorrelated. For close points, on the other hand, it can be argued they are masked by independent parts of the linear process, and so again they should not be correlated. Some examples of ciphers that use this approach include SEAL (see [29], for example) and Scream [27], where the non-linear process is very much like a block cipher, and the output from each step is obtained by adding together the current state of the non-linear process and some entries from fixed (or slowly modified) secret tables. Other examples are PANAMA [18] and MUGI [1], where the linear process (called buffer) is an LFSR (Linear Feedback Shift Register), which is used as input to the non-linear process, rather than to hide the output. Yet another example is SNOW [23], where the linear LFSR is used both as input to the non-linear finite state machine, and also to hide its output.

2.2 MUGI Design Approach

MUGI is a pseudorandom number generator, designed especially for the purpose of beeing the core component of a stream cipher.

MUGI is a PANAMA-like keystream generator (KSG). The principal part of a PANAMA-like KSG is a set (S, T, f) which consists of an internal S, its update function T, and the output filter f which abstracts the output sequence from the internal state S. We call the set (S, T) the internal-state machine. In addition we call a single application of the state update function a round. $S^{(t)}$ refers to the internal state at round t.

In PANAMA, the internal state is divided into two parts, state a and buffer b. The update function of PANAMA depends in a different way on different parts of the internal state. Note that each update function uses another internal state as a parameter. We denote the update functions of state a and buffer b with ρ and λ, respectively.

The noteworthily characteristic of PANAMA's ρ-function is its use of an substitution-permutation network (SPN) structure. For block ciphers (or pseudorandom permutations) there is a de facto standard construction, which uses a Fiestel network or SPN as a component (called round function) and iterates it for mixing. PANAMA uses a core mixing function ρ similar to the round function of a block cipher and a large buffer instead of fixed extended keys and iterations of round function.

On the other hand, the function λ is a simple linear transformation. The output filter

f drops about half of the bits of state a for each round. We call KSG which satisfies such characteristics PANAMA-like keystream generator. This can be formalized in the following definition.

Definition, [3]. Consider an internal-state machine consisting of two internal states, namely the state a, the state b, and their update functions ρ and λ. The keystream generator which consists of an internal-state machine $(a, b), (\rho, \lambda))$ and an output filter f is called PANAMA-like keystream generator if it satisfies the following conditions:
(1) ρ includes an SPN transformation that uses parts of buffer b as a parameter, and

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

(2) λ is a linear transformation that uses a part of state a as a parameter,

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

(3) f outputs a part of state a, which is typically no more than 1/2 of the bits of a.

As an interesting issue regarding the components employed in MUGI, note the following: MUGI employs the same substitution table S-box and the linear transformation as in AES.

2.3 MUGI Specification

MUGI is a KSG or pseudo random number generator (PRNG) with an 128-bit secret key K (secret parameter) and an 128-bit initial vector I (public parameter). It works on a round-by-round basis and it generates 64-bit length random bit string for each round. This section yields a specification of MUGI relevant for the security evaluation in this report: the in detail specification is given [1].

The structure of MUGI is the following.

- **Input**: Secret key K ,Initial vector I, Output size n (units)
- **Output**: Random number sequence $Out[i]$ $(0 \leq i \leq n)$
- **Algorithm**
  Initialization
  Step 1. Set the secret key K into state a. Then initialize buffer b by ρ.
  Step 2. Add the initial vector I into state a and recalculate state a by ρ.
  Step 3. Mix whole internal state by the update function.
  Random number generation
  Step 4. Run n rounds of update function and output a part of the internal state (64-bit)for each round.

The details of MUGI specification are given below.

Input

MUGI has two input as a parameter. One is an 128-bit secret key K and the other is an 128-bit initial vector I. I is a public parameter. The higher and lower units of K are denoted by K_0 and K_1, respectively. I_0 and I_1 are the corresponding components of I.

Internal State

State

State a consists of 3 units. Each of them is denoted by a_0, a_1, a_2 in rotation, i.e.

$$a = [\text{Higher}] \quad a_0 \mid\mid a_1 \mid\mid a_2 \quad [\text{Lower}]$$

Buffer

The buffer b consists of 16 units. Each of them is denoted by $b_0, ..., b_{15}$ in rotation in the same manner as the state a.

2.3.1 Update Function

The update function of PKSG can be described as a combination of ρ and λ, (the update functions of state a and buffer b) each of which uses another internal state as a parameter. In other words the update function *Update* of whole internal state is described as follows:

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

In the followings we explain ρ and λ of MUGI.

ρ

ρ is the update function of state a. It is a kind of target heavy Feistel structure with two F-functions and uses buffer b as a parameter. The function ρ is described as follows:

$$a_0^{(t+1)} = a_1^{(t)}$$

$$a_1^{(t+1)} = a_2^{(t)} \oplus F(a_1^{(t)}, b_4^{(t)}) \oplus C_1 \ ,$$

$$a_2^{(t+1)} = a^{(t)} \oplus F(a_1^{(t)}, b_{10}^{(t)} <<< 17) \oplus C_2 \ .$$

C_1 and C_2 in the equations above are constants. The F-functions of MUGI reuses the components of AES: S-box and MDS. For detail description of the S-box and MDS should be referred to Section 2.3.4.

λ

The function λ is the update function of the buffer b and it uses a part of state a as a parameter. λ is linear transformation of b and is described as follows:

$$b_j^{(t+1)} = b_{j-1}^{(t)} \ \ (j \neq 0, 4, 10) \ ,$$

$$b_0^{(t+1)} = b_{15}^{(t)} \oplus a_0^{(t)} \ ,$$
$$b_4^{(t+1)} = b_3^{(t)} \oplus b_7^{(t)} \ ,$$
$$b_{10}^{(t+1)} = b_9^{(t)} \oplus (b_{13}^{(t)} <<< 32) \ .$$

2.3.2 Initialization

The initialization of MUGI is divided into three steps. Firstly initialize the buffer b with a secret key K, secondly initialize the state a with an initial vector I, and mix whole internal state at last.

In the first step we set the secret key K into state a as follows:

$a_0 \ = K_0 \ ,$
$a_1 \ = K_1 \ ,$
$a_2 \ = (K_0 <<< 7) \oplus (K_1 >>> 7) \oplus C_0 \ .$

C_0 in above equations is a constant. Then iterate running only ρ and put a part of each $a^{(t)}$ into the buffer b as follows:

$$b_{15-i} = (\rho^{i+1}(a.0))_0 \ .$$

In the second step the mixed state $a(K) = \rho^{16}(a_0, 0)$ and the initial vector I are mixed. I is added to the state a as follows:

$a(K,I)_0 \ = a(K)_0 \oplus I_0 \ ,$
$a(K,I)_1 \ = a(K)_1 \oplus I_1 \ ,$
$a(K,I)_2 \ = a(K)_2 \oplus (I_0 <<< 7) \oplus (I_1 >>> 7) \oplus C_0 \ .$

Then the state a is mixed again by 16 rounds iteration of ρ. So the mixed state a is represented as $\rho^{16}(a(K,I), 0)$. The last step is 16 rounds iteration of whole update function $Update$, i.e.,

$$a^{(1)} = Update^{16}(\rho^{16}(a(K,I), 0), b(K)) \ ,$$

where the notation $b(K)$ in above equations means the buffer b initialized with the secret key K.

2.3.3 Pseudo Random Number Generation (PRNG)

After the initialization, MUGI runs as a pseudorandom number generator (PRNG) and generates 64-bit random number and transform the current internal state into the next one at each round. Denote the output at round t as $Out[t]$. So, the PRNG output sequence is given as:

$$Out[t] = a_2^{(t)} \ , t = 1, 2, ... \ ,$$

Table 1: Time table of MUGI

	Round t	Process	Input	Output
	-49	Inputting Key	K	-
	-48,...,-33	Mixing (by ρ)	-	-
	-32	Inputting IV	I	-
	-31,...,-16	Mixing (by ρ)	-	-
	-15,...,0	Mixing (by $Update$)	-	-
Generating bit strings	1,...	Outputing and Mixing	-	$Out[t]$

Accordingly, note that MUGI outputs the lower 64-bit of state a at the beginning of the round process.

Finally note that the processes from the initialization to the random number generation is described in Table 1.

2.3.4 Components

This section specifies the main components of MUGI. Recall that F-function is the main transformation in PRNG. The F-function performs 1-round SPN structure and consists of the following two main components:

- bytewise substitution, denoted as S-box, and
- 4×4 matrix over $GF(2^8)$, denoted as MDS.

S-box.
The main nonlinear transformation of MUGI is the bytewise substitution S-box same as one in AES. In other words,the substitution given by S-box is the composition of the inverse $x \to x^{-1}$ over GF(2^8)and an affine transformation.

Matrix MDS.
The linear transformation of the F-function is the combination of a 4×4 matrix and a bytewise shuffling. MUGI uses MDS matrix which is the component of AES.

F-function.
The F-functionis performs a key addition during the initialization (the data addition from buffer), a non-linear transformation using the S-box, a linear transformation using MDS matrix M and byte shuffling.

Also, note the following: The S-box and the matrix MDS can be combined in a single table lookups on a 32-bit processor. This allows the fast implementation.

3 Security Evaluation Preliminaries

This section summarizes methodological issues relevant for security evaluation of MUGI.

According to the current criteria (see [6] and [5]), the general methodological issues for security analysis include the following.

Resistance to Cryptanalysis. A stream cipher should be resistant at the relevant security level against to possible cryptanalytic attacks. However, when assessing the relevance of a cryptanalytic attack a number of factors should be included such as the overall complexity of the attack (time and space complexity), and volume and type of data required to mount the attack, for example. Any cryptanalytic attack is based on the assumption that complete structure of a stream cipher is known, and that the only one unknown element is the secret key.

Design Philosophy and Transparency. An important consideration when assessing the security of a stream cipher is the design philosophy and transparency of the design. It is easier to have confidence in the assessment of the security if the design is clear and straightforward, and is based on well-understood mathematical and cryptographic principles.

Strength of Modified Primitives. One common technique to assess the strength of a stream cipher is to assess a modified one, obtained by changing or removing a component of the considered stream cipher. Conclusions about the original stream cipher based on assessment of the modified one have to be carefully considered as the influence may or may not be straightforward.

Testing. The purpose of testing is to highlight anomalies in the operation of a stream cipher that may indicate cryptographic weakness and require further investigation.

The techniques for security evaluation of a stream cipher are based on the following:
• consideration of certain characteristics of the underlying structure of a stream cipher, as well as the keystream sequence itself;
• resistance against the attacks for predicting a keystream, or either recovering or reducing uncertainty of the secret key or the plaintext; these attacks can be:
- general attacks applicable to a class of stream ciphers;
- specialized attacks developed for a particular stream cipher.

It is customary when analysing stream ciphers to consider known plaintext attacks which essentially means that an attacker, a cryptanalyst, knows a large volume of keystream. The cryptanalyst's task is then usually classified in one of the following three ways.

(i) Distinguishing Attack. The cryptanalyst gives a method that allows the keystream of certain length to be distinguished from a "random" sequence of the same length.

(ii) Prediction. The cryptanalyst gives a method that allows the prediction of further keystream elements more accurately than guessing.

(iii) Key Recovery. The cryptanalyst gives a method that recovers the secret key from the keystream sequence.

The techniques used to analyse stream ciphers typically use either mathematical and statistical properties or certain approximations. The security evaluation should include

the main general characteristics of a stream cipher and the attacks to which it should be resistant as given below. A good stream cipher must pass all these consideration, but it is only a necessary request and final security evaluation result should include other aspects: for example specific issues related to a particular stream cipher.

Finally note the following.

1. It is very important to consider specialized attacks dedicated to a particular keystream generator because there are a number of very illustrative examples where certain keystream generator is fully resistant against all general attacks but is breakable by the dedicated attacks.

2. Also, it is very important to consider security of the modified versions of a keystream generator because this consideration can indicate a lot about the structural characteristics and critical points for security of the original keystream generator.

3.1 Attack Models

According to the previous discussion, this section summarizes a number of common attack models. These attack models are considered as the good models to study the security of stream ciphers, but note that they do not cover all possible attacks.

For example, one thing to remember is that stream ciphers (and most encryption algorithms in general) do not provide for message integrity. This must be done by some external algorithm, such as a MAC (Message Authentication Code). Note that it is possible to use the lack of message integrity checks to partially determine some of the plaintext. A MAC is particularly important for binary additive stream ciphers. For this class of ciphers, flipping a bit of the ciphertext will flip the corresponding bit of the plaintext, and only affect that one bit. This can be used by an attacker to change messages. A MAC will prevent this type of attack. This property, that a change in the ciphertext will produce a known predictable change in the plaintext, is called malleability.

3.1.1 Searches over Secret Keys Space

The basic attack against any symmetric key cryptosystem is the brute force attack. In this attack, the attacker keeps guessing what the key is until they guess correctly. In general, one known plaintext, or the ability to recognize a correct plaintext is all that is needed for this attack. However, all good cryptosystems should be designed such that this attack is impractical. For a key of size n, a brute force search would consist of trying all 2^n keys to see which one works. If it is possible to recognize the correct plaintext, then on average the correct key will be found in 2^n.

Recently, a more efficient brute force attack method for stream ciphers based on the time/memory/data tradeoffs is reported in [10]. Time/memory/data tradeoffs are based on a speed up of a brute force search by pre-computing some values and storing them in memory and employment a large data collections. Accordingly, this is a trading of memory and data usage for time to perform the attack.

3.1.2 Distinguishing Attacks

A distinguisher is a probabilistic polynomial-time algorithm A. A takes as input N bits of data, which are either from the real stream cipher or are completely random data. A then has to, in polynomial time, output either Real or Random decision. If this is correct with a non-negligible probability, then this is a good distinguisher. The parameter N indicates how good the distinguisher is. In most cases, larger N will produce a more accurate distinguisher.

A distinction could be made between two different type of distinguisher. The first type of distinguisher is what has already been discussed, and is called a polynomial-time distinguisher.

These two notions are equivalent from the information-theoretical viewpoint, though the difference in bias can be significant.

3.1.3 Key Weaknesses

There are also some more specific models that should be discussed. These models are related to the secret key characteristics. Recalling that the secret key completely determines the output sequence from a keystream generator, in a good keystream generator, each bit of the output should depend on the entire key, and the relationship between the key and a given bit (or set of bits) should be extremely complicated.

Key-Biased Output

The first condition is that every bit of the output is **dependent** on the entire key. This means that changing any single bit of a key should have a 1/2 probability of affecting each bit of the output. When this property holds, then in order to brute force a key, every possible key must be tried, and there will be no relationship between bits of the key and the output. This means that uncertainties of the individual bits of the key multiply when calculating the total number of possible keys.

Let us see what happens when this property does not hold. Assume that the 8 bits of the output is dependent only on the first 8 bits of the key. This means that the first 8 bits of the key can be recovered independently of all other key bits based on the first 8 output bits.

Accordingly, if we first check all possible values of the first 8 bits of the key, we can determine the correct ones by comparing the given output with the outputs generated by to the considered hypothesis. Once, these bits are determined, then the remained brute force should be preformed only over a reduced key space of dimension 2^{n-8}.

Related Key Attacks

The second condition is that the relationship between the key and each bit of the output should be **complex**. What this means is that a given known relationship between two keys should not produce a known relationship in the output of the keystream generator. This kind of information can also be used to provide an attack by reducing the effective brute keysearch time. These kind of attacks are known as related key attacks.

The following is an illustrative example. Assume we have a keystream generator that has the property that the bitwise compliment of the key produces the bitwise compliment of the keystream. This information can be used to (slightly) speed up a brute force search. For each keystream generated by a guess of the key, complement the keystream and see if that one generates the correct plaintext. If it does, then the complement of the key is correct. In this way, you never have to check the compliment of any key you have already checked. This reduces the search space by a factor of two.

3.1.4 Initialization Vectors Based Attacks

.

One problem with binary additive keystream ciphers is that the same key will always produce the same keystream. This means that repeatedly using the same key is just as bad as reusing a one-time-pad. To solve this problem, the concept of initialization vectors is very important. An initialization vector (IV) is a random value that changes with every instance of the cipher that is used to add some randomness to the output of the cipher. Since this value is random and unique, it makes the output of the stream cipher different than other outputs, even if the same key is used. This is useful when key exchange is expensive. A common method of adding an IV to a cipher is to combine it with the real key in some fashion. For example, prepending a small IV to the real key to form a larger session key is quite common. Another way of using an IV is to encrypt the IV with the real key, or encrypt the real key with the IV, and use this encrypted value as a session key.

It is important to recognize the proper role of the IV. In this kind of usage, the initialization vector is not part of the secret key, and does not need to be kept secret. This means that it can be transmitted in the clear to the recipient.

However, the use of IV's can be security weakness, depending on how the IV is used. A good example is any cipher in which partial knownledge of the key together with some ciphertext can be used to find more of the key. If this is true, then just prepending the IV to the key to form a session key can leak information about the real key.

4 Previously Reported Results on Security Evaluation

This section summarize and discuss the MUGI security self-evaluation claims [2], [3], the MUGI screening observations, [4], and a recently reported results relevant for MUGI security evaluation, [14].

4.1 Self-Evaluation Summary

[2] yields a summary of designers intentions and achievements regarding MUGI pseudo-random number generator construction. The documents [2] and [3] give the evidence on expected security and performance of MUGI. [2]-[3] take into account the widely used

(standard) techniques relevant for security evaluation of MUGI, as well as certain dedicated techniques.

The following security issues are considered: (i) randomness test based on FIPS 140-1; (ii) discussion of period, linear complexity, divide-and-conquer attack; (iii) employment of block ciphers like dedicated evaluation of PRNG; (iv) re-synchronization attacks on MUGI; (v) design and analysis of the key setup. More details on these issues are given bellow.

4.1.1 Preliminaries

MUGI PRNG can be considered as the combination of a LFSR and a round function. The considered LFSR is relatively simpler and much larger than ones used in an usual LFSR-based key stream generators, meanwhile MUGI round function is similar to ones used in a block cipher. Because of this structure there are no "standard" methods at present to evaluate the security of this type PRNG. On the other hand it can be directly checked that MUGI is resistant against the fast correlation attacks including the most recent ones reported in [32] and [13].

Self evaluation report [2] follows the main lines of the evaluation procedure given in Section 3. Particularly, it is taken into account that MUGI PRNG must satisfy following two requirements: (i) The output sequence has good enough randomness; (ii) Any two output sequences generated by different initial data should be significantly different.

4.1.2 Standard General Evaluation Methods

Randomness tests

This section summarizes the results of statistical randomness tests reported in [2]. The tests that were used to examine the randomness of the output sequences of MUGI originated from these from FIPS 140-1, [26] but appropriately modified to be suitable for testing PRNG output sequences. The employed tests were: Frequency test and Run test (including detection of a long run).

Frequency test
The frequences of 1-,2-,4-, and 8-bit patterns were checked. The tests were employed using 2^{22}, 2^{26} and 2^{30}-bit-length sequences to take into account the effect of the sequences length, as well, and 512 randomly selected initial states were considered. The obtained test results do not contradict to the statement that the PRNG outputs are random-like.

Run test
The employed method is that described in [29]. Here is an illustration of the obtained results. For instance, in the test on 2^{22}-bit-length sequences, 3 of 512 output sequences generated were distinguished from truly random sequences with probability 0.99. The expected value that a run of length 31 exists in a 2^{22}-bit stream is 2^{-11}. Detection of a run of length 31 is expected once by 4 times because 512 the sets of initial data were

employed. A detail analysis of the test results imply that run-test results are appropriate.

Period

For each secret key K and each initial vector I, the MUGI output sequence should have a period longer than 2^{128} because the MUGI key length is 128 bits. Estimating the period of the MUGI output sequences is difficult because its update function is non-linear. However, it is claimed in [2] that the huge internal state and the design of ρ imply that the period of its output sequence is longer than that generated by the OFB-mode of a 128-bit block cipher.

Linear Complexity

Linear complexity of MUGI output sequence seems to be very difficult to estimate, but it is claimed in [2] that it is unlikely that it is a critical issue.

Divide-and-Conquer Attack

It s well known that a divide-and-conquer attack could be used when a PRNG has several internal states and one of them has an independent update function. Using this assumption, an attacker hypothesizes this part of the internal state at the first. However, MUGI PRNG has a huge internal state and it is impossible to isolate its update function. Thus, it is claimed in [2] that applying divide-and-conquer attack on MUGI is difficult.

4.1.3 Employment of Block Ciphers Evaluation Methods

Note that MUGI PRNGs structure is similar to that of block ciphers, and accordingly it is natural to consider the resistance of MUGI against block ciphers attacks dedicated to MUGI structure. These approaches were considered in [2]-[3] and they will be discussed in this section.

Differential / Linear Characteristics of F-function

The MUGI F-function has a SPN-structure and consists of byte-wise non-linear transformation using S-box, byte-wise linear transformation by (4×4) matrix MDS, and bytes shuffling.

The S-box and the matrix MDS are the same as those used in AES, [15]. Hence, for example, the maximum differential and linear probability of S-box are 2^{-6} and the branch number of the matrix MDS is five. (Note that the linear probability is normalized.)

Linear Cryptanalysis

Linear cryptanalysis is a standard attack used against block ciphers (see [28] or [29], for example). It employs a linear combination of plural bits that consists of the input block and the output block. The attack is effective when there is a linear combination with

a value that is different from the uniform one. This technique uses the linear correlation between the input and the output blocks, and it requires construction of a linear approximation of output units. Applying this technique to MUGI PRNG is more difficult than applying to a block cipher because the buffer which corresponds to the round keys of block ciphers is dynamically updated. A problem of constructing an effective linear approximations and consideration of calculating the lower bound on the number of active S-boxes required for any possible linear approximation is discussed in [2]. The analysis implies that the output sequence can not be distinguished from a truly random one.

This issue is also analyzed in [3] according to the following. Note that searching for relevant linear combinations corresponds to counting active S-boxes in linear approximations for evaluating the linearity of a MUGI output sequence. Denote the number of active S-boxes of a linear approximation with AS. Taking into account that the linear probability of the MUGI S-box is 2^{-6}, it can be assumed that the linear characteristic of the output sequence of MUGI is sufficiently small if there is no linear approximation with $AS < 22$. It has been proved in [2] and [3] that the following statement holds: "For all linear approximations of MUGI, $AS \geq 22$." The proof of this statement is based on the following two steps:
- Constructing a linear approximation of ρ;
- Searching a path including the buffer.
The employed linear approximation of ρ is based on an equivalent transformation of ρ. Actually, this transformation is not equivalent in the common sense, but it is equivalent in the sense that mask patterns are not changed by the transformation. It is shown that any linear approximations consisting of one or plural rounds are given as the combination of two particular approximations. Also, it is shown that some important paths exist. Only the five paths assure that the number of active S-boxes is greater than five. Note that the branch number of matrix MDS does not assure the number of active S-boxes for a linear approximation, even if it includes several active F-functions. This property is quite different from those of block ciphers.

In the second step a search is needed for a path including the buffer that gives a linear approximation consisting of only output units. For PRNGs, the attacker can observe any number of rounds. So, it is possible to construct the linear approximations with the outputs of any rounds. Furthermore, some linear approximations may skip intermediate ρ functions and there is a possibility that observing more rounds increase the deviation. Accordingly, this feature makes it difficult to search all the paths.

The special attention is related to two rounds, the first and last round of the path. These rounds and their neighbors must meet some required conditions. Finally, it is possible to classify the paths by using these conditions and accordingly calculate the lower bound on active S-boxes for each case.

Other Attacks

Some other attacks against block ciphers such as differential cryptanalysis, higher order differential attack, and interpolation attack, are discussed in [2], as well. All of these attacks are chosen plaintext attacks. Applying these attacks to MUGI PRNG seems to be very difficult. Generally, it is impossible to get the required chosen plaintexts from the

output sequence. For example, if the attacker can construct a distinguisher that consists of 16 output rounds, the number of the possible outputs is $2^{64 \cdot 16}$. This means that the computational complexity required to get one chosen plaintext is more than $2^{64 \cdot 8}$. In addition, using differential cryptanalysis as an example, when the attacker searches the differential characteristics, the attacker assumes the possibility of being able to observe all of the internal states at some round. This assumption is not valid if the initialization is appropriate.

4.1.4 Re-Synchronization Attacks

This section discusses the possibility of using re-synchronization attack [16] against MUGI. Recall that re-synchronization attack can be used against keystream generators, which have not only a secret key, but also a public parameter. It is an effective attack if the initialization of the algorithm is too simple. Under the assumption that the secret key is fixed, the attacker first searches for some relationship between the public parameters and corresponding outputs. If some relationship has a high probability, he can guess information about the secret key from it.

Differential and Linear Characteristics

In [2]-[3], differential and linear characteristics have been chosen for evaluating the relationship between inputs and outputs. The attacks against block ciphers using the first characteristics are well known as differential cryptanalysis, [9], and linear cryptanalysis, [28].

The design of MUGI PRNG, especially its ρ function, is quite similar to a block ciphers design. This suggests that these two statistical properties are well suited for evaluating the relationship between the initial vector I and a corresponding internal state.

An analysis has been performed ignoring the XOR to the buffer and output generation, i.e., by considering only the iteration of ρ and evaluating its differential and linear characteristics. These evaluation methods can be applied in the same way as they are applied to block ciphers.

The relationship between the initial vector I and corresponding state $a^{(t)}$ transformed by t iterations of ρ implies that more than 23 iterations of ρ have differential and linear characteristics with probability higher than 2^{-128}. However, the buffer b influences the differential and linear characteristics of state a only after round -9, i.e., 22 rounds after setting I. Therefore, it can be concluded that to observe the deviation due to these characteristics is difficult after round $t > 0$.

On the other hand, the differential characteristic depends on an output sequence and the buffer which has more than two buffer-units. The correlation between one of them and I is too small to observe. Therefore, the attacker can not exploit that correlation.

The conditions for linear cryptanalysis are similar.

To observe the correlation between keys and corresponding outputs is more difficult than the correlation between initial vectors and the corresponding outputs because of the first mixing step.

Accordingly, it is claimed in [2]-[3] that no security flaw can be found by using differential and linear cryptanalysis.

Square-Attack Variants

Because of the highly byte-oriented structure, some of the Square attack, [17], variants should be considered. The Square attack is currently one of the most successful attack against the block ciphers with the SPN-structure, e.g., Rijndael, the AES. [2]-[3] has examined the applicability of the attack and investigate the possible properties. Consequently, it has been concluded that any possible variant of the Square attack do not have reduction impact on the security of MUGI PSNG.

The Square attack against a block cipher is basically a chosen-plaintext attack where an attacker chooses a number of related plaintext blocks each of which is typically differentiate only in a byte or a word. Because of the saturation at the input of a non-linear function, the attacker can expect to control the intermediate values in some extent. From the ciphertext side, the attacker partially decrypts the intermediate value which is still controlled because of the saturated plaintext blocks. If the attacker guesses the key for the partial decryption, then the attacker can distinguish the correct and incorrect round keys.

In a stream cipher, an attacker must try to select different values of either key or IV values to mount this attack. Therefore the possible applications of the Square attack must be either a related-key cryptanalysis or a chosen initial-vector attack.

Related-Key Attacks

The following consideration has been reported in [2]-[3]. At first, we define the model of the attack. We assume that the attacker does not know the key value. To mount the saturation property, the attacker can run a number of key initializations, where the keys differ only in a part of the key value, especially in this discussion we will concentrate on a key-dependent runs where the keys differ in one word. The attacker cannot observe anything until the pseudorandom number sequence comes out. We check if the attacker may find any properties at the output sequences amongst a number of runs.

The saturated key set will inject the saturation property during the buffer initialization. At firrst, we investigate how buffers are initialized with the properties. For simplicity, we ignore the key padding rule so that we give the attacker the most flexibility for setting the initial state values. An important property is one of an intermediate word such that in each run the concerning word has different value, i.e., the word is saturated. Another important property is the property that for all runs the value are constant. The third one property is "balanced" that means the XOR-summation over all runs is zero. An analysis related to the above properties and given in [2]-[3], implies that thanks to the subsequent randomization after IV injection, the undesired property must be destroyed until the output sequence is generated. Therefore it is claimed that the related-key attack based on the Square attack is not feasible.

Chosen IV Attacks

This attack may be more practical than the above related-key cryptanalysis. However, the IV does not inject any value to the buffer until the 16-round mixing completes. Taking the number of controllable rounds into account, 16-round mixing is sufficient to destroy the saturation property due to IV.

Non-Linear Buffer Relation

The initial buffer content is generated only by the secret key. The key setup algorithm generate each initial unit per a round. Since the round function of the key setup is far from the random permutation, there are relations between initial buffer values. These relations between buffer values are discussed in [2].

Accordingly, if all the units (64-bit registers) in the buffer and the state has the property Ω_{64} (see [2]) (it is not necessary that bytes beyond the unit are the same), the non-linear function F does not change this property. On the other hand, thanks to the subsequent constant XORing after two F functions, this property is immediately destroyed. Consequently because of the constant XORing we think the attack based on the property Ω does not imply an effective attack to MUGI PRNG.

4.1.5 Analysis of the Key Setup

The key setup algorithm is designed so that the resultant initialization of the buffer and the state is enough randomized to generate a secure pseudorandom sequence after the initialization. Security issues of the key setup are discussed in this section based on the results reported in [2].

(Im)possibility of the linearly-keyed buffer

Generally speaking, the buffer value is initialized non-linearly with respect to the raw key value because of the update function ρ. However in some special cases, the non-linearity may reduce. One of those cases are ones when many inputs to the ρ functions are the same. This may cause the extremely simple linear relations between the input (the raw key) and the output (the initialized buffer values). Here, a possibility of appearance of a key that initializers the buffer with extremely simple linear relations is considered.

Recall that there are two stages in the buffer initialization:
(A) the key-dependent initialization, and;
(B) the randomization after setting the initial vector, IV hereafter.
Note that if the algorithm takes the key that the buffer initialization by (A) has sufficiently high non-linearity, then it is very unlikely to happen that the resultant buffer (just before outputting the pseudorandom sequence) contains linear relations (with no more than 16 word unknown variables). From this remark, the analysis has two objectives: verification of the fact that the used functions are sufficiently non-linear for most of the key space elements, and that the size of the keys that generate the linear buffer relationship is small enough and negligible. Note that for those limited number of weak keys (in the sense

of the linear buffering) there are another non-linear randomization on to all buffer value after injection of the IV which implies further mixing of the buffer and state contents. Consequently we conclude that the key setup sufficiently randomizes the buffer with respect to the linear-buffer initialization.

The following statements are given in [2]:
• Case 1 (One-round iteration): ρ function does not have any fixed points, i.e., no key generates the same output for all buffer units.
• Case 2 (Two-round iteration): The structure with two iterations of ρ function does not have many fixed points, i.e., no obvious weak key classes with respect to the two-value initialized buffer do not exist.
• Case 3 (Three-round iteration): The structure with three iterations of ρ function does not have any fixed points.

The final analysis that includes consideration of more than three rounds cas, as well, implies that the size of class of the weak keys is too small to mount a meaningful attack.

4.2 Security Comments from MUGI Screening Evaluation Phase

According to [4], the screening evaluation phase of MUGI has pointed out a number of issues including the following ones which seem to be the most important for this report.
(i) At the time of screening evaluation, there was no reported security flaw but an independent (not by MUGI designers) security evaluation is strongly recommanded.
(ii) S-box in F-function and matrix MDS are the same as these used by AES, however theirs design methods should be studed.
(iii) MUGI should be compared with PANAMA and MULTI-S01.

4.3 An Attack on Ciphers Based on Linear Masking

Recently, [14], has described a cryptanalytical technique for distinguishing some stream ciphers from a truly random process. The ciphers to which this method applies consist of two processes: one is a "non-linear process" (say, a typical round function in a block ciphers), and the other is a "linear process" such as an LFSR (or even fixed tables). These processes can feed each other, and the output of the cipher can be the linear sum of both processes. This combination seems to be attractive for designing fast stream ciphers. Examples of such ciphers include SEAL and PANAMA, and the newer SNOW, MUGI and Scream. The idea behind the distinguishing technique is very simple. We first concentrate on the non-linear process, looking for a characteristic that can be distinguished from random: For example, a linear approximation that has noticeable bias. We then look at the linear process, and find some linear combination of it that vanishes. If we now take the same linear combination of the output stream, then the linear process would vanish, and we are left with a sum of linear approximations, which is itself a linear approximation. This technique is not limited to linear approximations: In some sense, it can be used with "any distinguishing characteristic" of the non-linear process.

In [14] it is analyzed in details two types of "distinguishing characteristics", and show some examples of its use for specific ciphers. The effectiveness of this approach, when applied to a specific "distinguishing properties" is also considered. One property is a linear

approximation of the non-linear process, and the other is a "low-diffusion" attack. The characteristics of these attacks are demonstrated by analyzing a few ciphers. Specifically, it is shown a linear attack that can distinguish SNOW from a random process after seeing roughly 2^{95} words of output, with work-load of about 2^{100}. Also, a low-diffusion attack on Scream-0 is proposed, that distinguishes it from a random process after seeing only 2^{43} bytes of output, using 2^{50} space and 2^{80} time.

Finally, the authors of [14] claim the following:

- We believe that similar attacks can be devised against PANAMA and MUGI, but we did not try to look at them yet.

Accordingly, main applications of the technique reported in [14] can be the following ones.

Linear attacks.
Perhaps the most obvious use of the above technique, is to devise linear attacks. This is also the easiest case to analyze. An exact formula has been obtained for the amount of text that must be observed in order to distinguish the cipher from random using a linear approximation, as a function of the quality of the original approximation of the non-linear process, and the weight distribution of some linear code that is related to the linear process of the cipher.

Low-diffusion attacks.
Another type of attacks uses the low diffusion in the non-linear process. Namely, some input and output bits of this process depend only on very few other input and output bits. For this type of attacks, it has been analyzed the amount of text that an attacker needs to see, as a function of the number of bits in the low-diffusion characteristic. This analysis is substantially harder than for the linear attacks. Indeed, here we do not have a complete characterization of the possible attacks of this sort, but only an analysis for the most basic such attacks.

Part II: Novel Approach for Cryptanalysis of MUGI

Following the final notes from Section 3 (see the statements 1 and 2 just before Section 3.1) main aim of this report part is to point out a possibility for developing certain novel specialized cryptanalytic attacks against MUGI. The approach to be proposed in the next two sections is an algebraic cryptanalytic method based on solving an overdefined system of nonlinear equations.

As an illustration of the algebraic cryptanalysis based on overdefined systems of quadratic equations (of an encryption scheme different from MUGI) the recent results reported in [32] could be considered.

5 Cryptanalysis of a Simplified MUGI

Main objective of a simplified MUGI consideration is to point out and discuss an undesirable property of the main MUGI nonlinear component. This undesirable property is a weakness which can be employed for mounting cryptanalytic attacks against simplified as well as full MUGI.

5.1 Specification of a Simplified MUGI

A simplified MUGI which is under consideration has the following characteristics:

- It has the same initialization algorithm as full MUGI and it employs the same secret key and the public initialization value.
- It employs a simplified PRNG algorithm: Main differences between a simplified MUGI specified for the cryptanlaysis purposes and the original one are the following:
  - all the rotation operations are excluded from PRNG algorithm;
  - MDS matrices are excluded from PRNG algorithm;
  - XOR-ing with the known constants is excluded.

Accordingly, PRNG of the simplified MUGI can be splitted into eight identical autonomous blocks.

Employing a similar notation as in description of the original MUGI, each of these blocks can be specified as follows:

$$a_0^{(t+1)} = a_1^{(t)}$$

$$a_1^{(t+1)} = a_2^{(t)} \oplus S(a_1^{(t)}, b_4^{(t)}) \ ,$$
$$a_2^{(t+1)} = a^{(t)} \oplus S(a_1^{(t)}, b_{10}^{(t)}) \ .$$

$$b_j^{(t+1)} = b_{j-1}^{(t)} \ \ (j \neq 0, 4, 10) \ ,$$

$$b_0^{(t+1)} = b_{15}^{(t)} \oplus a_0^{(t)} \ ,$$
$$b_4^{(t+1)} = b_3^{(t)} \oplus b_7^{(t)} \ ,$$
$$b_{10}^{(t+1)} = b_9^{(t)} \oplus (b_{13}^{(t)}) \ .$$

$$BlockOut[t] = a_2^{(t)} \ , \tag{1}$$

where each variable is an 8-bit byte. Also, recall that S denotes the S-box which is identical to the one employed in AES.

Finally, note that the nonlinearity in the simplified MUGI is only due to the S-boxes. Accordingly, in the next sections certain characteristics of the S boxes, relevant for developing a cryptanalytic attack, will be in detail analyzed.

5.2 Characteristics of MUGI S-boxes

This section points out very important characteristics of MUGI S-boxes which are same as AES (Rijndael) S-boxes, using the results very recently reported in [21] related to Rijndael security evaluation.

5.2.1 S-boxes and Overdefined Algebraic Equations

The main non-linear part of MUGI keystream generator are the S-boxes. MUGI S-boxes are the same S-boxes developed and used in Rijndael (AES).

Let $S : GF(2)^s \rightarrow GF(2)^s$ be such an S-box $S : x = (x_1...x_s) \rightarrow y = (y_1...y_s)$. In Rijndael, like for all other "good" block ciphers, the S-boxes are build with "good" boolean functions. There are many criteria on boolean functions that are more or less applied in cryptography. One of them is that each y_i should have a high algebraic degree when expressed as a multivariate polynomial in the x_i. However all this does not assure that there is no "implicit" multivariate equations of the form $P(x_1, ..., x_s, y_1, ..., y_s)$ that are of low algebraic degree.

Such "implicit" equations has already been used to cryptanalyse public-key crypto systems, the Matsumoto-Imai cryptosystem in [36] and the HFE cryptosystem in [20], as well as very recently for block ciphers Rijndael and Serpent in [21].

For a specific degree of the equations d (usually $d = 2$) we are interested in the actual number r of such equations $P(x_1, ..., x_s, y_1, ..., y_s)$. Unlike for "explicit" equations $y_i = f(x_1...x_s)$, this number r can be bigger than s. We are also interested in the number of monomials that appear in these equations denoted by t, and counted including the constant term. In general $t \approx \binom{s}{d}$. If $t << \binom{s}{d}$ the equations are sparse. If $r = s$, such equations are (approximatively) sufficient to fully describe the S-box: for each y there will be on average 1 solution x. Thus when $r >> s$, we will say that the system is overdefned.

5.2.2 MUGI S-boxes and Random S-boxes

When r is close to t, we may eliminate most of the terms by linear elimination, and obtain simpler equations that are sparse and maybe even linear. For this reason it is possible to measure the quality of our system of equations by the ratio $t/r \geq 1$. If t/r is close to 1, the S-box is considered as "bad". From this point of view, both overdefned systems (big r) and sparse systems (small t) will be "bad". Otherwise, if the system is not overdefned and not sparse, $t/r \approx O(s^{d-1})$, and such an S-box will be "good" (unless s is very small).

We shall see that the actual contribution of the S-boxes to the complexity of certain attacks is approximatively $\Gamma = (t/s)^{\lceil t/r \rceil}$, and on the other hand it will be shown that for MUGI (Rijndeal) S-boxes, it is possible to specify the overdefned systems of equations with quite a small Γ.

5.2.3 Overdefined Equations of MUGI S-boxes

MUGI employs the Rijndael S-box which has been chosen for optimality results with regard to linear, differential and high-order differential attacks, and is currently the unique S-box known that achieves all these optima (see [12] and [35] for details).

This uniqueness implies many very special properties. Rijndael S-box is a composition of the "patched" inverse in GF(256) with 0 mapped on itself, with a multivariate affine transformation $GF(2)^8 \rightarrow GF(2)^8$. Following [15] we call these functions respectively g and f and we call $S = f \circ g$. Let x be an input value and $y = g(x)$ the corresponding output value. We also note $z = S(x) = f(g(x)) = f(y)$. According to the definition of the S-box:

$$\forall x \neq 0 \quad 1 = xy \ .$$

This equation gives in turn 8 multivariate bi-linear equations in 8 variables and this leads to 8 bi-affine equations between the x_i and the z_j. As it will be explained more in details in the next section, 7 of these equations are true with probability 1, and the 8th is true with probability 255/256. The existence of these equations for g and S is obvious. Surprisingly, much more such equations exist. For example we have:

$$x = yx^2 \ .$$

Since $x \mapsto x^2$ is linear, if written as a set of 8 multivariate functions, the above equation gives 8 bi-affine equations between the x_i and the y_j , and in turn between the x_i and the z_j. Moreover this equation in GF(256) is symmetric with respect to the exchange of x and y. Thus we get 16 bi-affine equations true with probability 1 between the x_i and the z_j. From the above we have 23 quadratic equations between x_i and the z_j that are true with probability 1. These equations are explicitly computed in [21] and will be specified in the next section. In [21] it has reported the verification that they are all linearly independent, and have also that there are no more such equations (however there would be more if we allowed additional terms).

There are $t = 81$ terms present in these equations: these are:

$$\{1, x_1, ..., x_8, z_1, ..., z_8, x_1z_1, ..., x_8z_8\} \ ,$$

and there is no terms of the forms $x_i x_j$ or $z_i z_j$. Accordingly, we get $t/r \approx 3.52$ and $\Gamma \approx 2^{13.4}$.

Also note the following: In MUGI (Rijndael) S-box, if x is always different than 0, there 24 linearly independent quadratic equations. For one S-box, the probability of this 24th equation to be true is 255/256.

5.2.4 Specification of the Overdefined Equations

Recall that we use the following notation: We note x an input value and $y = g(x)$ the corresponding output value. We will also note $z = S(x) = f(g(x)) = f(y)$.

The definition of S implies:

$$\forall x \neq 0 \quad 1 = xy \ .$$

This equation gives in turn 8 bi-linear equations in 8 variables. In [21] the following resulting equations between the inputs and outputs of the whole S-box are determined:

$$\begin{aligned} 0 = {} & z_0x_4 + z_0x_5 + z_0x_1 + x_0z_6 + x_0z_4 + x_0z_1 + x_2z_7 + x_2z_4 + x_2z_2 + x_3z_6 + x_3z_3 + x_3z_1 + \\ & x_4z_6 + x_4z_5 + x_4z_4 + x_4z_2 + x_4z_1 + x_5z_6 + x_5z_7 + x_5z_5 + x_5z_3 + x_6z_6 + x_6z_7 + x_6z_5 + \\ & x_6z_4 + x_6z_2 + x_7z_5 + x_7z_3 + x_1z_5 + x_1z_3 + x_5 + x_7 \end{aligned} \tag{2}$$

$$\begin{aligned} 0 = {} & z_0x_0 + z_0x_3 + z_0x_4 + x_0z_5 + x_0z_3 + x_2z_6 + x_2z_3 + x_2z_1 + x_3z_6 + x_3z_5 + x_3z_4 + x_3z_2 + \\ & x_3z_1 + x_4z_6 + x_4z_7 + x_4z_5 + x_4z_3 + x_5z_6 + x_5z_7 + x_5z_5 + x_5z_4 + x_5z_2 + x_6z_5 + x_6z_3 + \\ & x_7z_6 + x_7z_2 + x_7z_1 + x_1z_7 + x_1z_4 + x_1z_2 + x_4 + x_6 \end{aligned} \tag{3}$$

$$\begin{aligned} 0 = {} & z_0x_2 + z_0x_3 + z_0x_7 + x_0z_7 + x_0z_4 + x_0z_2 + x_2z_6 + x_2z_5 + x_2z_4 + x_2z_2 + x_2z_1 + x_3z_6 + \\ & x_3z_7 + x_3z_5 + x_3z_3 + x_4z_6 + x_4z_7 + x_4z_5 + x_4z_4 + x_4z_2 + x_5z_5 + x_5z_3 + x_6z_6 + x_6z_2 + \\ & x_6z_1 + x_7z_5 + x_7z_1 + x_1z_6 + x_1z_3 + x_1z_1 + x_3 + x_5 + x_7 \end{aligned} \tag{4}$$

$$\begin{aligned} 0 = {} & z_0x_2 + z_0x_6 + z_0x_7 + z_0x_1 + x_0z_6 + x_0z_3 + x_0z_1 + x_2z_6 + x_2z_7 + x_2z_5 + x_2z_3 + x_3z_6 + \\ & x_3z_7 + x_3z_5 + x_3z_4 + x_3z_2 + x_4z_5 + x_4z_3 + x_5z_6 + x_5z_2 + x_5z_1 + x_6z_5 + x_6z_1 + x_7z_6 + \\ & x_7z_7 + x_7z_1 + x_1z_6 + x_1z_5 + x_1z_4 + x_1z_2 + x_1z_1 + x_2 + x_4 + x_6 + x_7 \end{aligned} \tag{5}$$

$$\begin{aligned} 0 = {} & z_0x_0 + z_0x_4 + z_0x_6 + z_0x_7 + x_0z_5 + x_0z_2 + x_2z_6 + x_2z_5 + x_3z_6 + x_3z_5 + x_3z_1 + x_4z_5 + \\ & x_4z_4 + x_5z_6 + x_5z_7 + x_5z_3 + x_5z_1 + x_6z_5 + x_6z_4 + x_6z_2 + x_6z_1 + x_7z_6 + x_7z_7 + x_7z_3 + \\ & x_1z_6 + x_1z_7 + x_3 + x_6 + x_1 \end{aligned} \tag{6}$$

$$0 = z_0x_3 + z_0x_4 + z_0x_6 + z_0x_1 + x_0z_7 + x_0z_4 + x_0z_1 + x_2z_6 + x_2z_7 + x_2z_5 + x_2z_4 + x_2z_2 +$$
$$x_2z_1 + x_3z_6 + x_3z_5 + x_3z_4 + x_3z_3 + x_3z_1 + x_4z_7 + x_4z_5 + x_4z_4 + x_4z_3 + x_4z_2 + x_5z_6 +$$
$$x_5z_7 + x_5z_4 + x_5z_3 + x_5z_2 + x_5z_1 + x_6z_5 + x_6z_4 + x_6z_3 + x_6z_2 + x_7z_7 + x_7z_4 + x_7z_3 +$$
$$x_7z_2 + x_7z_1 + x_1z_6 + x_1z_3 + x_0 + x_2 + x_7 \quad (7)$$

$$0 = z_0x_0 + z_0x_2 + z_0x_3 + z_0x_5 + z_0x_7 + x_0z_6 + x_0z_3 + x_2z_6 + x_2z_5 + x_2z_4 + x_2z_3 + x_2z_1 +$$
$$x_3z_7 + x_3z_5 + x_3z_4 + x_3z_3 + x_3z_2 + x_4z_6 + x_4z_7 + x_4z_4 + x_4z_3 + x_4z_2 + x_4z_1 + x_5z_5 +$$
$$x_5z_4 + x_5z_3 + x_5z_2 + x_6z_7 + x_6z_4 + x_6z_3 + x_6z_2 + x_6z_1 + x_7z_4 + x_7z_3 + x_7z_2 + x_1z_6 +$$
$$x_1z_7 + x_1z_5 + x_1z_4 + x_1z_2 + x_1z_1 + x_6 + x_7 + x_1 \quad (8)$$

$$1 = x_0 + x_6 + z_0x_2 + z_0x_5 + z_0x_6 + x_0z_7 + x_0z_5 + x_0z_2 + x_2z_5 + x_2z_3 + x_3z_7 + x_3z_4 + x_3z_2 +$$
$$x_4z_6 + x_4z_3 + x_4z_1 + x_5z_6 + x_5z_5 + x_5z_4 + x_5z_2 + x_5z_1 + x_6z_6 + x_6z_7 + x_6z_5 + x_6z_3 +$$
$$x_7z_6 + x_7z_7 + x_7z_5 + x_7z_4 + x_7z_2 + x_1z_6 + x_1z_4 + x_1z_1 \quad (9)$$

The frst 7 equations have no constant parts and therefore are also true for $x = 0$. Therefore we obtained here 7 equations that are true with probability 1, plus one additional equation that is true if and only if $x \neq 0$, i.e. with probability 255/256. The existence of these (quadratic) equations is obvious. Surprisingly, it is also shown in [21] that much more such equations exist. (It leads to systems of equations that have much more equations than unknowns, and allows more powerful attacks.)

Note the following:

$$\forall x \neq 0 \quad x = x^2y$$

This equation happens to be true also for x = 0. Accordingly:

$$\forall x \in GF(256) \begin{cases} x = x^2y \\ x^2 = x^4y^2 \\ \cdot \\ \cdot \\ \cdot \\ x^{128} = xy^{128} \end{cases}$$

Each of equation is the square of the previous one, and since the square is linear as a multivariate function, each these 8 equations will generate the same set (more precisely the same modulo a linear combination) of 8 multivariate equations on the x_i and the y_j

. We choose therefore one of these equations, for example the last. It is symmetric with respect to the exchange of x and y and we obtain the following 2 equations:

$$\begin{cases} x^{128} = xy^{128} \\ y^{128} = yx^{128} \end{cases}$$

Since $x \mapsto x^{128}$ is linear, if written as a set of 8 multivariate linear functions, each of above 2 equations will give 8 quadratic equations with 8 variables. The following resulting equations on the whole S-box are given in [21].

$$\begin{aligned} 0 = {} & x_3 + x_5 + x_6 + x_1 + x_2z_2 + x_5z_7 + x_7z_4 + x_7z_1 + x_7z_3 + x_0z_1 + x_6z_5 + x_6z_3 + x_7z_7 + x_4z_6 + \\ & x_4z_1 + x_4z_5 + x_4z_0 + x_4z_2 + x_1z_5 + x_1z_3 + x_5z_5 + x_5z_3 + x_5z_0 + x_3z_1 + x_3z_3 + x_6z_6 + x_3z_4 + \\ & x_2z_3 + x_2z_6 + x_4z_7 + x_0z_5 + x_0z_3 + x_1z_4 + x_1z_7 + x_6z_1 + x_3z_0 + x_4z_3 + x_0z_7 + x_1z_6 + x_2z_5 \end{aligned} \quad (10)$$

$$\begin{aligned} 0 = {} & x_3 + x_6 + x_1 + x_2z_4 + x_5z_1 + x_7z_1 + x_5z_6 + x_0z_6 + x_0z_4 + x_6z_3 + x_6z_4 + x_6z_7 + x_7z_7 + \\ & x_7z_5 + x_7z_2 + x_4z_5 + x_4z_0 + x_1z_5 + x_1z_3 + x_5z_5 + x_5z_3 + x_3z_1 + x_3z_3 + x_3z_6 + x_2z_1 + x_2z_3 + \\ & x_4z_7 + x_0z_5 + x_0z_3 + x_1z_2 + x_6z_1 + x_3z_5 + x_3z_0 + x_3z_2 + x_4z_3 + x_0z_7 + x_3z_7 + x_1z_6 + x_2z_0 \end{aligned} \quad (11)$$

$$\begin{aligned} 0 = {} & x_3 + x_4 + x_5 + x_1 + x_2z_2 + x_2z_7 + x_5z_1 + x_5z_4 + x_5z_7 + x_7z_6 + x_7z_4 + x_7z_1 + x_0z_6 + x_6z_5 + \\ & x_6z_2 + x_6z_7 + x_7z_7 + x_4z_6 + x_4z_1 + x_4z_5 + x_1z_3 + x_1z_0 + x_5z_3 + x_3z_3 + x_2z_1 + x_2z_3 + \\ & x_2z_6 + x_0z_5 + x_0z_3 + x_1z_4 + x_6z_1 + x_3z_5 + x_3z_0 + x_4z_3 + x_0z_2 + x_3z_7 + x_1z_1 + x_2z_5 + x_2z_0 \end{aligned} \quad (12)$$

$$\begin{aligned} 0 = {} & x_3 + x_4 + x_1 + x_2z_7 + x_5z_1 + x_5z_7 + x_7z_4 + x_0z_4 + x_0z_1 + x_6z_4 + x_6z_7 + x_7z_7 + x_7z_5 + \\ & x_7z_2 + x_4z_4 + x_4z_1 + x_1z_5 + x_1z_3 + x_1z_0 + x_5z_5 + x_3z_1 + x_3z_3 + x_3z_6 + x_6z_6 + x_5z_2 + \\ & x_2z_3 + x_4z_7 + x_0z_3 + x_0z_0 + x_1z_2 + x_1z_7 + x_6z_1 + x_3z_5 + x_4z_3 + x_1z_1 + x_1z_6 + x_2z_5 + x_2z_0 \end{aligned} \quad (13)$$

$$\begin{aligned} 0 = {} & x_2 + x_6 + x_7 + x_1 + x_2z_2 + x_5z_1 + x_5z_4 + x_7z_4 + x_7z_1 + x_5z_6 + x_7z_3 + x_0z_6 + x_6z_3 + \\ & x_6z_2 + x_6z_4 + x_6z_7 + x_7z_7 + x_7z_2 + x_4z_6 + x_4z_0 + x_1z_0 + x_5z_5 + x_5z_3 + x_5z_0 + x_6z_6 + \\ & x_2z_1 + x_0z_0 + x_1z_4 + x_6z_1 + x_3z_0 + x_4z_3 + x_0z_2 + x_3z_7 + x_1z_6 \end{aligned} \quad (14)$$

$$\begin{aligned} 0 = {} & x_2 + x_3 + x_4 + x_5 + x_1 + x_2z_2 + x_2z_7 + x_5z_1 + x_5z_4 + x_7z_6 + x_7z_1 + x_5z_6 + x_0z_6 + \\ & x_0z_4 + x_0z_1 + x_6z_5 + x_6z_2 + x_6z_4 + x_6z_7 + x_7z_2 + x_4z_4 + x_4z_2 + x_1z_5 + x_1z_3 + x_5z_5 + \end{aligned}$$

$$x_5z_0 + x_3z_1 + x_3z_6 + x_6z_6 + x_5z_2 + x_3z_4 + x_2z_3 + x_2z_6 + x_4z_7 + x_0z_5 + x_0z_3 + x_0z_0 +$$
$$x_1z_2 + x_1z_4 + x_1z_7 + x_0z_7 + x_1z_1 + x_1z_6 + x_2z_5 + x_2z_0 \quad (15)$$

$$0 = x_0 + x_2 + x_3 + x_7 + x_2z_4 + x_5z_4 + x_5z_7 + x_7z_6 + x_7z_1 + x_5z_6 + x_0z_6 + x_0z_4 + x_0z_1 +$$
$$x_6z_2 + x_7z_7 + x_4z_6 + x_4z_4 + x_4z_1 + x_4z_5 + x_4z_0 + x_4z_2 + x_1z_5 + x_1z_3 + x_1z_0 + x_5z_5 +$$
$$x_6z_6 + x_5z_2 + x_3z_4 + x_2z_1 + x_2z_6 + x_7z_0 + x_0z_5 + x_0z_3 + x_1z_2 + x_1z_7 + x_6z_1 + x_3z_2 +$$
$$x_0z_2 + x_0z_7 + x_3z_7 + x_1z_6 \quad (16)$$

$$0 = x_3 + x_5 + x_2z_4 + x_2z_7 + x_5z_1 + x_5z_7 + x_7z_6 + x_7z_1 + x_5z_6 + x_7z_3 + x_0z_6 + x_0z_1 + x_6z_5 +$$
$$x_6z_3 + x_6z_0 + x_6z_7 + x_7z_5 + x_4z_4 + x_4z_1 + x_4z_0 + x_1z_5 + x_1z_3 + x_5z_5 + x_5z_3 + x_5z_0 +$$
$$x_3z_3 + x_3z_6 + x_5z_2 + x_2z_3 + x_2z_6 + x_0z_0 + x_1z_7 + x_3z_5 + x_3z_2 + x_4z_3 + x_0z_2 + x_1z_1 + x_2z_5 \quad (17)$$

$$0 = x_5 + x_7 + z_7 + z_5 + z_3 + z_1 + x_5z_1 + x_5z_4 + x_7z_3 + x_0z_6 + x_0z_4 + x_0z_1 + x_6z_3 + x_7z_2 + x_4z_4 +$$
$$x_4z_2 + x_1z_5 + x_1z_0 + x_5z_3 + x_6z_6 + x_3z_4 + x_2z_3 + x_4z_7 + x_7z_0 + x_6z_1 + x_3z_7 + x_2z_5 + x_2z_0 \quad (18)$$

$$0 = x_3 + x_5 + x_7 + z_6 + z_7 + z_5 + z_4 + z_3 + x_2z_2 + x_2z_4 + x_2z_7 + x_7z_1 + x_6z_5 + x_6z_0 +$$
$$x_6z_2 + x_6z_4 + x_7z_7 + x_7z_2 + x_4z_6 + x_4z_1 + x_5z_3 + x_5z_0 + x_3z_1 + x_3z_3 + x_6z_6 + x_5z_2 +$$
$$x_3z_4 + x_0z_5 + x_0z_3 + x_0z_0 + x_1z_4 + x_1z_7 + x_6z_1 + x_4z_3 \quad (19)$$

$$0 = x_3 + x_5 + x_6 + x_7 + x_1 + z_6 + z_5 + z_3 + z_2 + x_5z_1 + x_5z_7 + x_7z_6 + x_7z_1 + x_0z_4 + x_6z_5 +$$
$$x_6z_3 + x_6z_0 + x_6z_7 + x_4z_6 + x_4z_4 + x_4z_1 + x_4z_5 + x_4z_0 + x_4z_2 + x_1z_3 + x_3z_3 + x_6z_6 +$$
$$x_5z_2 + x_2z_1 + x_2z_3 + x_2z_6 + x_7z_0 + x_1z_4 + x_3z_0 + x_3z_2 + x_0z_2 + x_0z_7 + x_1z_1 \quad (20)$$

$$0 = x_3 + x_4 + x_5 + x_1 + z_4 + z_3 + z_1 + z_0 + x_2z_2 + x_2z_4 + x_5z_1 + x_5z_6 + x_0z_6 + x_0z_1 + x_6z_5 +$$
$$x_6z_2 + x_6z_4 + x_6z_7 + x_7z_7 + x_7z_5 + x_4z_6 + x_4z_5 + x_4z_0 + x_1z_3 + x_1z_0 + x_5z_0 + x_3z_1 +$$
$$x_6z_6 + x_2z_1 + x_2z_6 + x_4z_7 + x_7z_0 + x_0z_3 + x_1z_2 + x_3z_2 + x_4z_3 + x_3z_7 + x_2z_5 + x_2z_0 \quad (21)$$

$$0 = x_2 + x_3 + x_5 + x_6 + x_1 + z_6 + z_2 + z_0 + x_2z_7 + x_5z_1 + x_5z_4 + x_5z_7 + x_7z_6 + x_7z_4 + x_7z_3 +$$

$$x_6z_5 + x_7z_7 + x_7z_2 + x_4z_6 + x_4z_5 + x_1z_5 + x_1z_0 + x_5z_5 + x_5z_3 + x_5z_0 + x_3z_1 + x_3z_6 +$$
$$x_6z_6 + x_3z_4 + x_2z_6 + x_7z_0 + x_0z_5 + x_0z_0 + x_1z_2 + x_1z_7 + x_6z_1 + x_3z_0 + x_0z_2 + x_3z_7 + x_1z_1 \quad (22)$$

$$0 = x_0 + x_3 + x_4 + x_5 + x_1 + z_6 + z_7 + z_5 + z_4 + z_3 + z_1 + z_0 + x_5z_1 + x_5z_7 + x_7z_4 + x_5z_6 +$$
$$x_0z_4 + x_0z_1 + x_6z_5 + x_6z_3 + x_6z_0 + x_6z_4 + x_7z_7 + x_7z_5 + x_4z_6 + x_4z_4 + x_4z_1 + x_4z_5 +$$
$$x_4z_2 + x_1z_5 + x_5z_0 + x_3z_1 + x_3z_3 + x_6z_6 + x_5z_2 + x_2z_3 + x_2z_6 + x_4z_7 + x_7z_0 + x_1z_4 +$$
$$x_1z_7 + x_6z_1 + x_3z_5 + x_3z_0 + x_0z_7 + x_1z_1 + x_1z_6 + x_2z_0 \quad (23)$$

$$0 = x_2 + x_3 + x_7 + x_1 + z_6 + z_7 + z_5 + z_4 + z_3 + z_2 + z_1 + 1 + x_2z_2 + x_2z_4 + x_2z_7 + x_5z_4 +$$
$$x_5z_7 + x_7z_1 + x_7z_3 + x_0z_6 + x_6z_5 + x_6z_3 + x_6z_0 + x_6z_2 + x_6z_4 + x_6z_7 + x_7z_7 + x_4z_6 +$$
$$x_4z_4 + x_4z_1 + x_4z_5 + x_4z_0 + x_1z_5 + x_1z_3 + x_1z_0 + x_5z_5 + x_5z_0 + x_3z_1 + x_3z_6 + x_2z_1 +$$
$$x_2z_6 + x_0z_3 + x_0z_0 + x_3z_0 + x_3z_2 + x_4z_3 + x_3z_7 + x_1z_1 + x_2z_5 \quad (24)$$

$$0 = x_0 + x_7 + x_1 + z_6 + z_2 + z_1 + z_0 + 1 + x_2z_4 + x_5z_4 + x_5z_7 + x_7z_4 + x_7z_1 + x_7z_3 + x_6z_2 +$$
$$x_6z_4 + x_6z_7 + x_4z_5 + x_4z_0 + x_1z_5 + x_2z_1 + x_2z_6 + x_0z_5 + x_1z_2 + x_1z_7 + x_3z_5 + x_3z_0 +$$
$$x_4z_3 + x_0z_2 + x_0z_7 + x_1z_1 + x_1z_6 \quad (25)$$

Accordingly for each S-box we have 23 bi-affine equations between the x_i and the z_j. It is claimed in [21] that it has been verified that all these equations are linearly independent and that there are no more such equations. Moreover, if x is always different than 0, the 24 linearly independent equations are available.

Also note the following:

- **Fully quadratic equations.**
  It is possible to see that if we consider fully quadratic equations, not only bi-affine, for each S-box there are $r = 39$ quadratic equations with $t = 137$. The additional 16 equations come from the following two equations:

$$\begin{cases} x^4y = x^3 \\ y^4x = y^3 \end{cases}$$

  However using $r = 39$ and $t = 137$ does not imply any gain over using $r = 23$ and $t = 81$. This is due to the fact that it gives $\Gamma = 2^{16.4}$ instead $2^{13.4}$.

- **A General remark on inverse-based S-boxes.**
  It is easy to see that if the S-box on s is an affine transformation of the inverse function in GF(2^s), then it will give $3s-1$ bi-affine equations true with probability 1, and one additional equation true with probability $1-\frac{1}{2^s}$. It is conjectured in [21] that there is no more such equations. Up till now, it seemed a very good idea to use such S-boxes in practical ciphers. This was due to the fact that the inverse function (and its affine equivalents) has many optimality results with regard to linear, differential and high-order differential attacks, (see [12] and [35]). In [21] it is reported that the computer simulations for many permutations including all the possible powers of GF(2^s) which show that the inverse (and it's equivalents) is the worse in terms of the number of such bi-affine equations. It is an open problem to find any other non-linear function $GF(2^s) \rightarrow GF(2^s)$ that admits so many equations, for some s ¿0. Therefore though in many cases the ciphers are probably still very secure, it seems not advisable to use such S-boxes.

5.3 Algorithm for Cryptanalysis of the Simplified MUGI

The previous consideration show that the characteristics of MUGI S boxes open a door for developing the following algebraic method for cryptanalysis of the simplified MUGI.

Main steps of the algorithm are the following ones.

- Input:
  For each of eight blocks, the PRNG outputs: $BlockOut[t]$, $t = 1, 2, ..., N$.
- Processing: For each autonomous part of the simplified MUGI do the following
  1. employing the set of S box overdefined equations, construct the system of algebraic MQ equations based on $BlockOut[t]$, where the unknown variables are the buffer initial state bits and $a_0^{(t)}$, $a_1^{(t)}$, $t = 1, 2, ..., N$.
  2. solve the constructed system of equations
- Output:
  Recovered the buffer b and state a initial contents.

The main part of the proposed algorithm is solving the system of multivariate quadratic equations. This problem is the objective of the following section.

5.4 Methods for Solving Multivariate Quadratic Equations

The problem of solving multivariate quadratic (MQ) equations is a known and rather natural NP-hard problem. Still, little is known about the actual hardness of it.

Solving MQ with the XL Algorithm

The origin forsolving MQ equations to be discussed here was the relinearization algorithm proposed in [37]. From the relinearization algorithm, it seemed obvious that if the

system of equations is overdefned, then the problem is much easier. [37] makes an important discovery about the MQ problem: Solving it should be much easier for overdefned systems.

A new algorithm called XL, that can be seen as an improved version of relinearization is reported in [19].

It seems that for a random system of quadratic equations over GF(2) (or one that looks random) that has a unique solution, the XL method should always work (but maybe not for some very special systems). In [34] it is stated that "From the theory of Hilbert-Serre, we may deduce that the XL program will work for many interesting cases for D large enough". From [19] it seems also that XL could be subexponential, however very little is known about the actual behaviour of such algorithms for very big systems of equations.

The XL algorithm and all the basic facts about it are recalled below from [19].

How XL Works

We consider the problem of solving m quadratic equations with n variables that are in GF(2). In general, number of quadratic terms in these equations is about $t \approx n^2/2$ (but it can be less).

Let $D = 2, 3, ...$ be a parameter of the XL algorithm. What the algorithm basically does, is to multiply each possible equation $1...m$ by all possible products of $D-2$ variables. Thus we get about: $R \approx \binom{n}{D-2} m$ new equations. The total number of terms that appear in these equations is about $T = \binom{n}{D}$. We expect that most of the equations are linearly independent. Then, we pick a sufficiently big D such that

$$R = \binom{n}{D-2} m \geq \binom{n}{D} = T \ .$$

Obviously the number of linearly independent equations cannot exceed the number of terms T. We expect that if the system has a unique solution, then there is such a D for which $R \geq T$, and such that also the number $Free$ of linearly independent equations in R will be very close to T. Then if the rank deficit $T - Free$ is not too big, we expect that the system will be solved. It is easy when $T - Free$ is a very small number, but still possible when $T - Free$ is quite big. For example let T' be the number of terms out of T that contain only the first 40 variables. If $Free > T - T' + 40$, then we are able to obtain (by progressive elimination of terms) to obtain a system of 40 equations with 40 variables that can be solved by the exhaustive search. Then we fix these 40 variables and we should obtain $T - Free$ much smaller in the new system, and it will be, probably, not necessary to repeat the above "trick" with some other 40 variables. We expect that the D value for which XL works is equal or very close to the theoretical value D for which $R \geq T$. Thus, the XL algorithm is expected to succeed when:

$$R \geq T \ \ \Rightarrow \ \ m \geq \binom{n}{D} / \binom{n}{D-2} \approx n^2/D^2 \ .$$

This gives

$$D \approx \frac{n}{\sqrt{m}}$$

and the complexity of the attack is about

$$T^w \approx \binom{n}{D}^w \approx \binom{n}{n/\sqrt{m}}^w$$

which for $w \leq 3$ being the exponent of the Gaussian reduction. On the other hand, it is unclear what value w will be realistic in the attacks. From the above formula it seems that XL is subexponential, however very little is known about the actual behaviour of XL for very big systems of equations.

Unicity of the Solution

In [19], authors made many computer simulations on XL algorithm in the field GF(127). In some cases XL failed, and this is apparently due to the fact that the system had many solutions, not in the base field GF(127), but in some algebraic extension. Indeed such manipulations on the equations that are done in XL (described above): multiplying equations by monomials and combining them, conserve all the solutions in the algebraic closure of GF(127).

This is not a problem for small fields, for example GF(2). When multiplying such equations by monomials of a small degree, we will make explicit usage of the equation of the field $x_i^2 = x_i$ for each of the variable x_i, and always write x_i instead of x_i^2. Such repeated interaction with the equation of the field will eliminate all the solutions with variables being not in GF(2). Another problem with XL is that if there are many solutions, there is no simple algebraic equation that would englobe all of them, and the algorithm has to fail.

Conversely it seems that for a system of quadratic equations over a small field GF(2) (and also other GF(q) with q small), that has only one solution in the base field GF(2), the XL method will always work, except maybe for some very special systems.

XL and Sparsity

It is obvious that if in the initial system $t < n^2/2$, i.e. not all possible $n^2/2$ quadratic terms are present, XL will work better. After multiplying each of the equations $\binom{n}{D-2}$, by one of the terms, it may happen that not all the possible $\binom{n}{D}$ terms will be obtained. In this case we might obtain a strictly smaller D, for which the number of linearly independent equations will be big enough. Since the algorithm is exponential in D, lowering it even by one, will yield a dramatic improvement in the complexity. This improvement will be even better if the terms have some specific structure that will allow us to multiply them by only some selected monomials. This should be done in such away that, as much as possible different products of some monomial with some of the initial terms (i.e. present in the initial equations), should lead to identical terms of degree D. Thus we will generate many equations while maintaining the total number of terms small.

Does XL Always Work

It is important to understand that the XL algorithm will not always work. Following the XL complexity evaluation, an overdefned system of equations (big m/n) leads to a dramatic improvement in XL complexity compared to other systems with the same number of variables. Still it is easy to produce overdefined systems on which it fails. For example if we mix two systems of equations with separate sets of variables, one of which is very much overdefned, and the other of which is not, we will still obtain a largely overdefned system of equations. However applying XL will only find solutions to one of the systems, and never to the other.

Bad things may also happen when variables are linearly dependent. For example consider a system of $m = 100$ equations with $n = 100$ variables over GF(2). If we apply XL to this system we have: $D \approx n/\sqrt{m} \approx 10$ and the complexity of the XL attack is very big: about $\binom{n}{d}^{2.376} \approx 2^{104}$. Now we add just 10 additional variables that are linear combinations of the existing variables. It allows to write 10 new linear equations and to derive $10 \star n = 10 \star 100$ new quadratic equations. Everything seems correct: all these equations will be linearly independent. Now we have a new system of $m' = 1110$ quadratic equations with $n' = 200$ variables. If we naively apply XL, we get $D' \approx n'/\sqrt{m'} \approx 4$ and the complexity of the XL attack would be only: $\binom{n}{D'}^{2.376} \approx 2^{53}$. It is less than before, though our system is just the expansion of the previous system. In reality, the XL algorithm will certainly fail for this second (very special) system. The exact analysis of the complexity of XL for systems having dependent variables is not as simple anymore. For example in the relinearization technique from [37] and [19], when some variables are products of some other variables, less linearly independent equations than expected are obtained, see [19]. The relinearization algorithm still works, but not as well as XL: it seems that adding new variables that are defined as combinations of the previous variables is a bad idea. It will create more than expected linear dependencies at some further stage, see [19]. There are many questions open about XL and similar methods. In general we tend to believe that, if such methods doesn't not work, there is usually a combinatorial or algebraical reason for this, and sooner or later we will find out how to prove that it does not work. Currently it seems that (at least) these conditions should be satisfied for methods such as XL (or relinearization) to work:
1. The system should have a unique solution.
2. The variables should be "well mixed".
3. There shouldn't be possible to exhibit a subsystem and a variable change, for which the subsystem contains less terms than the expected contribution from this subsystem, to the total number of linearly independent equations.

On the other side, if we are not able to prove that the attack fails, one should assume that it may (or may not)work and should do computer simulations, that would either invalidate the claim, either give a partial confirmation.

6 A Discussion on Cryptanalysis of Full MUGI

The previous section proposes an algebraic method for cryptanalysis of the simplified MUGI based on the fact the an overdefined set of MQ equations can be constructed related to each employment of an S-box.

It can be directly shown that the method proposed for cryptanalysis of the simplified MUGI can be employed for cryptanalysis of the full MUGI because it is actually not sensitive on the following operations:
- transposition (rotation) of the unknown variables;
- linear transformation of the unknown variables;
- XOR-ing the unknown variables with the known constants.

Accordingly, the underlying problem of full MUGI cryptanalysis is the same as the underlying problem of the simplified MUGI cryptanalysis with the following main difference: The number of unknown variables which must be considered simultaneously in the case of full MUGI is much larger than in the case of the simplified MUGI because the full MUGI can not be splitted into eight autonomous parts.

- Input:
  PRNG outputs: $Out[t]$, $t = 1, 2, ..., N$.
- Processing:
  1. Employing the set of S box overdefined equations, construct the system of algebraic MQ equations baed on $Out[t]$ where the unknown variables are the buffer initial state bits and $a_0^{(t)}$, $a_1^{(t)}$, $t = 1, 2, ..., N$.
  2. Solve the system of equations constructed in the previous step.
- Output:
  Recovered the buffer b and state a initial contents.

The system of equations which should be constructed in the processing step 2 is based on the following equations.

$$a_0^{(t+1)} = a_1^{(t)}$$

$$a_1^{(t+1)} = a_2^{(t)} \oplus F(a_1^{(t)}, b_4^{(t)}) \oplus C_1 \ ,$$
$$a_2^{(t+1)} = a^{(t)} \oplus F(a_1^{(t)}, b_{10}^{(t)} <<< 17) \oplus C_2 \ .$$

C_1 and C_2 in the equations above are constants. The F-functions of MUGI reuses the components of AES: S-box and MDS.

$$b_j^{(t+1)} = b_{j-1}^{(t)} \ \ (j \neq 0, 4, 10) \ ,$$

$$b_0^{(t+1)} = b_{15}^{(t)} \oplus a_0^{(t)} \ ,$$
$$b_4^{(t+1)} = b_3^{(t)} \oplus b_7^{(t)} \ ,$$

$$b_{10}^{(t+1)} = b_9^{(t)} \oplus (b_{13}^{(t)} <<< 32) \ .$$
$$Out[t] = a_2^{(t)} \ ,$$

Obviously, the main part of the proposed algorithm for MUGI cryptanalysis is solving the system of multivariate quadratic equations. This issue is discussed in more detail bellow.

Instead of solving a system of m multivariate quadratic equations with n variables of degree $K = 2$ as it is considered in [19], it is possible to consider higher degree equations, i.e. the general case $K \geq 2$, assuming that the system is highly overdefined. Such an approach is considered in [22].

The key issue in the equations solving is manipulating the equations according to the following. We refer to the equation $l_i(x_0, ...x_{n-1}) = 0$ as simply the equation ℓ_i. Because the right hand of all the equations is always 0, it is very useful to identify a multivariate polynomial and an equation that says it is equal to 0. Thus the equation $x_0 \cdot \ell_2(x_0, ...x_{n-1}) = 0$ is referred to as the equation $x_0\ell_2$. Each solution x that satisfies all the equations ℓ_i, also satisfy the equations such as $x_i\ell_j$ and in general any linear combination of products of the form $\prod_i x_i^{r(i)}\ell_j$.

The equations of the form $\prod_{j=1}^{k} x_{ij}\ell_i = 0$, with all the i_j being pairwise different, are of type $x^k l$, and we call $x^k l$ the set of all these equations. For example the initial equations A are of type ℓ.

We also denote by x^k the set of all terms of degree exactly k, $\prod_{j=1}^{k} x_{ij}$.

Let D be the parameter of the XL algorithm. Let ℓ_i be the initial equations. The XL algorithm consists of multiplying both sides of these equations by products of variables. In a general case XL is specified as follows:

The XL algorithm, [22]:
Execute the following steps:

1. Multiply: Generate all the products $\prod_{j=1}^{k} x_{ij}\ell_i$ with $k \leq D - K$, so that the total degree of these equations is $\leq D$.

2. Linearize: Consider each monomial in the x_i of degree $\leq D$ as a new variable and perform Gaussian elimination on the equations obtained in 1.
   The ordering on the monomials must be such that all the terms containing one variable (say x_1) are eliminated last.

3. Solve: Assume that step 2 yields at least one univariate equation in the powers of x_1. Solve this equation over the finite field.

4. Repeat: Simplify the equations and repeat the process.

The main problem in the XL algorithm is that in practice not all the equations generated are independent. Let $Free$ be the exact number of equations that are linearly independent in XL. Very little is known about the exact value of Free for $D \geq 3$. It is demonstrated in [22] that XL also works very well for $K > 2$ and over GF(2). Moreover it is possible to explain the origin of the linear dependencies that appear in the XL algorithm, and thus to predict the exact value $Free$ in XL. The formula on $Free$ given in [22], though not rigorously proven to be exact, is always true in the performed simulations.

Note that the XL algorithm consists of multiplying the initial m equations ℓ_i by all possible monomials of degree up to $D-K$, so that the total degree of resulting equations is D.

Let R be the number of generated equations and T be the number of all monomials. We have, (noting that the first term is dominant):

$$R = m \cdot \sum_{i=0}^{D-K} \binom{n}{i} \approx m \cdot \binom{n}{D-K} .$$

It is likely that not all of these equations are linearly independent, and we denote by $Free$ the exact number of the independent ones. We have $Free \leq R$ and $Free \leq T$. The basic principle of XL is the following: for some D we will have $R \geq T$. Then we expect that $Free \approx T$, and obviously it cannot be bigger than T. More precisely, following [19], when $Free \geq T - D$, it is possible by Gaussian elimination, to obtain one equation in only one variable, and XL will succeed.

Obviously for XL algorithm to work it is not necessary that most of the equations are linearly independent. It is sufficient that for some D, the number $Free$ of linearly independent equations satisfies $Free \geq T - D$.

Finally, in the context of consideration the security impacts of the above discussion, recall the following. It is well known from [38] that breaking a good cipher should require "as much work as solving a system of simultaneous equations in a large number of unknowns of a complex type". This seemed very easy to achieve so far, as solving systems of equations can become intractable very easily. For example, recently, in [25] it is shown how to represent Rijndael with one big equation to solve. The equation is so big 2^{50} terms for a 128-bit cipher, that it is unlikely to produce the security consequences of Rijndael. Similarly, though every cipher can obviously be described in terms of a system of multivariate equations over GF(2), it does not mean that it can be broken.

On the other hand a number of results have been reported showing that algebraic method can be efficiently employed for cryptanalysis of certain cryptographic schemes: See, for example, recently reported results [31], [32] and [11], related to the secret key ciphers, as well as recall on the attacks that have appeared in public key cryptography - the cryptanalysis of Matsumoto-Imai cryptosystem in [36] and the attack on the basic version of HFE cryptosystem reported in [20]. In these attacks the security collapses suddenly after discovery (either theoretical or experimental) of the existence of additional (multivariate) equations, that are not obvious and have not been anticipated by the designers of the original cryptosystems.

Part III: Concluding Discussions and References

7 Security Comparison of MUGI, PANAMA and MULTI-S01

MUGI is developed as a PANAMA-like keystream generator, and accordingly, a natural question is weather MUGI suffers from the weaknesses identified in PANAMA.

Some weaknesses of PANAMA/MULTI-S01 have been reported (see [30], for example), and this report points out certain potential weakness of MUGI. Based on these results, as well as the previously reported ones, this section yields a security comparison of MUGI, PANAMA and MULTI-S01.

Recall that, in [30], an equivalent scheme of PANAMA keystream generator is developed and it points out that the buffer actually consists of 64 substructures which does not interact each with other within the buffer. This characteristic is not only undesirable, but potentially a dangerous one. The possibility for identification of 64 mutually independent substructures of the buffer is a consequence of the rotation rule employed in the buffer. Accordingly, it is pointed out that certain PANAMA like keystream generators obtained by some simplifications of original PANAMA keystream generator can be broken with complexity substantially smaller than complexity of exhaustive search over all possible secret keys. The simplified schemes considered in [30] preserve the underlying PANAMA concept including employment of a large internal memory divided into two parts, the buffer and the state which influence each other, and the same output function as original PANAMA. Two algorithms for cryptanalysis, Algorithm I and Algorithm II, are developed for cryptanalysis of the simplified PANAMA schemes, PANAMA-S1 and PANAMA-S2, respectively, are proposed in [30]. Time complexity of Algorithm I is proportional to 2^{144}, and time complexity of Algorithm II is proportional to 2^{65}. Both the algorithms require only $O(100)$ keystream generator output symbols. Finally, a modified PANAMA-SM has also been specified and cryptanalyzed in [30]. Breakability of PANAMA-SM by Algorithm III, which is a modified Algorithm II and has the same time complexity proportional to 2^{65} and requires the same sample length of $O(100)$, indicates the following: (i) certain critical elements for security of PANAMA; (ii) nature of certain weaknesses in PANAMA algorithm.

The following is pointed out in [30]. As the main critical point, an inappropriate selection of the rotation rule in the buffer is identified. Also, a very undesirable characteristic is that PANAMA-S1 which can be considered as a significantly simplified PANAMA-SM is more resistant on cryptanalysis because Algorithm I and Algorithm III have time complexities proportional to 2^{144} and 2^{65}, respectively.

Although PANAMA keystream generator is resistant against the developed algorithms for cryptanalysis of PANAMA-S1 and PANAMA-S2/PANAMA-SM, it appears that par-

ticularly Algorithm III implies a strong suggestion for certain modifications of PANAMA keystream generator, for example in a manner suggested in [30].

On the other hand, although MUGI is a PANAMA-like keystream generator its simplifications are resistant against the attacks which can work against simplified PANAMA.

Note that according to [3] the MUGI design rationale aims to make its security evaluation easier than PANAMMA. It is claimed in [3] that an 8-bit substitution table is adopted to improve the security of ρ, and in addition an extended Fiestel network is adopted in ρ instead of a simple SPN-structure, in order to simplify the evaluation.
(Also, from implementation point of view an aim of MUGI design was to achieve efficiency in hardware implementations: Particularly, a gate-efficient implementation was a design goal.)

In order to gain on security convincing, MUGI design approach was employment of two already security evaluated elements:

- PANAMA-like design concept, and
- components already used in AES: S-box and MDS.

Accordingly, the design on MUGI was based on belief that the AES S-box does not have weaknesses. On the other hand, very recently reported results [21] have disclosed certain undesirable and potentially dangerous properties of AES S-box. The underlying problem of AES S-box is that its algebraic structure enables construction of an overdefined system of the equations which relate S-box inputs and outputs. This characteristic of S-box yields an origin for deveoping certain algebraic cryptanalytic attacks on MUGI based on techniques for solving overdefined system of low degree nonlinar equations. A framework for these attacks is proposed in this report. Although the efficiency of these attacks is not provable at this moment, theirs existence is a serious warning regarding the security of MUGI.

A security comparison of MUGI, PANAMA and MULTI-S01 has to take into account significant differences between these three encryption techniques, namely the following ones:
- MUGI employes secret key of only 128 bits:
- PANAMA/MULTI-S01 employ secret keys of 256 bits.
- MUGI and PANAMA are "ordinary" stream cipher weather MULTI-S01 is a stream cipher which in parallel performs the integrity check.

Note that assuming employment of a 128-bit secret key in PANAMA, the reported results do not indicate any weakness of PANAMA nor a simplified PANAMAs considered in [30].

On the other hand, assuming that a secret key longer than 128 bits is employed in MUGI, certain security claims do not hold any more (for example see Sections 4.1.3 and 4.1.4). Also, increasing secret key length increase vulnerability of MUGI by the proposed (see Sections 5 and 6) algebraic cryptanalysis based on the overdefined systems of equations because the attacks with complexity greater than 2^{128} should be taken into account.

For example, the above implies the following:

- If an attack of provable complexity 2^{μ}, $128 < \mu < 256$, exists, it does not imply a security weakness of MUGI (because it employes a secret key of only 128 bits) but it does imply the weaknesses of PANAMA/MULTI-S01.

- MUGI and PANAMA are weak against the integrity attacks, weather MULT-S01 includes the integrity check.

On the other hand, no one of the reported results yield a provable efficient way for attacking any of these three stream ciphers, but the reported results point out warning which do not allow an unrestricted recommendation of employment of any of these cryptographic techniques.

Also note the following. The relative security issue has been addressed in regarding NESSIE project, as well. It is pointed out in [7] that when assessing cryptographic primitives designed to operate to the same security level in similar environments, it is natural to wish to compare their security - however, care has to be taken when making such comparisons. One measure that has been suggested is the security margin, but there is no general consensus about its definition or use. Furthermore, it is claimed in [7], whilst the NESSIE project tries to ensure that each submitted primitive receives equivalent cryptanalysis, it is the case that some designs are easier to analyse than others.

According to the previous statements, a precise quantitative comparison of MUGI, PANAMA and MULTI-S01 seems not possible. At this moment, as an answer to the comparison request, only certain qualitative statements can be given, and the main claims are the following ones.

- No one of the considered encryption techniques is perfect from the security point of view.

- At this moment, no one algorithm is known for efficient breakability of the considered encryption techniques, but this fact has a limited significance and definitively does not imply theirs security.

- A number of undesirable and potentially dangerous properties are reported regarding the considered encryption techniques. The reported results are a strong warning on the security estimation implying a need for careful re-evaluation of the current security statements because novel more powerful attacking attempts can be expected in the future.

- Increasing length (beyond 128) of MUGI secret key increases potential vulnerability of MUGI because certain algebraic attacks based on the overdefined systems of equations could have complexity significantly smaller than the dimension of the secret keys space.

8 Conclusions

MUGI is a stream cipher which has been very recently [3], exposed to a wide cryptographic community.

The first security evaluations of MUGI are the self-evaluation results reported in [2] and [3]. These reports claim that MUGI is resistant on all at that moment known attacks, and that it has a number of desirable properties from the security point of view. Particularly it should be noted that MUGI has a large internal memory, and it employs a relatively short secret key of 128 bits. The previous make a number of "standard" attacks related to recovering the internal state inefficient because the complexity of such an attack is greater than the complexity of simple brute force search for the secret key.

Analyzing and recognizing previous results regarding the security evaluation of MUGI reported in [2] and [3], this report yields certain novel approaches for cryptanalysis of MUGI. Previously reported results show that MUGI keystream generator passes the following security checks:
(a) consideration of design philosophy,
(b) statistical testing checks,
(c) resistance against known cryptanalytic attacks.
Sections 3 and 4 of this report yields summary of impacts of various known techniques for cryptanalysis relevant for MUGI security evaluation. Because the issues (a)-(c) have already been covered in an appropriate way (from the present knowledge point of view), this report addresses a framework for novel cryptanalytic approaches relevant for MUGI security consideration. Novel approaches relevant for the security evaluation are proposed and discussed in Sections 5 and 6 of this Report.

Accordingly, this report yields a security evaluation of MUGI based on the following:
- consideration of various known attacks against stream ciphers relevant for MUGI;
- consideration of novel approach for MUGI attacking.
- a security comparison of MUGI and similar stream ciphers PANAMA/MULTI-S01.

Security Evaluation Based on Various Known Attacks

In Sections 3 and 4, this report has re-considered various security evaluation approaches relevant for MUGI. As the first, the following general approaches were discussed: randomness tests, period, linear complexity, and standard divide-and-conquer attack.
Then, the following dedicated evaluation methods were considered:
- employment of block ciphers like evaluation methods;
- re-synchronization attacks;
- analysis of the key setup.
Particularly the above evaluation methods have included the following:
- differential/linear characteristics of main MUGI components;
- square-attack variants;
- related-key attacks;
- chosen initial values attacks;
- buffer non-linear characteristics.

It is very important to note that a number of the security claims from [2]-[3] are dependent on the secret key length: Resistance against differential and linear cryptanalysis is claimed assuming the secret key of 128 bits, but not for the longer ones.

Also, a recently proposed method in [14] for attacking a class of stream ciphers has been discussed. This method enables distinguishing some stream ciphers from a truly random process. In [14] it is, also, noted that such an attack could be mounted against MUGI, as well, but without specification of this claim. At this moment it heuristically seems that the claim is very likely correct one, but still no one precise algorithm is known as well as its impacts. Anyway, even if an algorithm can be mounted, it will give only a method for distinguishing MUGI output sequence from the random one, and very likely it will not contribute to the techniques for the secret key recovering.

Novel Approaches for MUGI Cryptanalysis

In Sections 5 and 6, this report proposes a framework for mounting the attacks against MUGI employing an algebraic method for recovering the complete initial internal state. The proposed framework is based on the following undesirable characteristic: a possibility for constructing an overdefined system of nonlinear equations which can be solved in a much MORE efficient manner than the corresponding non-overdefined system. MUGI component which opens a door for cryptanalysis is employed S-box which is the same as one employed in AES. A very undesirable property of that S-box which is the main nonlinear components of MUGI is the following one: the S-box can be described by an overdefined system of algebraic equations.

Accordingly, this report points out that MUGI seems to be vulnerable by a class of algebraic attacks based on existence of certain overdefined system of multivariate equations. The fact that solving such overdefined systems of equations of low degree, is much easier than expected, has been demonstrated [19], as a development of an earlier linearization technique proposed in [37]. The possibility to use multivariate polynomial equations approach in cryptanalysis of block ciphers was recently brought to public attention in [21].

The main open problem regarding the proposed approach for cryptanalysis based on the S-box overdefined set of equations is estimation of its complexity. At this moment a possibility for theoretical analysis of this issue is not known, and experimental analysis on the real size examples is not feasible. On the other hand some of the reported results performed on the toy-examples imply that the attacks based on the overdefined equations could be the dangerous ones against MUGI.

The proposed approach for MUGI cryptanalysis follows the same underlying idea and the results of cryptanalysis of AES reported in [21]. It can be conjucted that the weakness of AES S-boxes related to possibility of construction the sets of the overdefined equations implies more serious consequences to MUGI in comparison with AES because larger number of appropriate equations can be constricted in the case of a stream cipher in comparison with a block cipher. .

Also note the following: Employment of a longer secret keys (longer than 128 bits)

increases potential vulnerability of MUGI assuming employment of the cryptanalytic approach proposed in this report because the complexity of the proposed approach does not depend on the secret key size implying larger gain for larger secret key size assuming comparison with an exhaustive search over secret keys space.

Security Comparison of MUGI, PANAMA and MULTI-S01

The comparison issue is considered in Section 7 and the main claims are the following ones.
- No one of the considered encryption techniques is perfect from the security point of view.
- At this moment, no one algorithm is known for efficient breakability of the considered encryption techniques, but this fact has a limited significance and definitively does not imply theirs security.
- A number of undesirable and potentially dangerous properties are reported regarding the considered encryption techniques. The reported results are a strong warning on the security estimation implying a need for careful re-evaluation of the current security statements because novel more powerful attacking attempts can be expected in the future.
- Increasing length (beyond 128) of MUGI secret key increases potential vulnerability of MUGI because certain algebraic attacks based on the overdefined systems of equations could have complexity significantly smaller than the dimension of the secret keys space.

Final Security Statements

MUGI is a stream cipher very recently presented to a wider cryptographic community, and at this moment no one provable efficient method for its cryptanalysis is known.

Security evaluation of MUGI by various "standard" methods support the statement on its security and resistance against known attacks in the sense that no one of cryptanalytic attack known at the moment of its design has complexity lower than the exhaustive search over all possible secret keys assuming that the secret key is no longer than 128 bits. If a longer secret key is employed, the previous statement does not hold for all "standard" attacks, particularly not for the differential/linear cryptanalysis.

At this moment, the main undesirable property of MUGI is a possibility to associate with its S-boxes (which are the same as in AES) an overdefined system of equations. Also, this report points out a novel framework for attacking MUGI based on an algebraic approach. Although, at this moment, the efficiency of this approach is not proved, it is a strong warning towards careful reconsideration of MUGI security particularly taking into account advances in methods for efficient solving of the overdefined systems of equations.

Also, note that this report has been produced within a very limited time of only one month but it points out some previously unknown features of MUGI, and this is also a reason more why further research towards MUGI security is recommendable.

References

[1] "MUGI Pseudorandom Number Generator - Specification Ver. 1.3". Hitachi, Ltd., May 14, 2002.

[2] "MUGI Pseudorandom Number Generator - Self-evaluation Report". Hitachi, Ltd., Sept. 9, 2001.

[3] D. Watanabe, S. Furya, K. Takaragi and B. Preneel, "A new keystream generator MUGI", FSE2002, *Lecture Notes in Computer Science*, vol. 2365, pp. 179-194, 2002.

[4] K. Sakurai, "Report on Present State of MUGI Cipher Evaluation (screening evaluation)", January 28, 2002.

[5] "CRYPTREC Report 2000", Information Technology Promotion Agency (IPA), Japan, August 2001,
http://www.ipa.go.jp/security/index-e.html

[6] "New European Schemes for Signatures, Integrity and Encryption (NESSIE) Project",
http://www.cryptonessie.org.

[7] "Description of Methodology for Security Evaluation", NESSIE Public Report, NES/DOC/RHU/WP3/D10/3, 2000,
http://www.cryptonessie.org.

[8] "Update on the selection of algorithms for further investigation during the second round", NESSIE Public Report, NES/DOC/ENS/WP5/D18/1, 2002,
http://www.cryptonessie.org.

[9] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems", *Journal of Cryptology*, vol. 4, pp. 3-72, 1991.

[10] A. Biryukov and A. Shamir, "Cryptanalytic time/memory/data tradeoffs for stream ciphers", ASIACRYPT2000, *Lecture Notes in Computer Science*, vol. 1976, pp. 1-13, 2000.

[11] P. Camion, M.J. Mihaljević and H. Imai, "Two alerts for design of certain stream ciphers: Trapped LFSR and weak resilient function over GF(q)", SAC2002, to appear in *Lecture Notes in Computer Science*, (18 pages) 2002.

[12] A. Canteaut and M. Videau, "Degree of composition of highly nonlinear functions and applications to higher order differential cryptanalysis", EUROCRYPT2002, *Lecture Notes in Computer Science*, vol. 2332, pp. 518-533, 2002.

[13] P. Chose, A. Joux and M. Mitton, "Fast correlation attacks: An algorithmic point of view", EUROCRYPT2002, *Lecture Notes in Computer Science*, vol. 2332, pp. 209-221, 2002.

[14] D. Coppersmith, S. Halevi and C. Jutla, "Cryptanalysis of stream ciphers with linear masking", CRYPTO2002, *Lecture Notes in Computer Science*, vol. 2442, 2002.

[15] J. Daemen and V. Rijmen, "AES proposal: Rijndael", (the revised text), 2000, http://csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf

[16] J.Daemen, R.Govaerts, J.Vandewalle, "Resynchronization weaknesses in synchronous stream ciphers", EUROCRYPT93 , *Lecture Notes in Computer Science*, vol. 765, pp. 159-169, 1994.

[17] J. Daemen, L. Knudsen and V. Rijmen, "The block cipher Square", FSE'97, *Lecture Notes in Computer Science*, vol. 1267, pp. 159-169, 1997.

[18] J. Daemen and C. Clapp, "Fast hashing and stream encryption with PANAMA", Fast Software Encryption - FSE'98, *Lecture Notes in Computer Science*, vol. 1372, pp. 60-74, 1998.

[19] N. Courtois, A. Klimov, J. Patarin and A. Shamir, "Effcient Algorithms for solving Overdefned Systems of Multivariate Polynomial Equations", Eurocrypt'2000, *Lecture Notes in Computer Science*, vol. 1807, pp. 392-407, 2002.

[20] N. Courtois, "The security of Hidden Field Equations (HFE)", Cryptographers' Track RSA Conference 2001, *Lecture Notes in Computer Science*, vol. 2020, pp. 266-281, 2001.

[21] N.T. Courtois and J. Pieprzyk, "Cryptanalysis of Block Ciphers with Overdefined Systems of Equations", *Cryptology Eprint Archive*, report 2002/044, May 24, 2002, http://eprint.iacr.org.

[22] N.T. Courtois, "Higher order correlation attacks, XL algorithm and cryptanalysis of Toyocrypt", *Cryptology Eprint Archive*, report 2002/087, Jul 4, 2002, http://eprint.iacr.org.

[23] P. Ekdahl and T. Johansson, *SNOW - a new stream cipher*. Submitted to Nessie. Available at http://www.it.lth.se/cryptology/snow.

[24] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, D. Whiting "Improved Cryptanalysis of Rijndael", FSE 2000, *Lecture Notes in Computer Science*, vol. 1978, pp. 213-230, 2001.

[25] N. Ferguson, R. Schroeppel and D. Whiting, "A simple algebraic representation of Rijndael", SAC2001, *Lecture Notes in Computer Science*, vol. 2259, pp. 103-111, 2001.

[26] *FIPS 140-1: Security Requirements for Cryptographic Modules*. Federal Information Processing Standard (FIPS), Publication 140-1, National Institute of Standards and Technology, US Department of Commerce, Washington D.C., January 1994, available at http://www.itl.nist.gov/fipspubs/index.htm

[27] S. Halevi, D. Coppersmith and C. Jutla, "Scream: a software-efficent stream cipher", FSE2002, *Lecture Notes in Computer Science*, vol. 2365, pp. 195-209, 2002.

[28] M. Matsui, "Linear cryptanalysis method for DES cipher", EUROCRYPT93, Lecture Notes in Computer Science, vol. 765, pp.159-169, 1994.

[29] J. Menezes, P.C. van Oorschot and S.A.Vanstone, *HANDBOOK of APPLIED CRYPTOGRAPHY*. CRC Press, 1997.

[30] M.J. Mihaljević, *Report on Security Evaluation of PANAMA Stream Cipher*. Technical Report, CRYPTREC, Information-technology Promotion Agency (IPA), Government of Japan, Tokyo, Document No. 13-IPA-1060, 36 pages, Dec. 2001.

[31] M.J. Mihaljević and H. Imai, "Cryptanalysis of TOYOCRYPT-HS1 stream cipher", *IEICE Transactions on Fundamentals*, vol. E85-A, pp. 66-73, Jan. 2002.

[32] M.J. Mihaljević and R. Kohno, "Cryptanalysis of fast encryption algorithm for multimedia FEA-M", to appear in *IEEE Communications Letters*, vol. 6, Sept. 2002.

[33] M.J. Mihaljević, M.P.C. Fossorier and H. Imai, "Fast correlation attack algorithm with the list decoding and an application", FSE2001, *Lecture Notes in Computer Science*, vol. 2355, pp. 196-210, 2002.

[34] T.T. Moh, "On The Method of XL and Its Ineffciency Against TTM", *Cryptology Eprint Archive*, report 2001/047, June 6, 2001,
http://eprint.iacr.org.

[35] K. Nyberg, "Differentially Uniform Mappings for Cryptography", EUROCRYPT'93, *Lecture Notes in Computer Science*, vol. 765, pp. 55-64, 1993.

[36] J. Patarin, "Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88", CRYPTO'95, *Lecture Notes in Computer Science*, vol. 963, pp. 248-261, 1995.

[37] A. Shamir and A. Kipnis, "Cryptanalysis of the HFE Public Key Cryptosystem", CRYPTO'99, *Lecture Notes in Computer Science*.

[38] C.E. Shannon, "Communication theory of secrecy systems", *Bell System Technical Journal*, vol. 28, pp. 656-715, 1949.