Report on Security Evaluation of PANAMA Stream Cipher

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

December 17, 2001

Abstract

In this report, security of PANAMA keystream generator is considered starting from an appropriate simplified version of original PANAMA and by gradually increasing complexity of the starting version towards original one. An equivalent scheme of PANAMA keystream generator is proposed. This scheme was the main origin for developing a method for cryptanalysis of a class of PANAMA-like keystream generators called PANAMA-SM specified by a slight modification of original PANAMA. It is shown that cryptanalysis of this class of keystream generators is feasible with current technology although the considered keystream generators employ a large key of 256 bits, and huge internal memory of thousands of bits. Although, the developed method for cryptanalysis of a PANAMA-SM keystream generators can not be directly applied to PANAMA, the existence of this cryptanalytic method is an important alert related to security of original PANAMA. The features of PANAMA algorithm, disclosed in this report are, at least, undesirable and potentially dangerous. Most notably the developed cryptanalytic method indicate a strong recommendation for a modification of PANAMA in order to eliminate identified undesirable characteristics.

Report on Security Evaluation of PANAMA Stream Cipher

Content

1 Introduction

This report gives a security evaluation of PANAMA Stream Cipher based on PANAMA keystream generator algorithm according to the following contract request:

- "Investigate the security of the pseudorandom number generator PANAMA using known and unknown attacks. We welcome the evaluation for the simplified version of PANAMA."

PANAMA keystream generator is the core element of PANAMA stream cipher which is as secure as PANAMA keystream generator is. Accordingly, the goal of this report is to yield a security evaluation of PANAMA keystream generator [2] of stream cipher PANAMA based on security evaluation of certain PANAMA-like algorithms.

Executive Summary of the Report Results

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

Following the previous statement and the above contract request, security evaluation of PANAMA keystream generator is considered starting from an appropriate simplified version of PANAMA and by gradually increasing complexity of the starting initial version towards original PANAMA.

Main results of this report are the following:

- an equivalent representation of PANAMA keystream generator which opens a door for more detail security evaluation; this equivalent representation points out an undesirable feature that a part of PANAMA algorithm can be splitted into a number of mutually independent entities;
- development of algorithms for cryptanalysis of three PANAMA-like keystream generators obtained by certain simplifications or modification of original PANAMA: two of them are obtained by excluding certain operations from PANAMA algorithm, and the third one is a variant of PANAMA which contains all PANAMA components but two of them are modified (not simplified);
- more insight into PANAMA keystream generator via identification of critical points for PANAMA security;
- proposal for modifications of PANAMA keystream generator in order to remove security dangerous points of PANAMA.

Although the developed method for cryptanalysis of a class of PANAMA-like keystream generators can not be directly applied to PANAMA, the existence of this method is an important alert related to security of PANAMA.

The features of PANAMA algorithm, disclosed in this report, are at least undesirable and potentially dangerous, and accordingly, appropriate modifications suggested in this report are strongly recommendable, as well as further security evaluation of PANAMA.

Organization of the Report

The report is organized as follows. Section 2 summarizes the underlying approach for security evaluation of PANAMA keystream generator and the previously reported results on its security. Section 3 yields an overview of PANAMA keystream generator in order to avoid possible confusions. In Section 4 an equivalent scheme of PANAMA keystream generator is proposed which is important for further security analysis. Two simplified PANAMA keystream generators are specified and cryptanalyzed in Sections 5 and 6, where two algorithms for cryptanalysis, Algorithm I and Algorithm II, are proposed as well. Section 7 points out that a variant of PANAMA keystream generator which could be considered as a slight modification of original PANAMA, can be cryptanalyzed by the developed Algorithm III. Implications of the developed methods to security of PANAMA are discussed in Section 8 together with a proposal for PANAMA modification in order to strengthen its security characteristics. Summary of this report results and related conclusions are given in Section 9. A general discussion on approaches for security evaluation of stream ciphers is given in Appendix A, and a number of elements specific for PANAMA is summarized in Appendix B.

2 Security Evaluation Preliminaries and Reported Results

This section summarizes methodological issues relevant for security evaluation of PANAMA and the previously reported results on its security evaluation.

2.1 Security Evaluation Preliminaries

According to the current criteria (see [9] and [6]), 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 my 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 including specific issues related to a particular stream cipher. A more detailed consideration of this issue is given in Appendix A.

Finally note the following.

• 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.

• 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.

2.2 Previously Reported Results on Security Evaluation and Motivation for Further Work

The very first statements related to security evaluation are given as a part of PANAMA proposal [2]. Later on MULTI-S01 and PANAMA were objects of the security self-evaluation [3] and an external evaluation for CRYPTREC [4]. Very recently, certain results related to security of PANAMA hash function are reported in [5].

The statements given in [2] contain a number of mainly heuristic arguments supporting the claim of PANAMA security as a hash function and keystream generator. (For details see Appendix B.)

Self-evaluation of MULTI-S01 security, [3], yields a number of results related to security of MULTI-S01 enciphering, assuming that PANAMA is a secure keystream generator, as well as security evaluation of PANAMA keystream generator itself. It is pointed out in [3] that MULTI-S01 is secure assuming that PANAMA keystream generator is secure, and security of PANAMA keystream generator has been evaluated employing the following:
- randomness test
- difference propagation and linear correlation
- nonlinearity
- resistance against linear cryptanalysis
- attacks on certain simplified (reduced) PANAMA.

MULTI-S01 evaluation [4] yields a consideration of the security mainly against known general methods applicable to MULTI-S01. This evaluation contains the following.
- Structural aspects including: MULTI-S01 keystream combining operations, PANAMA structural aspects, equivalent keys, mode of operations and key management, bit dependences of PANAMA algorithm in stream cipher mode, and processing technique for a long message.
- Keystream properties including: period, linear complexity, and statistical analysis.
- Possible attacks including: simple attacks, divide and conquer attacks, correlation attacks, linear cryptanalysis, differential cryptanalysis, some other general attacks, and attacks on the integrity check.

Based on these analysis, [4] claims the following two basic flaws in MULTI-S01:
- Lack of security robustness - the security is compromised under minor violations of the key management rules; it is vitally important that implementations of MULTI-S01 adhere strictly to the key management rules that require a new key to be used for every encryption.
- Attacks on integrity checks - with knowledge of the key, it is a simple task to find two messages which produce the same integrity check which is a serious flaw in integrity check mechanism.

Very recently, a weakness of PANAMA hash function has been reported on [5]. A method for producing collisions is proposed. It is shown that complexity of finding a collision is 2^{82} which is significantly smaller than we can expect based on generic birthday paradox attack. Also, the suggestions for PANAMA modification in order to make it resistant against the developed attack are proposed in [5].

According to the reported results on security evaluation of PANAMA and MULTI-S01, we can specify following statements which points out a direction for further security evaluation.

• All the reported results support the claim that it is not known a cryptanalytic method for PANAMA keystream generator more efficient than exhaustive search over all possible keys.

• The reported results in certain extent address all the general methodological issues for the security evaluation discussed in Section 2.1, noting the following: The reported results on the security evaluation mainly cover security against general techniques for cryptanalysis, and only partially against specialized methods related to cryptanalysis of PANAMA-like keystream generators.

• Because the previously reported results do not contradict the statement that PANAMA keystream generator is resistant against presently known general methods for cryptanalysis, and because resistance of simplified PANAMA keystream generators is not enough considered yet, this report focuses its attention towards methods for cryptanalysis of PANAMA-like keystream generators with an intention to yield more insight of PANAMA security.

3 PANAMA Overview

In order to avoid the confusions due to certain missing or imprecise statements (for example, the keystream initialization and some formulas in algorithm description) of PANAMA keystream generator in [1], this section and Appendix B yield a specification of PANAMA keystream generator which is under consideration of this report. This specification is based on [2] where PANAMA has been proposed.

PANAMA keystream generator is based on a finite state machine with 544-bit *state* and a 8192-bit *buffer*. The state and buffer can be updated by performing an *iteration*.

There are two modes for the iteration function. A *Push* mode, that allows to inject an input and generates no output, and a *Pull* mode that takes no input and generates an output. A *blank* Pull iteration is a Pull iteration in which the output is discarded.

The PANAMA keystream generator is initialized by doing two Push iterations to inject the key and diversification parameter followed by a number of blank Pull iterations to allow the key and parameter to be diffused into the buffer and state. After this initialization, the scheme is ready to generate keystream bits at leisure by performing Pull iterations.

A block scheme of PANAMA keystream generator in the keystream sequence generation mode (Pull mode), as it is proposed in [2] is displayed in Fig. 1.

Also note that PANAMA keystream generator could be considered as a generalized nonlinear filter keystream generator with time varying filter function and the feedback.

PANAMA employes 32-bit words.

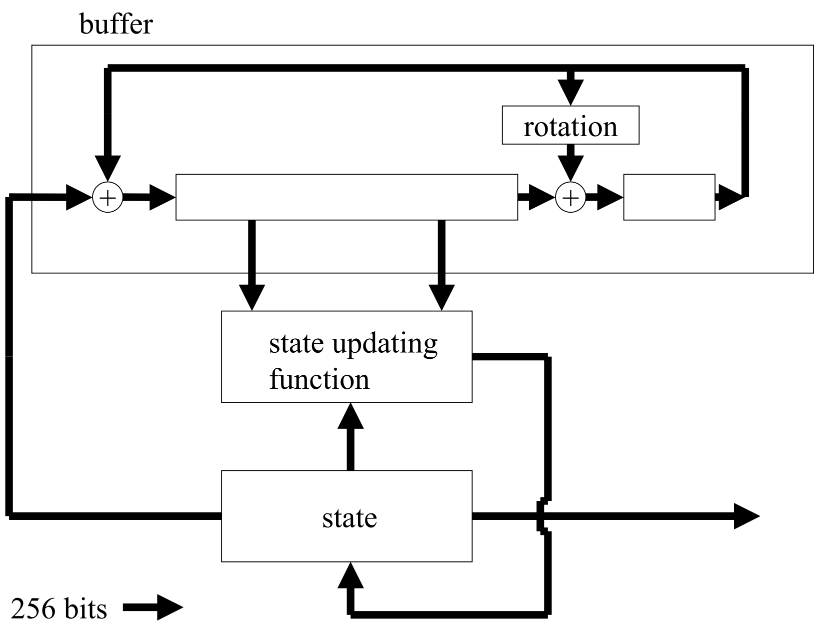

Figure 1: A block scheme of PANAMA keystream generator.

The state is denoted by a and consists of 17 words a_0 to a_{16}.

The buffer b is a linear feedback shift register with 32 *stages*, each consisting of 8 words. An 8-word stage is denoted by b^i, $i = 0, 1, ..., 31$, and its words $b^i_j, j = 0, 1, ..., 7$. Note that, both stages and words, are indexed starting from 0.

The three possible *modes* of the PANAMA module are Reset, Push and Pull. In Reset mode the state and buffer are set to 0. In Push mode an 8-word input is applied and there is no output. In Pull mode there is no input and an 8-word output is delivered. In a variant of Pull mode, blank Pull, the prepared output is discarded - not delivered.

The Buffer Updating

The buffer update operation is denoted by λ. Assuming that $d = \lambda(b)$ we have:

$$d^i = b^{i-1} \quad if \;\; i \neq 0, 25 \; ,$$

$$d^0 = b^{31} \oplus q \; ,$$

$$d^{25}_j = b^{24}_j \oplus b^{31}_{(j+2) mod 8} \;\; for \;\; 0 \leq j \leq 7 \; .$$

In Push mode q is the input block p, and in Pull mode it is part of the state a with its 8 component words given by

$$q_j = a_{j+1} \; , \;\; j = 0, 1, ..., 7 \; .$$

The State Updating

The state update operation is denoted by ρ. It is composed of four specific transformations:

$$\rho = \sigma \circ \theta \circ \pi \circ \gamma \ .$$

Here $\circ$ denotes the composition of transformations where the right-most transformation is executed first.

• γ is an invertible nonlinear transformation defined by:

$$c = \gamma(a) \ : \ \ c_j = a_j \oplus (a_{j+1} \text{ OR } \bar{a}_{j+2}) \ \ , \ \ j = 0, 1, ..., 16,$$

with the indices taken modulo 17.

• π is a permutation-class transformation which combines word shifts and a permutation of the word positions. If we we define τ_k to be a rotation over k positions from LSB to MSB, we have:

$$c = \pi(a) \ : \ \ c_i = \tau_k(a_j) \ ,$$

where

$$j = (7i) mod 17$$

$$k = (i(i+1)/2) mod 32 \ .$$

• θ is an invertible linear transformation defined by:

$$c = \theta(a) \ : \ \ c_j = a_j \oplus a_{j+1} \oplus a_{j+4} \ , \ \ j = 0, 1, ..., 16 \ ,$$

with the indices taken modulo 17.

• σ is a transformation which performs bitwise addition of the buffer and state words. Assuming $c = \sigma(a)$, the transformation σ is defined by:

$$\begin{array}{llll} c_0 & a_0 & \oplus & 00000001_{hex}, & \\ c_{j+1} & a_{j+1} & \oplus & \ell_j \ , & 0 \le j \le 7, \\ c_{j+9} & a_{j+9} & \oplus & b_j^{16} \ , & 0 \le j \le 7. \end{array}$$

In the Push mode ℓ corresponds with the input p, and in the Pull mode $\ell = b^4$.

The Output Function

The output function is such that after each iteration the generator output are the state elements a_9 - a_{16}.

4 An Equivalent Representation of PANAMA Keystream Generator

For all further considerations, we assume the following notation:

- $b^i_{j,k}(t)$ denotes kth bit of jth word of the buffer stage i in tth instant of time, i.e., after tth updating iteration;
- $a_{j,k}(t)$ denotes kth bit of jth word of the state in tth instant of time, i.e. after tth updating iteration.

Accordingly, the buffer updating transformation during a keystream generation phase, for each $k = 0, 1, ..., 31$, can be put into the following form.

$$b^i_{j,k}(t+1) = b^{i-1}_{j,k}(t) \quad if \quad i \neq 0, 25 \ , \ \ j = 0, 1, ..., 7 \ , \tag{1}$$

$$b^0_{j,k}(t+1) = b^{31}_{j,k}(t) \oplus a_{j+1,k}(t) \ , \ \ j = 0, 1, ..., 7 \ , \tag{2}$$

$$\begin{aligned} b^{25}_0(t+1) &= b^{24}_{0,k}(t) \oplus b^{31}_{2,k}(t) \ , \\ b^{25}_2(t+1) &= b^{24}_{2,k}(t) \oplus b^{31}_{4,k}(t) \ , \\ b^{25}_4(t+1) &= b^{24}_{4,k}(t) \oplus b^{31}_{6,k}(t) \ , \\ b^{25}_6(t+1) &= b^{24}_{6,k}(t) \oplus b^{31}_{0,k}(t) \ , \end{aligned} \tag{3}$$

$$\begin{aligned} b^{25}_1(t+1) &= b^{24}_{1,k}(t) \oplus b^{31}_{3,k}(t) \ , \\ b^{25}_3(t+1) &= b^{24}_{3,k}(t) \oplus b^{31}_{5,k}(t) \ , \\ b^{25}_5(t+1) &= b^{24}_{5,k}(t) \oplus b^{31}_{7,k}(t) \ , \\ b^{25}_7(t+1) &= b^{24}_{7,k}(t) \oplus b^{31}_{1,k}(t) \ . \end{aligned} \tag{4}$$

So, it is directly evident that for each $k = 0, 1, ..., 31$, we can identify two separate substructure in the buffer.

Based on the previous, an equivalent representation of PANAMA keystream generator is displayed in Fig. 2 and Fig. 3.

This equivalent representation of PANAMA keystream generator points out an undesirable characteristic of PANAMA that the buffer can be splitted into 64 separate parts which do not interact each with others within the buffer.

Finally note the following:

- This possibility for the separation is dominantly a consequence the employed rotation 32-words state (256 bits) for 2 words (16 bits), noting that 2 is a factor of 32.

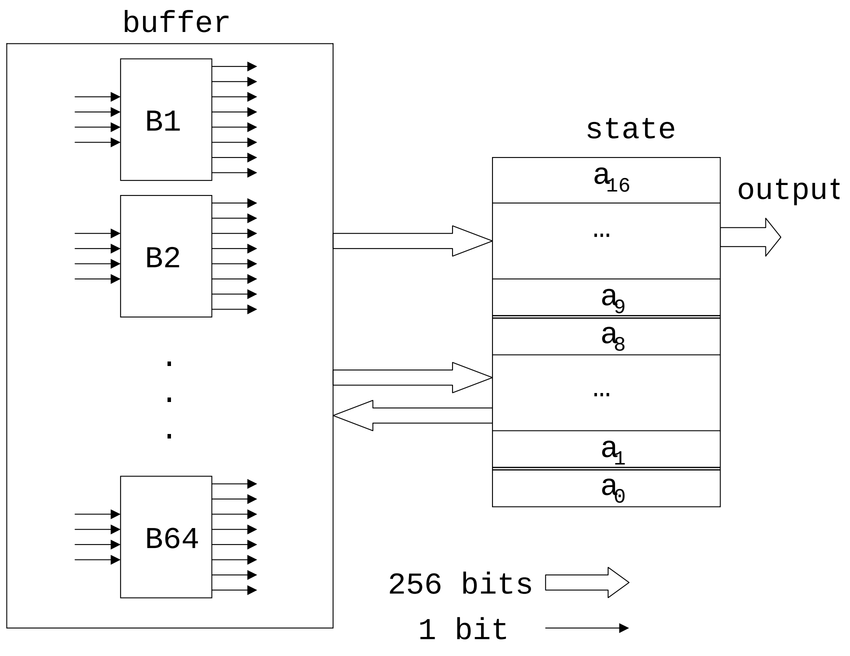

Figure 2: An equivalent macro representation of PANAMA.

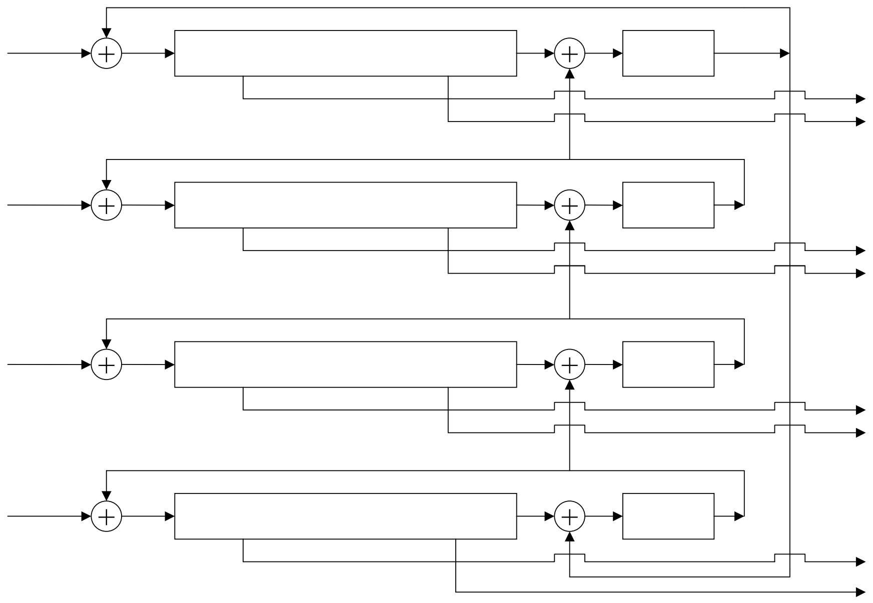

Figure 3: Structure of the B-blocks from Fig 2.

5 Cryptanalysis of PANAMA-S1: A Simplified PANAMA Keystream Generator

5.1 Specification of PANAMA-S1

PANAMA-S1 is a simplified PANAMA keystream generator obtained from the original one by the following simplifications:
- reducing the state updating function by excluding two components of its composition;
- excluding all the the rotation operations.

Accordingly, PANAMA-S1 has identical overall structure as PANAMA, i.e. it employes the same buffer and state, but theirs updating operations are simplified, and they are as follows:

- the buffer update operations

$$b_j^i(t+1) = b_j^{i-1}(t) \ \ , \ \ i \neq 0, 25 \ \ , \ \ j = 0, 1, ..., 7 \ \ ,$$

$$b_j^0(t+1) = b_j^{31}(t) \oplus a_j(t) \ \ , \ \ j = 0, 1, ..., 7 \ \ ,$$

$$b_j^{25}(t+1) = b_j^{24}(t) \oplus b_j^{31}(t) \ \ .$$

- the state update operations

$$\rho^* = \sigma \circ \pi^* \ \ ,$$

$$\pi^* : \quad a_j^*(t) = a_{(7j) mod 17}(t) \ \ .$$

Note that PANAMA-S1 preserves the underlying PANAMA concept including employment of a large internal memory devided into two parts, the buffer and the state which influence each other.

5.2 Underlying Ideas for Cryptanalysis

It can be directly shown that PANAMA-S1 can be split in 256 bit-layers of the structure displayed in Fig. 4, noting that 8 different buffer bit-layers share the same state bit-layer.

Recall that the employed transformation π^* performs the permutation of the state elements according to the following:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	7	14	4	11	1	8	15	5	12	2	9	16	6	13	3	10

Also note that in PANAMA-S1, the state element a_0 does not influence any other element of the state or buffer, so it can be neglected, and state can be considered as consisting of only 16 elements $[a_j]_{j=1}^{16}$.

Accordingly, knowing PANAMA-S1 outputs $[a_j(t)]_{j=9}^{16}$, $t = 1, 2, ..., n$, for each bit-layer k, $k = 0, 1, ..., 31$, we have the following

$$b_{1,k}^0(t+1) = a_{14,k}(t) \oplus b_{1,k}^4(t) \oplus b_{1,k}^{31}(t) \tag{5}$$

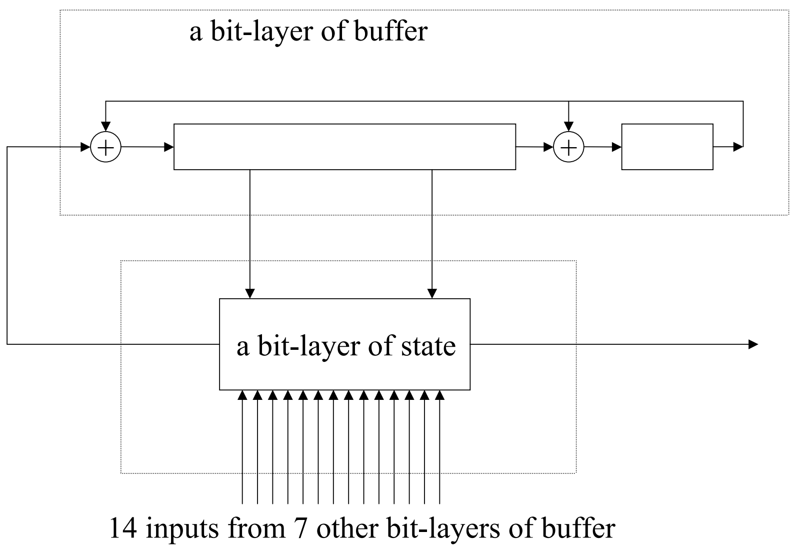

Figure 4: Block scheme of a bit-layer substructure in PANAMA-S1.

$$b^{0}_{3,k}(t+1) = a_{11,k}(t) \oplus b^{4}_{3,k}(t) \oplus b^{31}_{3,k}(t) \tag{6}$$

$$b^{0}_{6,k}(t+1) = a_{15,k}(t) \oplus b^{4}_{6,k}(t) \oplus b^{31}_{6,k}(t) \tag{7}$$

$$b^{16}_{0,k}(t) = a_{12,k}(t) \oplus a_{9,k}(t+1) \tag{8}$$

$$b^{16}_{2,k}(t) = a_{9,k}(t) \oplus a_{11,k}(t+1) \tag{9}$$

$$b^{16}_{3,k}(t) = a_{16,k}(t) \oplus a_{12,k}(t+1) \tag{10}$$

$$b^{16}_{5,k}(t) = a_{13,k}(t) \oplus a_{14,k}(t+1) \tag{11}$$

$$b^{16}_{7,k}(t) = a_{10,k}(t) \oplus a_{16,k}(t+1) \tag{12}$$

The previous relationships are origins for developing the algorithm for cryptanalysis of PANAMA-S1 proposed in the next section.

5.3 Algorithm for Cryptanalysis of PANAMA-S1

Algorithm I

1. *INPUT*:
   - PANAMA-S2 outputs $[a_j(t)]_{j=9}^{16}$, $t = 1, 2, ..., n$;

2. *INITIAL CALCULATION and INITIALIZATION*:

   (a) For $t = 1, 2, ..., n$ and $k = 0, 1, ..., 31$, calculate:
   $b_{0,k}^{16}(t) = a_{12,k}(t) \oplus a_{9,k}(t+1)$
   $b_{2,k}^{16}(t) = a_{9,k}(t) \oplus a_{11,k}(t+1)$
   $b_{3,k}^{16}(t) = a_{16,k}(t) \oplus a_{12,k}(t+1)$
   $b_{5,k}^{16}(t) = a_{13,k}(t) \oplus a_{14,k}(t+1)$
   $b_{7,k}^{16}(t) = a_{10,k}(t) \oplus a_{16,k}(t+1)$

   (b) For $i = 0, 1, ..., 24$ and $k = 0, 1, ..., 31$, set:
   $\hat{b}_{0,k}^{24-i}(0) = b_{0,k}^{16}(i+1)$
   $\hat{b}_{2,k}^{24-i}(0) = b_{2,k}^{16}(i+1)$
   $\hat{b}_{3,k}^{24-i}(0) = b_{3,k}^{16}(i+1)$
   $\hat{b}_{5,k}^{24-i}(0) = b_{5,k}^{16}(i+1)$
   $\hat{b}_{7,k}^{24-i}(0) = b_{7,k}^{16}(i+1)$

   (c) For $k = 0, 1, ..., 31$, set:
   $\hat{a}_{j,k}(0) = a_{j,k}(0)$, $j = 9, 10, ..., 16$.

3. *HYPOTHESIS TESTING*:
   For each $k = 0, 1, ..., 31$, do the following

   (a) *Setting a Working Hypothesis*
   Set a previously unconsidered hypothesis on:
   - $\hat{b}_{j,k}^{i}(0)$, $i = 25, 26, ..., 31$, $j = 0, 2, 3, 5, 7$;
   - $\hat{b}_{j,k}^{i}(0)$, $i = 0, 1, ..., 31$, $j = 1, 4, 6$;
   - $\hat{a}_{j,k}(0)$, $j = 1, 2, ..., 8$.

   (b) *Generation of the Corresponding Output*
   Running the bit-layer initialized according to algorithm steps 2(b), 2(c) and 3, calculate:
   $\hat{a}_{j,k}(t)$, $t = 1, 2, ..., n - 25$.

   (c) *Hypothesis Check*

      i. if $\hat{a}_{j,k}(t) = a_{j,k}(t+1)$, $t = 1, 2, ..., n - 25$,
      memorize the working hypothesis on initial contents of the buffer and state bit-layer as the correct one and proceed with consideration of the next bit-layer, i.e. next index k value.

      ii. if $\hat{a}_{j,k}(t) \neq a_{j,k}(t+1)$, for some $t = 1, 2, ..., n - 25$,
      check the next hypothesis.

4. *OUTPUT*: Recovered the state and buffer contents based on data memorized in Step 3(c).

5.4 Complexity of Cryptanalysis

Proposition 1. Assuming enough large n so that each cycle of the hypothesis checks yields an unique solution, the time complexity of PANAMA-S1 cryptanalysis is proportional to 2^{144}.

Proof. Algorithm I structure directly implies that its time complexity is proportional to the number of which have to be tested. Note that a same procedure should be repeated $32 = 2^5$ times because the algorithm performs the bit layer-by-layer recovering. The algorithm Step 3(a) implies that recovering of a bit-layer is proportional to testing $2^{5 \cdot 7} \cdot 2^{3 \cdot 32} \cdot 2^8 = 2^{139}$ hypothesis. Accordingly, we have the proposition statement.

Proposition 2. Sample length n required for cryptanalysis of PANAMA-S1 is $O(100)$.

Proof. Note that we need 25 PANAMA-S1 output symbols for the initialization in Step 2(b). Note that each hypothesis check in Algorithm I Step 3(c) assumes testing of a candidate for solution of a system of nonlinear equations with $5 \cdot 7 + 3 \cdot 32 + \cdot 8 = 139$ unknown binary variables due to the hypothesis set in Step 3(a). So we need 139 keystream generator outputs to have a system of 139 equations with 139 unknown binary variables. Taking into account that we need a unique solution for each bit-layer testing, we obtain the proposition statement.

6 Cryptanalysis of PANAMA-S2: A Simplified PANAMA Keystream Generator

6.1 Specification of PANAMA-S2

PANAMA-S2 is a simplified PANAMA keystream generator obtained from the original one according to the following. PANAMA-S2 differs from the original one only in the transformation ρ structure: Instead of original ρ transformation

$$\rho = \sigma \circ \theta \circ \pi \circ \gamma$$

PANAMA-S2 employs the following simplified transformation ρ^*,

$$\rho^* = \sigma \circ \pi \ . \tag{13}$$

Note that the excluded transformations γ and θ operate only over a bit-layer, and that the key transformation which operates over different bit-layers π is included.

6.2 Underlying Ideas for Cryptanalysis

Recall that the employed transformation π includes the permutation of the state elements according to the following:

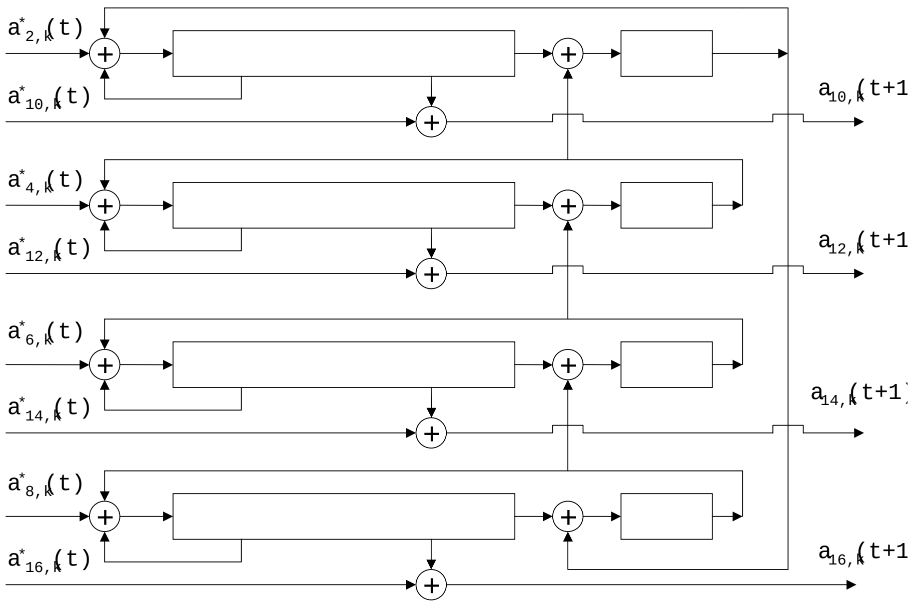

Figure 5: Block scheme of the kth substructure of TYPE 1 in PANAMA-S2.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	7	14	4	11	1	8	15	5	12	2	9	16	6	13	3	10

Note that for any t, the state elements $a_9(t)$, $a_{10}(t)$, $a_{11}(t)$, $a_{12}(t)$, $a_{13}(t)$, $a_{14}(t)$, $a_{15}(t)$ and $a_{16}(t)$, to which the transformation π will be applied, are known because they are the state components which have been outputed in the previous time instant. So, after the transformation π is performed in the tth time instant, we know the following eight state elements:

$a^*_2(t)$:	9	bits rotated $a_{14}(t)$
$a^*_4(t)$:	2	bits rotated $a_{11}(t)$
$a^*_7(t)$:	24	bits rotated $a_{15}(t)$
$a^*_9(t)$:	26	bits rotated $a_{12}(t)$
$a^*_{11}(t)$:	3	bits rotated $a_9(t)$
$a^*_{12}(t)$:	28	bits rotated $a_{16}(t)$
$a^*_{14}(t)$:	18	bits rotated $a_{13}(t)$
$a^*_{16}(t)$:	16	bits rotated $a_{10}(t)$

So, taking into account Fig. 2 and Fig. 3, we can identify in PANAMA-S2 two types, TYPE 1 and TYPE 2, of mutually independent substructures displayed in Fig. 5 and Fig. 6, respectively.

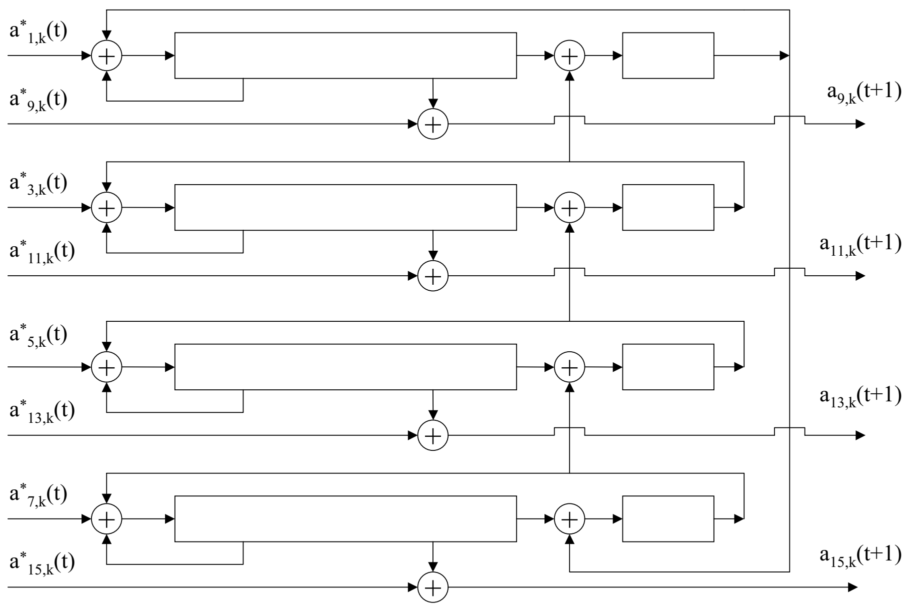

Figure 6: Block scheme of the kth substructure of TYPE 2 in PANAMA-S2.

Accordingly, as the starting point for cryptanalysis, note that in Fig. 5, in each time instant t the following elements are known:
• inputs $a^*_2(t)$, $a^*_4(t)$, $a^*_{12}(t)$, $a^*_{14}(t)$, $a^*_{16}(t)$ and
• all the outputs.

Following the previous statements, the developed algorithm for cryptanalysis consists of the following two phases:

- For each of 32 bit-layers, cryptanalyze the substructure of TYPE 1 in PANAMA-S2 based on the known inputs and outputs of the nonautonomous automata from Fig. 5 and recover the register contents and unknown inputs.
- For each of 32 bit-layers, cryptanalyze the substructure of TYPE 2 in PANAMA-S2 based on the recovered information from the previous phase and the known inputs and outputs of the nonautonomous automata from Fig. 6, and recover the register contents and unknown inputs.

6.3 Algorithm for Cryptanalysis of PANAMA-S2

Algorithm II

- *INPUT*:
  - PANAMA-S2 outputs $[a_j(t)]_{j=9}^{16}$, $t = 1, 2, ..., n$;

- *TWO PHASES PROCESSING BASED ON HYPOTHESIS TESTING*:

1. *PHASE I*

   For each bit level $k = 0, 1, ..., 31$, do the following.

   (a) *Initial Recalculation*
   - For $j = 12, 14, 16$, calculate $b_{j-9}^{16}(t)$, $t = 1, 2, ..., n-1$,

   $$b_{j-9}^{16}(t) = (rot_{(j(j+1)/2)mod32}\ a_{(7j)mod17}(t)) \oplus a_j(t+1)\ \ .$$

   - For $j = 2, 4$, calculate $a_j^*(t)$, $t = 1, 2, ..., n-1$,

   $$a_j^*(t) = (rot_{(j(j+1)/2)mod32}\ a_{(7j)mod17}(t))\ \ .$$

   (b) *Setting a Working Hypothesis*
   1. For $j = 3, 5, 7$, set: $\hat{b}_{j,k}^{24-i}(25) = b_{j,k}^{16}(i+1)$, $i = 0, 1, ..., 24$.
   2. Set a previously unconsidered hypothesis on:
      - $\hat{b}_{1,k}^{i}(25)$, $i = 0, 1, ..., 24$,
      - $\hat{b}_{j,k}^{i}(25)$, $i = 25, 26, ..., 31$, $j = 1, 3, 5, 7$.

   (c) *Recalculation*
   For $t = 25, 26, ..., n-1$, do the following:
   1. for $j = 1, 3$, update the buffer bit-layer (j, k),
      $\hat{b}_{j,k}^{i}(t+1) = \hat{b}_{j,k}^{i-1}(t)$ if $i \neq 0, 25$,
      $\hat{b}_{j,k}^{0}(t+1) = \hat{b}_{j,k}^{31}(t) \oplus a_{(j+1),k}^{*}(t)$,
      $\hat{b}_{j,k}^{25}(t+1) = \hat{b}_{j,k}^{24}(t) \oplus \hat{b}_{((j+2)mod8),k}^{31}(t)$.

   2. for $j = 5, 7$, update the buffer bit-layer (j, k),
      $\hat{b}_{j,k}^{i}(t+1) = \hat{b}_{j,k}^{i-1}(t)$ if $i \neq 0, 25$,
      $\hat{b}_{j,k}^{0}(t+1) = b_{j,k}^{16}(t+1)$,
      $\hat{b}_{j,k}^{25}(t+1) = \hat{b}_{j,k}^{24}(t) \oplus \hat{b}_{((j+2)mod8),k}^{31}(t)$.

(d) *Hypothesis Check*

i. if $\hat{b}^0_{3,k}(t) = b^{16}_{3,k}(t)$, $t = 26, 27, ..., n$, do the following:

A. for $j = 6, 8$, calculate

$a_{j,k}(t) = \hat{b}^0_{(j-1),k}(t+1) \oplus \hat{b}^{31}_{(j-1),k}(t)$, $t = 25, 26, ..., n-1$

B. memorize

$\hat{b}^i_{j,k}(t)$, $i = 0, 1, ..., 31$, $j = 1, 3, 5, 7$, $t = 51$;

$a_{j,k}(t)$, $j = 6, 8$, $t = 51$.

ii. if $\hat{b}^0_{3,k}(t) \neq b^{16}_{3,k}(t)$, for some $t = 26, 27, ..., n$, check the next hypothesis.

2. *PHASE II*

For each bit level $k = 0, 1, ..., 31$, do the following.

(a) *Initial Recalculation*

Using the input data and the memorized data from Phase I, do the following.

- For $j = 9, 11, 13$, calculate $b^{16}_{j-9}(t)$, $t = 26, 27, ..., n-1$,

$$b^{16}_{j-9}(t) = (rot_{(j(j+1)/2)mod32}\ a_{(7j)mod17}(t)) \oplus a_j(t+1) \quad .$$

- For $j = 3, 7$, calculate $a^*_j(t)$, $t = 26, 27, ..., n-1$,

$$a^*_j(t) = (rot_{(j(j+1)/2)mod32}\ a_{(7j)mod17}(t)) \quad .$$

(b) *Setting a Working Hypothesis*

i. For $j = 0, 2, 4$, set: $\hat{b}^{24-i}_{j,k}(50) = b^{16}_{j,k}(i+26)$, $i = 0, 1, ..., 24$.

ii. Set a previously unconsidered hypothesis on:

- $\hat{b}^i_{6,k}(50)$, $i = 0, 1, ..., 24$,
- $\hat{b}^i_{j,k}(50)$, $i = 25, 26, ..., 31$, $j = 0, 2, 4, 6$.

(c) *Recalculation*

For $t = 50, 51, ..., n-1$, do the following:

i. for $j = 3, 7$, update the buffer bit-layer (j, k),

$\hat{b}^i_{j,k}(t+1) = \hat{b}^{i-1}_{j,k}(t)$ if $i \neq 0, 25$,

$\hat{b}^0_{j,k}(t+1) = \hat{b}^{31}_{j,k}(t) \oplus a^*_{(j+1),k}(t)$,

$\hat{b}^{25}_{j,k}(t+1) = \hat{b}^{24}_{j,k}(t) \oplus \hat{b}^{31}_{((j+2)mod8),k}(t)$.

ii. for $j = 0, 5$, update the buffer bit-layer (j, k),

$\hat{b}^i_{j,k}(t+1) = \hat{b}^{i-1}_{j,k}(t)$ if $i \neq 0, 25$,

$\hat{b}^0_{j,k}(t+1) = b^{16}_{j,k}(t+1)$,

$\hat{b}^{25}_{j,k}(t+1) = \hat{b}^{24}_{j,k}(t) \oplus \hat{b}^{31}_{((j+2)mod8),k}(t)$.

(d) *Hypothesis Check*

i. if $\hat{b}^0_{2,k}(t) = b^{16}_{2,k}(t)$, $t = 51, 52, ..., n$, do the following:

A. for $j = 0, 5$, calculate
$a_{j,k}(t) = \hat{b}^0_{(j-1),k}(t+1) \oplus \hat{b}^{31}_{(j-1),k}(t)$, $t = 51$

B. memorize
$\hat{b}^i_{j,k}(t)$, $i = 0, 1, ..., 31$, $j = 1, 3, 5, 7$, $t = 51$;
$a_{j,k}(t)$, $j = 6, 8$, $t = 51$.

ii. if $\hat{b}^0_{3,k}(t) \neq b^{16}_{3,k}(t+16)$, for some $t = 51, 52, ..., n$, check the next hypothesis.

- *OUTPUT*:
Recovered complete contents of the state and buffer based on the data memorized in the step (d).i.B of Phase I and Phase II.

6.4 Complexity of Cryptanalysis

Proposition 3. Assuming enough large n so that each cycle of the hypothesis checks yields an unique solution, the time complexity of PANAMA-S2 cryptanalysis is proportional to 2^{65}.

Proof. Algorithm II structure directly implies that its time complexity is proportional to the number of hypothesis which have to be tested. Note that a same procedure should be repeated $32 = 2^5$ times because the algorithm performs the bit layer-by-layer recovering. The algorithm steps (b)ii of Phase I and Phase II imply that recovering of a bit-layer is proportional to testing $2^{5 \cdot 7} \cdot 2^{25} = 2^{60}$ hypothesis. Accordingly, we have the proposition statement.

Proposition 4. Sample length n required for cryptanalysis of PANAMA-S2 is $O(100)$.

Proof. Note that we need 25+25=50 PANAMA-S2 output symbols for the initialization in step (b)i of Phase I and Phase II. Note that each hypothesis check in Algorithm II steps (d)i of Phase I and Phase II assume testing of a candidate for solution of a system of nonlinear equations with $5 \cdot 7 + 25 = 60$ unknown binary variables due to the hypothesis set in the steps (b)ii of Phase I and Phase II. So we need 60 keystream generator outputs to have a system of 60 equations with 60 unknown binary variables. Taking also into account that we need a unique solution for each bit-layer testing, we obtain the proposition statement.

7 Cryptanalysis of PANAMA-SM: A "Slightly Modified" PANAMA Keystream Generator

7.1 Specification of PANAMA-SM

PANAMA-SM could be considered as a "slightly modified" PANAMA keystream generator obtained from the original one according to the following. PANAMA-SM differs from the original one only in the transformation ρ: Instead of original ρ transformation

$$\rho = \sigma \circ \theta \circ \pi \circ \gamma$$

PANAMA-SM employs the following modified ρ^* transformation

$$\rho^* = \sigma \circ \theta^* \circ \pi \circ \gamma^* \ , \tag{14}$$

where γ^* and θ^* are the transformation defined as follows.

- γ is a transformation such that:

$$c^* = \gamma^*(a) \ :$$

$$c^*_j = f_j(\{a_i\}_{i=0}^{16}) \ \ , \ \ j = 0, 1, ..., 8 \ ,$$

$$c^*_j = f_j(\{a_i\}_{i=9}^{16}) \ \ , \ \ j = 9, 10, ..., 16 \ ,$$

where the functions f_j, $j = 0, 1, ..., 16$, are arbitrary.

In a particular case, γ^* can be only a slightly modified original γ transformation such that:

$$c^*_j = a_j \oplus (a_{j+1} \text{ OR } \bar{a}_{j+2}) \ \ , \ \ j = 0, 1, ..., 8,$$

$$c^*_{9+j} = a_{9+j} \oplus (a_{9+(j+1)mod8} \text{ OR } \bar{a}_{9+(j+2)mod8}) \ \ , \ \ j = 0, 1, ..., 7.$$

Recall that the original γ transformation is specified as follows:

$$c = \gamma(a) \ :$$

$$c_j = a_j \oplus (a_{j+1} \text{ OR } \bar{a}_{j+2}) \ \ , \ \ j = 0, 1, ..., 8,$$

$$c_{9+j} = a_{9+j} \oplus (a_{(9+j+1)mod17} \text{ OR } \bar{a}_{(9+j+2)mod17}) \ \ , \ \ j = 0, 1, ..., 7.$$

- θ^* is a transformation such that the following is valid:

$$c^* = \theta^*(a) \ \ :$$

$$c^*_j = \phi_j(\{a_i\}_{i=0}^{16}) \ \ , \ \ j = 0, 1, 3, 5, 6, 8, 10, 13, 15 \ ,$$

$$c^*_j = \phi_j(a_2, a_4, a_7, a_9, a_{11}, a_{12}, a_{14}, a_{16}) \ \ , \ \ j = 2, 4, 7, 9, 11, 12, 14, 16 \ ,$$

where the functions ϕ_j, $j = 0, 1, ..., 16$, are arbitrary.

Recall that the original θ transformation is specified as follows:

$$c = \theta(a) \ : \ \ c_j = a_j \oplus a_{j+1} \oplus a_{j+4} \ , \ \ j = 0, 1, ..., 16 \ ,$$

with the indices taken modulo 17.

Accordingly, PANAMA-SM keystream generator preserves all underlying characteristic of original PANAMA, and employs on;y a slightly different updating function of the state.

7.2 Algorithm for Cryptanalysis of PANAMA-SM

Note that, although more complex than PANAMA-S2, because the state updating function of PANAMA-SM consists of composition of four transformations identical or of same nature as in original PANAMA, PANAMA-SM is in the following manner equivalent to PANAMA-S2:
- when the state elements 9-16 are known, after employment of the transformation γ^*, all the state elements on positions 9-16 are still known;
- the state elements on the positions 2,4,7,9,11,12,14,16, depend only on the corresponding buffer elements from current iteration and the keystream output of the previous iteration.

Accordingly, following the underlying ideas of Algorithm II, and as its appropriate modification, the following Algorithm III is developed for cryptanalysis of PANAMA-SM. Note that Algorithm III has the same structure as Algorithm II, and so, it can be directly shown that it has time complexity proportional to 2^{65}, and requires the keystram sequence of length $O(100)$ for the cryptanalysis.

Algorithm III

- *INPUT*:
  - PANAMA-SM outputs $[a_j(t)]_{j=9}^{16}$, $t = 1, 2, ..., n$;

- *TWO PHASES PROCESSING BASED ON HYPOTHESIS TESTING*:

  1. *PHASE I*

     For each bit level $k = 0, 1, ..., 31$, do the following.

     (a) *Initial Recalculation*
     - For $j = 12, 14, 16$, calculate $b_{j-9}^{16}(t)$, $t = 1, 2, ..., n-1$,

     $$b_{j-9}^{16}(t) = \phi_j(\{rot_{(\ell(\ell+1)/2) mod 32} \ f_{(7\ell) mod 17}(\{a_i(t)\}_{i=9}^{16})\}_{\ell=2,4,7,9,11,12,14,16}) \oplus a_j(t+1) \ .$$

- For $j = 2, 4$, calculate $a_j^*(t)$, $t = 1, 2, ..., n-1$,

$$a_j^*(t) = \phi_j(\{rot_{(\ell(\ell+1)/2) mod 32}\ f_{(7\ell) mod 17}(\{a_i(t)\}_{i=9}^{16})\}_{\ell=2,4,7,9,11,12,14,16}) \ .$$

(b) *Setting a Working Hypothesis*

i. For $j = 3, 5, 7$, set: $\hat{b}_{j,k}^{24-i}(25) = b_{j,k}^{16}(i+1)$, $i = 0, 1, ..., 24$.

ii. Set a previously unconsidered hypothesis on:
- $\hat{b}_{1,k}^i(25)$, $i = 0, 1, ..., 24$,
- $\hat{b}_{j,k}^i(25)$, $i = 25, 26, ..., 31$, $j = 1, 3, 5, 7$.

(c) *Recalculation*

For $t = 25, 26, ..., n-1$, do the following:

i. for $j = 1, 3$, update the buffer bit-layer (j, k),
$\hat{b}_{j,k}^i(t+1) = \hat{b}_{j,k}^{i-1}(t)$ if $i \neq 0, 25$,
$\hat{b}_{j,k}^0(t+1) = \hat{b}_{j,k}^{31}(t) \oplus a_{(j+1),k}^*(t)$,
$\hat{b}_{j,k}^{25}(t+1) = \hat{b}_{j,k}^{24}(t) \oplus \hat{b}_{((j+2) mod 8),k}^{31}(t)$.

ii. for $j = 5, 7$, update the buffer bit-layer (j, k),
$\hat{b}_{j,k}^i(t+1) = \hat{b}_{j,k}^{i-1}(t)$ if $i \neq 0, 25$,
$\hat{b}_{j,k}^0(t+1) = b_{j,k}^{16}(t+1)$,
$\hat{b}_{j,k}^{25}(t+1) = \hat{b}_{j,k}^{24}(t) \oplus \hat{b}_{((j+2) mod 8),k}^{31}(t)$.

(d) *Hypothesis Check*

i. if $\hat{b}_{3,k}^0(t) = b_{3,k}^{16}(t)$, $t = 26, 27, ..., n$, do the following:

A. for $j = 6, 8$, calculate
$a_{j,k}(t) = \hat{b}_{(j-1),k}^0(t+1) \oplus \hat{b}_{(j-1),k}^{31}(t)$, $t = 25, 26, ..., n-1$

B. memorize
$\hat{b}_{j,k}^i(t)$, $i = 0, 1, ..., 31$, $j = 1, 3, 5, 7$, $t = 51$;
$a_{j,k}(t)$, $j = 6, 8$, $t = 51$.

ii. if $\hat{b}_{3,k}^0(t) \neq b_{3,k}^{16}(t)$, for some $t = 26, 27, ..., n$, check the next hypothesis.

2. *PHASE II*

For each bit level $k = 0, 1, ..., 31$, do the following.

(a) *Initial Recalculation*

Using the input data and the memorized data from Phase I, do the following.

- For $j = 9, 11, 13$, calculate $b_{j-9}^{16}(t)$, $t = 26, 27, ..., n-1$,

$$b_{j-9}^{16}(t) = \phi_j(\{rot_{(\ell(\ell+1)/2) mod 32}\ f_{(7\ell) mod 17}(\{a_i(t)\}_{i=9}^{16})\}_{\ell=2,4,7,9,11,12,14,16}) \oplus a_j(t+1) \ .$$

- For $j = 3, 7$, calculate $a^*_j(t)$, $t = 26, 27, ..., n-1$,

$$a^*_j(t) = \phi_j(\{rot_{(\ell(\ell+1)/2) mod 32}\ f_{(7\ell) mod 17}(\{a_i(t)\}_{i=9}^{16})\}_{\ell=2,4,7,9,11,12,14,16})\ .$$

(b) *Setting a Working Hypothesis*

i. For $j = 0, 2, 4$, set: $\hat{b}^{24-i}_{j,k}(50) = b^{16}_{j,k}(i + 26)$, $i = 0, 1, ..., 24$.

ii. Set a previously unconsidered hypothesis on:
- $\hat{b}^i_{6,k}(50)$, $i = 0, 1, ..., 24$,
- $\hat{b}^i_{j,k}(50)$, $i = 25, 26, ..., 31$, $j = 0, 2, 4, 6$.

(c) *Recalculation*
For $t = 50, 51, ..., n-1$, do the following:

i. for $j = 3, 7$, update the buffer bit-layer (j, k),
$\hat{b}^i_{j,k}(t+1) = \hat{b}^{i-1}_{j,k}(t)$ if $i \neq 0, 25$,
$\hat{b}^0_{j,k}(t+1) = \hat{b}^{31}_{j,k}(t) \oplus a^*_{(j+1),k}(t)$,
$\hat{b}^{25}_{j,k}(t+1) = \hat{b}^{24}_{j,k}(t) \oplus \hat{b}^{31}_{((j+2) mod 8),k}(t)$.

ii. for $j = 0, 5$, update the buffer bit-layer (j, k),
$\hat{b}^i_{j,k}(t+1) = \hat{b}^{i-1}_{j,k}(t)$ if $i \neq 0, 25$,
$\hat{b}^0_{j,k}(t+1) = b^{16}_{j,k}(t+1)$,
$\hat{b}^{25}_{j,k}(t+1) = \hat{b}^{24}_{j,k}(t) \oplus \hat{b}^{31}_{((j+2) mod 8),k}(t)$.

(d) *Hypothesis Check*

i. if $\hat{b}^0_{2,k}(t) = b^{16}_{2,k}(t)$, $t = 51, 52, ..., n$, do the following:

A. for $j = 0, 5$, calculate
$a_{j,k}(t) = \hat{b}^0_{(j-1),k}(t+1) \oplus \hat{b}^{31}_{(j-1),k}(t)$, $t = 51$

B. memorize
$\hat{b}^i_{j,k}(t)$, $i = 0, 1, ..., 31$, $j = 1, 3, 5, 7$, $t = 51$;
$a_{j,k}(t)$, $j = 6, 8$, $t = 51$.

ii. if $\hat{b}^0_{3,k}(t) \neq b^{16}_{3,k}(t+16)$, for some $t = 51, 52, ..., n$, check the next hypothesis.

- *OUTPUT*:
Recovered complete contents of the state and buffer based on the data memorized in the step (d).i.B of Phase I and Phase II.

8 Discussion of PANAMA Security and a Proposal for the Modification

Having in mind all the reported results on security evaluation of PANAMA keystream generator, this report focusses on the security evaluation of PANAMA via security evaluation of its certain simplified and similar versions.

Sections 5 and 6 present techniques, Algorithm I and Algorithm II, developed for cryptanalysis of two simplified PANAMA keystream generators, PANAMA-S1 and PANAMA-S2, respectively. Section 7 shows that Algorithm II developed for cryptanalysis of PANAMA-S2 can be adapted to Algorithm III for cryptanalysis of a more complex PANAMA-SM which is very close to original PANAMA keystream generator.

These results yields identification of:
- certain boundaries of simplification of the PANAMA like keystream generators which implies breakability of the generators;
- the critical components of PANAMA on which the security rests, and this is a more notable implication.

Particularly, note the following:
• A very undesirable feature is that a more simplified PANAMA, PANAMA-S1 requires more complex cryptanalysis than PANAMA-S2/PANAMA-SM.
• The opportunity for cryptanalysis of PANAMA-S2/PANAMA-SM is based on the fact that buffer employes word rotation operation and that the rotation parameter allows identification of certain substructures. This undesirable characteristic opens a door for employing a divide-and-conquer approach to the cryptanalysis.

Particularly, cryptanalysis of PANAMA-S2 shows that complete security of PANAMA keystream generator rest on the following two transformations of the state updating function: γ and θ. Without these two transformations PANAMA keystream generator is breakable with complexity proportional only to 2^{65} assuming that a short output from the keystream generator of length $O(100)$ is available.

Note that PANAMA-SM has the same overall structure consisting of the buffer, the state, theirs updating functions and the output function, as original PANAMA with:
- identical the buffer and state;
- identical the buffer updating function;
- identical output function;
- the state updating function with two identical components, π and σ, and two other γ and θ, modified but of same level of complexity as the original ones, noting that employed transformation γ^* and θ^* are not unique but can belong to a family of transformations.

Breakability of PANAMA-SM by Algorithm III with time complexity proportional to 2^{65} based on a sample of $O(100)$ indicates the following:
- certain critical elements for security of PANAMA;
- nature of certain weaknesses in PANAMA algorithm.

Heuristically speaking, the employed modified transformations γ^* and θ^* seem to be "not too far" from the original ones, , which implies a suspicion that some future research can point out a way for cryptanalysis of original PANAMA keystream generator.

Although PANAMA keystream generator is resistant against the developed algorithms for cryptanalysis of PANAMA-S1 and PANAMA-S2/PANAMA-SM, it appears that particularly Algorithm III implies a strong suggestion for certain modifications of PANAMA.

The intention of the suggestion for a modification, proposed here, is to point out only an approach for improvement, but do not suggest any particular modification because it should be also based on consideration of the implementation suitability.

Recommendable modifications of PANAMA are as follows.

- Modify the employed rotation operation in the buffer.
- Either the word or bit rotation operation is employed in the buffer the rotation parameter should be selected so that it involves all the layers. Accordingly if the word rotation is employed the rotation parameter should be co-prime to 8, and if the bit rotation is employed it should be co-prime to 256.
- Additionally, employment of the rotation parameter in the buffer which vary in time can increase the security margin.

9 Conclusions

Recognizing previous results regarding the security evaluation of PANAMA keystream generator and MULTI-S01 reported in [2], [3], [4], this report focuses on novel approaches relevant for security evaluation of PANAMA based on results of cryptanalysis of certain PANAMA-like keystream generators. Previously reported results show that PANAMA keystream generator and MULTI-S01 pass the following security checks:
(a) consideration of design philosophy,
(b) resistance against known cryptanalytic attacks,
(c) statistical testing checks.
Also, [3] contains consideration of security of certain simplified PANAMA algorithms, but based on the all reported results, security evaluation of simplified/modified PANAMA structures is the most open issue. Accordingly this report addresses this most open issue because the issues (a)-(c) have already been covered in an appropriate way from the present knowledge point of view.

The reasons for consideration of the security strength of a modified keystream generator include the following. It is very important to consider specialized attacks dedicated to a particular keystream generator because there are a number of examples where certain keystream generator is fully resistant against all general attacks but is breakable by a especially developed attack. The previous also implies that it is very important to consider security of a modified keystream generator because these dedicated considerations can tell us a lot about the structural characteristics and critical points for security of the original keystream generator. Also note that strength consideration of a modified cryptographic primitive could open a door for certain dedicated (specialized) attacks.

This report points out novel insights of PANAMA keystream generator yielding relevant elements for its security evaluation. Security evaluation of PANAMA keystream generator is considered starting from an appropriate simplified version of PANAMA, and by gradually increasing complexity of the starting initial version towards original PANAMA.

Main results of this report are the following:
• an equivalent representation of PANAMA keystream generator which opens a door for more detail security evaluation;
• development of two algorithms for cryptanalysis of PANAMA like keystream generators;
• identification of certain critical points for PANAMA security;
• proposal for modifications of PANAMA.

The developed equivalent scheme of PANAMA keystream generator 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.

This report points 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. Both the considered simplified schemes, PANAMA-S1 and PANAMA-S2, preserve the underlying PANAMA concept including employment of a large internal memory devided 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 PANAMA-S1 and PANAMA-S2, respectively. 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 cryptanalyzed. 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.

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.

The proposal for modification is related to modification of PANAMA buffer updating function in order to make it resistant against identified devide-and-conquer attacks.

Although PANAMA keystream generator is resistant against the developed algorithms for cryptanalysis of PANAMA-S1 and PANAMA-S2/PANAMA-SM, it appears that particularly Algorithm III implies a strong suggestion for certain modifications of PANAMA keystream generator in a manner suggested in the previous section.

Finally, note that this report has been produced within a very limited time of only one month and a half but shows a number of previously unknown features of PANAMA, and this is also a reason more why further research towards PANAMA keystream generator security is recommendable.

10 Appendix A: Background for Security Evaluation

Cryptographic techniques play an important role in information protection, and stream ciphers are one of the main cryptographic techniques of very high importance for developing the security mechanisms for information technologies.

One of the central issues in contemporary cryptology is related toward a high-level methodology for evaluation of the basic cryptographic techniques. A main goal of the evaluation of a cryptographic technique is to yield an estimate of its cryptographic strength, as well as to provide elements to compare in a fair and acceptable way different cryptographic techniques.

With a very rare exemption, it is not possible to exactly prove the cryptographic security of a stream cipher. Accordingly, it is very important to employ an appropriate methodology for the security evaluation which will provide as much as possible information on security of a stream cipher.

In this appendix we summarize and discuss main methods for security evaluation of stream ciphers following the frameworks of current projects [9],[10], and [6],[8]. Both of these projects are devoted to evaluation of a large number of proposals for different basic cryptographic techniques (cryptographic primitives) and selection the best ones which will be recommended for use in different applications including e-government and e-commerce. Also, the statements are based on other relevant references (including a number of published results of authors of this paper) and they are illustrated by certain examples.

10.1 Stream Ciphers

The stream cipher encryption is based on employment of a sequence called keystram.

Let the information be a sequence of elements from an alphabet: we call this sequence a plaintext. A stream cipher is an encryption algorithm which transforms a sequence of elements from a plaintext alphabet into another sequence called the ciphertext. An inverse operation is performed for deciphering, i.e. obtaining the plaintext based on its encrypted form - ciphertext.

A stream cipher encrypts one individual character of a plaintext message at a time, using an encryption transformation which varies with time. Such a cipher is typically implemented by the use of a so-called pseudorandom number generator or a keystream generator which expands a short secret key into a long running key sequence. A keystream generator is equivalent to a finite state machine that, based on a secret key, generates a keystream for controlling an encryption transformation. The initial state of a keystream generator is determined by a secret key. We can essentially regard a stream cipher as a keystream generator under the control of a short secret key.

Most stream ciphers are based on simple devices that are easy to implement and run efficiently. Examples of such devices include linear feedback shift registers (LFSRs). Such simple devices produce outputs which have certain desirable properties but these outputs also have a drawback that an output is predictable given some previous output. Thus, the output of such devices is typically used as the inputs to certain (nonlinear) function that produces the keystream (see [11], for example).

Finally, note that generally speaking, there are two main approaches for construction of a keystream generator:
(A): a construction which is resistant against all known attacks;
(B): a construction which yields security under some assumptions.

Almost all of the reported constructions follow the approach (A). According to the approach (A) we are confident that particular keystream generator is insecure if a successful attack on it can be found, but a main drawback of the approach (A) is that if we do not know any successful attack, we still do not have any proof of the security.

The approach (B) implies to prove the absence of attacks under some assumptions. As an example note that a particular construction which follows the approach (B) is submitted to the project [9]. If somehow an attack can be found it implies only that the assumption was false. Elements of provable-security approach include: notations of security, underlying assumptions and employed reduction technique which move the problem of a keystream security to a problem of the assumption security. Previous also illustrates a main drawback of the approach (B): it is a serious problem to develop a high efficient and provable-secure keystream generator. An important issue to build a provable-secure keystream generator is to employ an appropriate atomic primitive (hard problem) which yields high implementation efficiency, as well. Finally, note that the provable-security will not help if there are real attacks not covered by the model, for example.

10.2 General Evaluation Criteria and Methodological Issues

Usually, the following criteria are employed for security evaluation of a cryptographic technique including the stream ciphers.

- An attack should be at least as difficult as the generic attack against a stream cipher, i.e. exhaustive search over all possible secret keys.

- Existence of an attack requiring lower computation resources than claimed by the stream cipher designers would usually disqualify the stream cipher as a widely recommendable one.

- Stream cipher should be evaluated within the stated environment. Thus, consideration of vulnerability to side channel attacks (e.g. timing attacks, power analysis, ...) may be appropriate.

Accordingly, 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. The attacks related to recovering the secret key are usually classified as follows: (i) *ciphertext-only attack* - the attacker has obtained a set of intercepted ciphertexts and has some general information about nature of the plaintext; (ii) *known plaintext attack* - the attacker obtains a set of plaintexts and the corresponding ciphertexts; (iii) *chosen plaintext attack* - the attacker choses a set of plaintexts and obtains in some way the corresponding ciphertexts; (iv) *chosen ciphertext attack* - the attacker choses a set of ciphertexts and obtains in some way the corresponding plaintexts.

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.

Cryptographic Environment. In certain cryptographic environments, a cryptographic technique may have been designed to posses intrinsic security advantages or disadvantages. Such properties should be considered when assessing security of a stream cipher, as well.

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

10.3 Techniques for Security Evaluation

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 (see [12], for example).

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 approximations to the output. 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 including specific issues related to a particular stream cipher.

10.3.1 General Characteristics to be Evaluated

Period. Clearly, the period of keystram sequence employed in a stream cipher must be large enough that the keystream has virtually no chance of being repeated.

Keystream Complexity: Linear and Nonlinear. The linear complexity of a sequence is the length of the shortest LFSR that can produce the sequence. The linear complexity of a sequence is easily calculated using the Berlekamp-Massey algorithm (see [11], for example), noting that it can be theoretically estimated in certain cases, as well (see [13], for example). If this linear complexity is too small, then an attacker can reproduce the sequence on a simple device and recover the secret key. Other complexity mesures (nonlinear complexity, for example) should have appropriate values, as well.

Statistical Properties. Ideally, the keystram generator should produce a memoryless sequence of bits. Accordingly, the binary equivalent of a keystream sequence should be a realization of independent identically distributed random variables with the parameter equal to 0.5. If the keystream deviates from these distributions, than an attacker may be able to use this deviation to predict future keystream bits. The same characteristics can be requested for certain internal keystream generator sequences as well. The sets of statistical tests which a good stream cipher should passed are specified in [14](pp.35-36) and [9], for example. The statistical test for a stream cipher employed by [9] are the following ones:

Dyadic Complexity Test	Percolation Test
Constant Runs Test	Frequency Test
Collision Test	Overlapping m-tuple Test
Gap Test	Coupon Collector's Test
Universal Maurer Test	Poker Test
Spectral Test	Correlation Test
Rank Test	Linear Complexity Test
Nonlinear Complexity Test	Ziv-Lempel Complexity Test

10.3.2 General Attacks to be Evaluated

Time-Memory Trade-Offs. In such attacks, the time required to find the secret key by an exhaustive search is reduced at the expense of increasing the memory required to execute the attack (see [15] and [16]).

Divide-and-Conquer Attacks. In such attacks, a portion of the secret key is guessed. The constraints now placed on the keystream may allow the determination of remainder of the key faster than searching this remainder.

Correlation Attacks. In a correlation attack (see [17]), the output of a keystream generator is correlated in some manner with the output of much simpler device, such as component LFSR of the generator. This correlation can sometimes be exploited to determine the key. A very important class of the correlation attacks is *fast correlation attack* (see [18] for basic fast correlation attack and [22]-[32] for the advanced ones, for example). Also, other attacks related, for example, to the cases when non-uniform decimation or time-varying combination function are employed should be carefully considered (see [33] and [35], for example).

Distinguishing Attacks. In such attacks, a method is given for distinguishing the keystream from a genuinely random sequence of the type described above.

Rekeying Attacks. There are many uses for which a stream cipher is rekeyed. It is sometimes possible to exploit this rekeying in order to find the keys used.

10.4 Some Remarks

Security evaluation of a stream cipher is usually a very complex issue. It requires a serious and creative consideration of a number of general and specific issues. It is very important to note that the security evaluation result strongly depends on the skills of the evaluation experts: Following the same evaluation guidelines an expert could be able to find a weakness in the stream cipher and the other one could not.

11 Appendix B: PANAMA Overview Details

11.1 Initialization and Keystream Bits Generation

PANAMA keystream generator is initialized by first loading the 256-bit key K, the 256-bit diversification parameter Q and performing 32 additional blank Pull iterations. During keystream generation an 8-word block z is delivered at the output for every iteration. In practice, the diversification parameter allows frequent resynchronization without the need to change the key.

The sequence diagram of PANAMA keystream operations is given by Table 1.

Table 1: The sequence diagram of PANAMA keystream generator operations

time step t	mode	input	output
-34	reset	-	-
-33	Push	K	-
-32	Push	Q	-
-31,...,0	Pull	-	-
1,...	Pull	-	z_t

11.2 Design Rationale

The updating transformation of the state has high diffusion and distributed nonlinearity. Its design is aimed at providing very high nonlinearity and fast diffusion for multiple iterations. This is realized by the combination of four distinct transformations each with its specific contribution. There is one for nonlinearity, one for bit dispersion, one for inter-bit diffusion, and one for injection of the buffer and input bits.

The buffer behaves as a linear feedback shift register and ensures that input bits are injected into the state over a wide interval of iterations. In the push mode, the input to the shift register is formed by the external input, and in the Pull mode, by part of the state.

The design goal for PANAMA keystream generator is that it should be hermetic and K-secure. K-security implies among other things that, given part of the keystream outputs corresponding with a given key and for chosen values Q, the most efficient way to gain knowledge on the key or on the complementary part of the keystream output is exhaustive key search.

For every Pull iteration 16 words of buffer are injected into the state and 8 state words are given at the output. In the short term, the number of buffer bits that are injected into the state is twice as large as the number of bits given at the output. The feedback from the state to the buffer causes the buffer contents to be renewed every 32 iterations. These factors cause the correlation between output bits and linear combinations of the state and buffer bits to be too small for practical use in cryptanalysis.

Resynchronization attacks should be made infeasible by 32 blank Pull iterations after loading the initialization blocks. Because of the feedback from the state to the buffer in the Pull mode, the state and almost all buffer stages depend in a complicated way on these blocks at the end of the initialization phase.

The security characteristics of the state updating transformation include the following.

γ is a nonlinear transformation with the following propagation properties: The maximum correlation between the input and output, and the difference propagation probability diminish exponentially.

The transformation θ corresponds to the multiplication by a binary polynomial modulo $1 \oplus x^{17}$. It was selected from the invertible polynomials with Hamming weight 3 on the basis of its good diffusion properties. A single input difference gives rise to three output differences.

The cyclic shift coefficients of π, described by the simple expression $(i(i+1)/2) mod 32$ form an array of 17 different constants. The ward permutation factor 7 is chosen to let every component of ρ depends on 9 state bits. For the chosen π parameters it has been verified that $\rho \circ \rho$ has propagation and correlation properties that are close to optimal with respect to the space of possible π parameters. On the average, a difference in a single bit diffuses to 6 bits after one iteration, 36 bits after two, 216 after three and over all the state after 4 iterations.

σ includes the addition of a constant to q_0 to prevent symmetric properties. The constant value 00000001_{hex} was chosen for its simplicity.

Since γ, π, θ and σ are all invertible, the state updating transformation ρ is also invertible.

References

[1] *A Symmetric Key Encryption Algorithm: MULTI-S01 (An Integrity-Aware Block Encryption Based on Cryptographic Pseudorandom Number Generator)*, HITACHI, LTD., 18 pages, 2000.

[2] 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.

[3] *Self-Evaluation Report MULTI-S01 (revised for 2001 submission)*, HITACHI, LTD, 24 pages, 2001.

[4] *Evaluation of MULTI-S01 Algorithm*, CRYPTREC Internal Report, 38 pages, 2000.

[5] V. Rijmen, B.V. Rompay, B. Preneel and J. Vandewalle, "Producing collisions for PANAMA", Fast Software Encryption - FSE2001, Yokohama, Japan, April 2001, *Preproceedings*, pp. 39-53, 2001 (to appear in Lecture Notes in Computer Science).

[6] H. Imai and A. Yamagishi, "CRYPTREC Project - Cryptographic Evaluation Project for Japanese Electronic Government", invited talk at ASIACRYPT2000, Kyoto, Japan, Dec. 3-7, 2000, *Lecture Notes in Computer Science*, vol. 1976, pp. 399-400, 2000.

[7] IPA-ISEC: Call for attack to evaluate Cryptographic Techniques.
www.ipa.go.jp/security/enc/CRYPTREC/index-e.html, 2000.

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

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

[10] "Security Evaluation of NESSIE First Phase", *NESSIE Report*, Sept. 2001.
http://www.cryptonessie.org

[11] A. Menezes, P.C. van Oorschot and S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, Boca Roton, 1997.

[12] M. J. Mihaljević and H. Imai, "Cryptanalysis of TOYOCRYPT-HS1 stream cipher", to appear in *IEICE Transactions on Fundamentals*, vol. E85-A, Jan. 2002.
(preliminary presented as: M. J. Mihaljević and H. Imai, "Effective secret key size of TOYOCRYPT-HS1 stream cipher", *The 2001 Symposium on Cryptography and Information Security - SCIS2001*, Oiso, Japan, January 23-26, 2001, Proceedings, 665-667.)

[13] J. Golić and M. J. Mihaljević, "Minimal linear equivalent analysis of a variable memory binary sequence generator", *IEEE Transactions on Information Theory*, vol. 36, pp. 190-192, Jan. 1990.

[14] NIST Federal Information Processing Standard Publication 140-2 "Security requirements for Cryptographic Modules". US National Institute for Standards and Technology, US Department of Commerce, June 2001.
http://csrc.nist.gov/publications/fips.

[15] M. E. Hellman, "A cryptanalytic time-memory trade-off," *IEEE Trans. Inform. Theory*, vol. IT-26, no. 4, pp. 401-406, July 1980.

[16] A. Biryukov and A. Shamir, "Cryptanalytic Time/ Memory/ Data Tradeoffs for Stream Ciphers", Advances in Cryptology - ASIACRYPT2000, *Lecture Notes in Computer Science*, vol. 1976, pp. 1-13, 2000.

[17] T. Siegenthaler, "Decrypting a class of stream ciphers using ciphertext only," *IEEE Trans. Comput.*, vol. C-34, pp. 81-85, 1985.

[18] W. Meier and O. Staffelbach, "Fast correlation attacks on certain stream ciphers," *Journal of Cryptology*, vol. 1, pp. 159-176, 1989.

[19] M. J. Mihaljević and J. Dj. Golić, "A fast iterative algorithm for a shift register initial state reconstruction given the noisy output sequence," Advances in Cryptology - AUSCRYPT '90, *Lecture Notes in Computer Science*, vol. 453, pp. 165-175, 1990.

[20] M. J. Mihaljević and J. Dj. Golić, "A comparison of cryptanalytic principles based on iterative error-correction," Advances in Cryptology - EUROCRYPT '91, *Lecture Notes in Computer Science*, vol. 547, pp. 527-531, 1991.

[21] M. J. Mihaljević and J. Golić, "Convergence of a Bayesian iterative error-correction procedure on a noisy shift register sequence", Advances in Cryptology - EUROCRYPT'92, *Lecture Notes in Computer Science*, vol. 658, pp. 124-137, 1993.

[22] T. Johansson and F. Jonsson, "Improved fast correlation attacks on stream ciphers via convolutional codes," Advances in Cryptology - EUROCRYPT'99, *Lecture Notes in Computer Science*, vol. 1592, pp. 347-362, 1999.

[23] T. Johansson and F. Jonsson, "Fast correlation attacks based on turbo code techniques," Advances in Cryptology - CRYPTO'99, *Lecture Notes in Computer Science*, vol. 1666, pp. 181-197, 1999.

[24] T. Johansson and F. Jonsson, "Fast correlation attacks through reconstruction of linear polynomials," Advances in Cryptology - CRYPTO2000, *Lecture Notes in Computer Science*, vol. 1880, pp. 300-315, 2000.

[25] A. Canteaut and M. Trabbia, "Improved fast correlation attacks using parity-check equations of weight 4 and 5," Advances in Cryptology - EUROCRYPT'2000, *Lecture Notes in Computer Science*, vol. 1807, pp. 573-588, 2000.

[26] V. V. Chepyzhov, T. Johansson and B. Smeets, “A simple algorithm for fast correlation attacks on stream ciphers,” Fast Software Encryption - FSE2000, *Lecture Notes in Computer Science*, vol. 1978, pp. 180-195, 2001.

[27] M. J. Mihaljević, M. P. C. Fossorier and H. Imai, “A low-complexity and high-performance algorithm for the fast correlation attack,” Fast Software Encryption - FSE2000, *Lecture Notes in Computer Science*, vol. 1978, pp. 196-212, 2001.

[28] M. J. Mihaljević, M. P. C. Fossorier and H. Imai, “An algorithm for cryptanalysis of certain keystream generators suitable for high-speed software and hardware implementations,” *IEICE Trans. Fundamentals*, vol. E84-A, pp. 311-318, Jan. 2001.

[29] M. J. Mihaljević and J. Dj. Golić, “A method for convergence analysis of iterative probabilistic decoding,” *IEEE Trans. Inform. Theory*, vol. 46, pp. 2206-2211, Sept. 2000.

[30] M. P. C. Fossorier, M. J. Mihaljević and H. Imai, “Reduced complexity iterative decoding of Low Density Parity Check codes based on Belief Propagation,” *IEEE Trans. Comm.*, vol. 47, pp. 673-680, May 1999.

[31] M. J. Mihaljević, M. P. C. Fossorier and H. Imai, ”On decoding techniques for cryptanalysis of certain encryption algorithms”, *IEICE Trans. Fundamentals*, vol. E84-A, pp. 919-930, April 2001.

[32] M. J. Mihaljević, M.P.C. Fossorier and H. Imai, ”Fast correlation attack algorithm with the list decoding and an application”, *Fast Software Encryption Workshop - FSE2001*, Yokohama, Japan, April 2001, Pre-proceedings, pp. 208-222 (also to appear in *Lecture Notes in Computer Science*).

[33] J. Golić and M. J. Mihaljević, ”A generalized correlation attack on a class of stream ciphers based on the Levenshtein distance”, *Journal of Cryptology*, vol. 3, pp. 201-212, 1991.

[34] M. J. Mihaljević, ”A faster cryptanalysis of the self-shrinking generator”, *Lecture Notes in Computer Science*, vol. 1172, pp. 182-188, 1996.

[35] M. J. Mihaljević, ”A correlation attack on the binary sequence generators with time-varying output function”, ASIACRYPT’94, *Lecture Notes in Computer Science*, vol. 917, pp.67-79, 1995.