On IV Setup of Pomaranch

Mahdi M. Hasanzadeh[†] Shahram Khazaei[†] Alexander Kholosha[‡]

[†] Zaeim Electronic Industries Company, P.O. BOX 14155-1434, Tehran, Iran
[‡] The Selmer Center, University of Bergen, P.O. Box 7800, 5020, Bergen, Norway
{Hasanzadeh, khazaei}@Zaeim.com, Alexander.Kholosha@ii.uib.no

Abstract. Pomaranch is a synchronous bit-oriented stream cipher submitted to eSTREAM, the ECRYPT Stream Cipher Project. Following the recently published chosen IV [1] and correlation [7] key-recovery attacks, the authors changed the configuration of jump registers and introduced two new key-IV setup procedures for the cipher. We call the updated version as Tweaked Pomaranch vs. Original Pomaranch [4]. In this paper we use the findings of [7] to mount a chosen IV key-recovery attack on the Original Pomaranch with computational complexity of $O(2^{73.5})$. The attack is also applicable to the first key-IV setup proposal for Tweaked Pomaranch with computational complexity of $O(2^{117.7})$. The alternative key-IV setup for Tweaked Pomaranch is immune against our attack. Both versions of Pomaranch deal with 128 bit keys.

Keywords. ECRYPT Stream Cipher Project, Pomaranch, CJCSG, Jump Register, Cryptanalysis, Linear Equivalence Bias, Clock-Controlled LFSR, Security Evaluation.

1 Introduction

Pomaranch (also known as a Cascade Jump Controlled Sequence Generator or CJCSG) [4] is a synchronous bit-oriented stream cipher, one of the ECRYPT Stream Cipher Project [2] candidates. It uses 128-bit keys and in its original design - which we call Original Pomaranch - accommodates an Initial Value (IV) of 64 up to 112 bits long. The algorithm uses a one-clock-pulse cascade construction of so called jump registers [3] being essentially linear finite state machines with a special transition matrix. Moreover, the characteristic polynomial of the transition matrix was made to be primitive and satisfying additional constraints that arise from the need to use the register in a cascade jump control setup. The principal advantage of jump registers over the classical clock-controlled arrangements is their ability to move a Linear Feedback Shift Register (LFSR) to a state that is more than one step ahead but without having to step through all the intermediate states. The transition matrix of the jump registers in Pomaranch has been chosen so to secure the design against side-channel attacks while preserving all the advantages of irregular clocking.

Following the recently published chosen IV [1] and correlation [7] key-recovery attacks, the authors made some tweaks on the cipher. Firstly, they changed the configuration of jump registers and then introduced two different key-IV setup procedures for the cipher - one mixes the IV and key similarly to Original Pomaranch limiting the IV length to 78 bits and the other is totally different from the original version and can accommodate IV's up to 126 bits long [6]. These changes effectively counter the attacks introduced in [7, 1]. We call this updated version as Tweaked Pomaranch.

Paper [7] describes a new inherent property of jump registers that allows constructing their linear equivalences. This property was further investigated in [5]. In this paper we use the same idea to mount a resynchronization attack (IV attack) on Original Pomaranch and the first key-IV setup of Tweaked Pomaranch. The second key-IV setup of Tweaked Pomaranch is immune against our attack. In the rest of the paper we just consider Tweaked Pomaranch with the first key-IV setup and refer to Tweaked Pomaranch for convenience.

Our results show that the key of both Original and Tweaked Pomaranch can be found when a key is used with about 2^{35} chosen IV's. The required computational complexities are $O(2^{73.5})$ and $O(2^{117.7})$ for Original and Tweaked Pomaranch respectively. There are also many tradeoffs between the number of IV's and the required bit-stream from each IV.

2 Outline of Original and Tweaked Pomaranch

The key-stream generator of Pomaranch is depicted in Figure 1. The cipher consists of nine cascaded JR denoted by R_1 to R_9. Each JR is built on 14 memory cells which behave either as simple delay shift cells or feedback cells, depending on the value of JC sequence. At any moment, half of the cells in the registers are shift cells, while the other half is feedback cells. The initial configuration of cells is determined by the transition matrix A, and is used if the JC value is zero. If JC is one, all cells are switched to the opposite mode. This is equivalent to switching the transition matrix to $(A + I)$ [4].

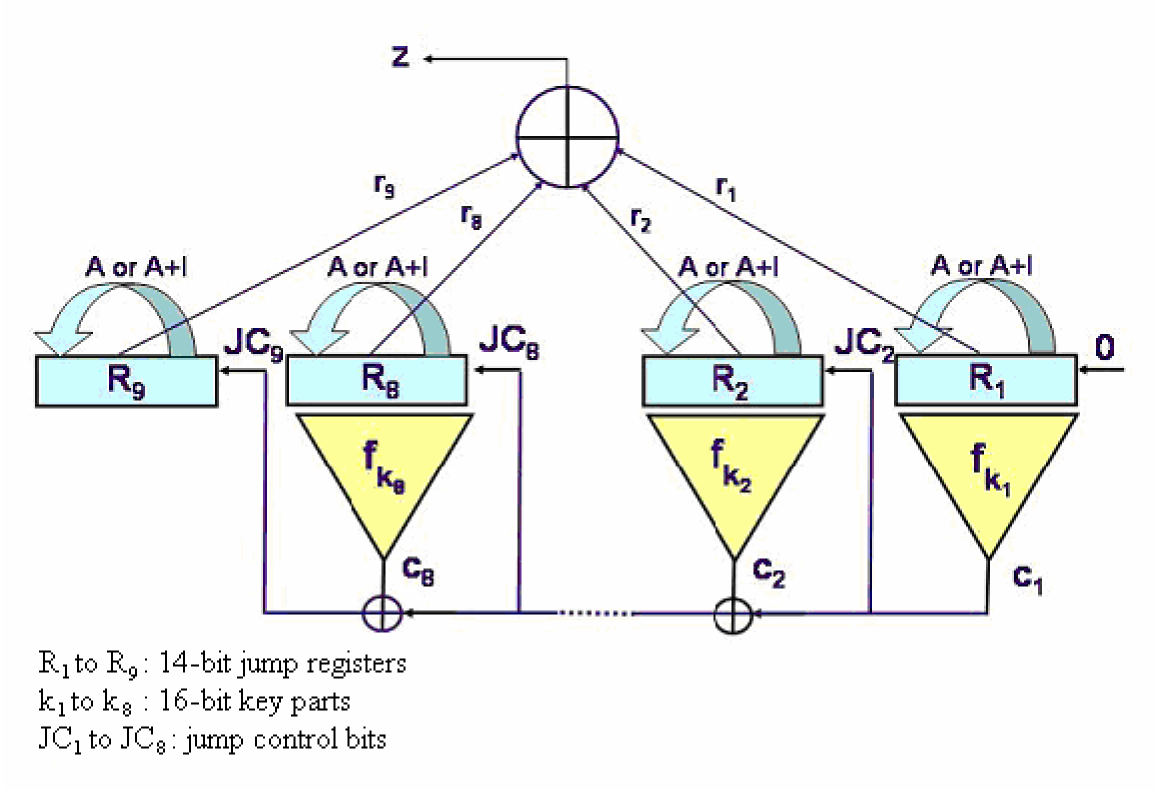
z
r9
r8
r2
r1
A or A+I
A or A+I
A or A+I
A or A+I
JC9
JC8
JC2
0
R9
R8
R2
R1
fk8
fk2
fk1
c8
c2
c1
R1 to R9 : 14-bit jump registers
k1 to k8 : 16-bit key parts
JC1 to JC8 : jump control bits

Figure 1. Schematic of the Pomaranch

The 128-bit key K is divided into eight 16-bit sub keys k_1 to k_8. At time t, the current states of the registers R_1^t to R_8^t are non-linearly filtered, using a function that involves the corresponding sub key k_i. These functions provide as output eight bits c_1^t to c_8^t, which are used to produce the jump control bits JC_1^t to JC_8^t controlling the registers R_2 to R_9 at time t, as following:

$$JC_i^t = c_1^t \oplus \cdots \oplus c_{i-1}^t\,, \qquad 2 \le i \le 9\,. \tag{1}$$

The jump control bit JC_1 of register R_1 is permanently set to zero. The key-stream bit z^t produced at time t is the XOR of nine bits r_1^t to r_9^t selected at second position of the registers R_1 to R_9, that is $z^t = r_1^t \oplus \cdots \oplus r_9^t$.

The only difference between the key-stream generator of Original and Tweaked Pomaranch is the configuration of the jump registers or equivalently the A matrix.

Key-IV Setup of Original Pomaranch [4]: During the cipher initialization, the content of registers R_1 to R_9 is first set to non-zero constant 14-bit values derived from π, then the sub keys k_i are loaded and the registers are run for 128 steps in a special mode (called Shift Mode). The main difference between the Key-Stream Generation Mode and the Shift Mode is that, in the latter the output of the filter function of register R_i (denoted by c_i) is added to the feedback of register R_{i+1}, with the tap from cell 1 in the register R_9 being added to the register R_1, making then what can be seen as a "big loop". Note that the configuration of the jump registers does not change in this mode (they all operate as if $JC_i = 0$). This process ensures that the states of the registers R_1 to R_9 after this key loading phase depend upon the entire key K. We denote these states by $R_1(K)$ to $R_9(K)$.

Next the IV is loaded into the registers. The IV can have any arbitrary length between 64 and 112 bits. First, the IV is expanded by cyclically repeating it until a length of exactly 126 (= 9×14) bits is obtained. This new string is then split into nine 14-bit parts, denoted by IV_1 to IV_9, which are XORed with the 14-bit states of registers $R_1(K)$ to $R_9(K)$ obtained at the end of the key loading. If any of the resulting states consists of 14 null bits, its least significant bit is set to one (this ensures that no state will be made up entirely of null bits). The resulting register states R_1 to R_9 form the nine initial states. The key-stream generation mode showed in Figure 1 is now activated, and the run-up consists of 128 steps in which the produced key-stream bits are discarded.

Key-IV Setup of Tweaked Pomaranch [6]: Following the recently published chosen IV attack [1], the authors introduced two different tweaks in key-IV setup of the cipher. In the first version, the length of IV is limited to 78 (= 6×13) bits; all IV's are expanded by cyclically repeating IV-bits until a length of exactly 117 (= 9×13) bits is obtained. First, the key K is loaded into the registers the same way as in the original version. Then for IV loading, the IV-bits are split into groups of 13 bits denoted by IV_i , $1\le i \le 9$. These 13 bit IV-values are XORed with the 13 most significant bits of the registers R_i, that is $R_i(K)$, $1 \le i \le 9$. Now all registers are checked for the all-zero state and if all-zero the least significant bit of the register is set to one.

The second proposed version for key-IV setup is totally different from the old version and uses IV's up to 126 bits length. Since our attack is just applicable on the first version of the newly proposed key-IV setup, we skip the description of this alternative and refer the reader to [6]. Both versions of key-IV setup effectively counter the chosen IV attack introduced in [1]. Note a slight difference between what the authors of [1] considered in their paper as the IV loading procedure and what is in Original Pomaranch. However, this modification does not affect their attack.

3 Description of the Attack

In [7, 5] it has been shown that there are certain linear relations in the output sequence of a Jump Register Section which hold with a fixed bias. Define the correlation coefficient of a binary random variable x as $\varepsilon = 1 - 2\Pr\{x = 1\}$. In particular, for JR's of Original Pomaranch the correlation coefficient of the linear relation $r^{t} \oplus r^{t+8} \oplus r^{t+14}$ is equal to $\varepsilon = 840/2^{14}$ provided that the JC sequence is purely random [7]. This value was called the Linear Equivalent Bias (LEB) in [5]. In [7] using this bias a correlation based key-recovery attack mounted on Original Pomaranch which has computational complexity of $O(2^{95.4})$ and requires $2^{71.8}$ bits of the key-stream generated using a single key and IV pair. In this section we explain how to improve this attack using different IV's.

3.1 Application to the Original Pomaranch

Suppose that we are given the first T bits of the Pomaranch key-stream generated from an unknown fixed key and $l+1$ known random IV's whose first part corresponding to R_1 (14 bits in Original Pomaranch and 13 bits in Tweaked Pomaranch) are the same. Let denote the IV's by IV^i ($0 \le i \le l$) and the output sequence corresponding to IV^i by $\{z^t(i)\}_{t=0}^{\infty}$.

We also denote the output sequence of the nth register by $\{r_n^t(i)\}_{t=0}^{\infty}$ when IV^i is used, thus $z^t(i) = r_1^t(i) \oplus \cdots \oplus r_9^t(i)$. Let introduce the following sequences:

$$e_n^t(i) = r_n{}^t(i) \oplus r_n{}^{t+8}(i) \oplus r_n{}^{t+14}(i)\ ,\ 0 \le i \le l\ , 2 \le n \le 9 \qquad \textbf{(2)}$$

$$u_n^t(i) = e_{10-n}^t(i) \oplus \cdots \oplus e_9^t(i)\ ,\ 0 \le i \le l\ ,\ 1 \le n \le 8 \qquad \textbf{(3)}$$

$$Z^t(i) = z^t(i) \oplus z^{t+8}(i) \oplus z^{t+14}(i)\ ,\ 0 \le i \le l\ . \qquad \textbf{(4)}$$

Using this notation the following relation holds for every $0 \le i \le l$:

$$Z^t(i) = r_1{}^t(i) \oplus r_1{}^{t+8}(i) \oplus r_1{}^{t+14}(i) \oplus u_8^t(i)\ . \qquad \textbf{(5)}$$

Since the correlation coefficient of the sequence $e_n^t(i)$, $2 \le n \le 9$, is equal to $\varepsilon = 840/2^{14}$, the correlation coefficient of the sequence $u_n^t(i)$, $1 \le n \le 8$, is equal to ε^n under the independence assumption of $e_n^t(i)$ sequences, $2 \le n \le 9$, for every $0 \le i \le l$.

In [7] the equation (5) has been used in a correlation attack to recover the initial sate of R_1 using a single IV (the assumption of using just one IV has been implicitly used). The required key-stream length and computational complexity are $N_0 = 14/C(0.5(1-\varepsilon^8)) \approx 2^{72.8}$ and $2^{14}N_0 \approx 2^{86.8}$ respectively (see [7] for details).

The main contribution of this paper is to increase first the correlation coefficient of $u_8^t(i)$ for a fixed value of *i*, i.e. *i* = 0 and then apply correlation attack. This method will considerably improve the attack. The idea of increasing the correlation coefficient of $u_8^t(0)$ is based on trying to estimate it using the following group of relations

$$Z^t(0) \oplus Z^t(i) = r_1{}^t(0) \oplus r_1{}^{t+8}(0) \oplus r_1{}^{t+14}(0) \oplus r_1{}^t(i) \oplus r_1{}^{t+8}(i) \oplus r_1{}^{t+14}(i) \oplus u_8^t(0) \oplus u_8^t(i)\ . \qquad \textbf{(6)}$$

Since the first part of IV's (IV^i , $0 \le i \le l$) are the same, we have $r_1^t(0) \oplus r_1^t(i) = 0$. Therefore, the relation (6) can be rewritten as

$$\Delta(i) = u_8^t(0) \oplus u_8^t(i)\ ,\ 1 \le i \le l\ , \qquad \textbf{(7)}$$

where $\Delta(i) = Z^t(0) \oplus Z^t(i)$ is completely known.

The ML estimation of $u_8^t(0)$ denoted by $\hat{u}_8^t(0)$ is achieved by comparing $\sum_{i=1}^{l} \Delta(i)$ with the threshold *l*/2. That is, we decide on $\hat{u}_8^t(0) = 0$, if $\sum_{i=1}^{l} \Delta(i) < l/2$ and on $\hat{u}_8^t(0) = 1$ otherwise. The error probability of this estimation is approximately equal to $Q(\sqrt{l}\varepsilon^8)$, where $Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt$. The estimation $\hat{u}_8^t(0)$ of $u_8^t(0)$ and $u_8^t(0)$ can be related by $u_8^t(0) = \hat{u}_8^t(0) \oplus w_8^t(0)$ where $w_8^t(0)$ is the estimation error whose correlation coefficient is equal to $\varepsilon' = 1 - 2Q(\sqrt{l}\varepsilon^8)$.

Using this estimation, the relation (5) for *i* = 0 turns into

$$Z^t(0) = r_1^t(0) \oplus r_1^{t+8}(0) \oplus r_1^{t+14}(0) \oplus \hat{u}_8^t(0) \oplus w_8^t(0)\ . \qquad \textbf{(8)}$$

Now the equation (8) can be used in a correlation attack to recover the initial state of R_1 for IV^0. The required key-stream length and computational complexity are $T = 14/C(0.5(1-\varepsilon'))$ and $2^{14}T$ respectively (see [7] for details).

Since in the first phase we must estimate $u_8^t(0)$ for $0 \le t \le T-1$ using *l* different IV's, the required computational complexity of this phase is *Tl* resulting in a total computational complexity of $C = T(l + 2^{14})$ for initial state recovery of R_1.

For every value of l between 2^{18} and 2^{64}, the minimum amount of the computational complexity is obtained which is equal to $C = 2^{73.5}$. The required key-stream length from each IV is equal to $T = 2^{73.5}/l$, where an attacker can choose the parameter l on his/her fitness.

After finding the initial state of R_1, we can eliminate the portion of $r_1^t(i)$ from the output sequence of Pomaranch for each IV. Define the sequence $z_1^t(i)$ as an XOR of $z^t(i)$ and $r_1^t(i)$ which is now available. Then similarly to (5) we have

$$Z_1^t(i) = r_2^t(i) \oplus r_2^{t+8}(i) \oplus r_2^{t+14}(i) \oplus u_7^t(i)\,, \qquad \textbf{(9)}$$

where

$$Z_1^t(i) = z_1^t(i) \oplus z_1^{t+8}(i) \oplus z_1^{t+14}(i)\,. \qquad \textbf{(10)}$$

The sequence $r_2^t(i) \oplus r_2^{t+8}(i) \oplus r_2^{t+14}(i)$ can be generated if we know both the 14-bit initial state of R_2 and 16-bit sub-key k_1 (totally 30 bits). In [7] the equation (9) has been used in a correlation attack to recover these 30 bits using a single IV. The required key-stream length and computational complexity are $N_0 = 30/C(0.5(1-\varepsilon^7)) \approx 2^{65.4}$ and $2^{30}N_0 \approx 2^{95.4}$ respectively (see [7] for details).

Again we can increase the correlation coefficient of $u_7^t(i)$ for a fixed value of i, i.e. $i = 0$, and then apply correlation attack. The following group of relations

$$Z_1^t(0) \oplus Z_1^t(i) = r_2^t(0) \oplus r_2^{t+8}(0) \oplus r_2^{t+14}(0) \oplus r_2^t(i) \oplus r_2^{t+8}(i) \oplus r_2^{t+14}(i) \oplus u_7^t(0) \oplus u_7^t(i) \qquad \textbf{(11)}$$

can be used to estimate $u_7^t(0)$ similarly to (6). In the first part we assumed that the IV's IV^i are the same in the first part. Here, we must force the IV's IV^i to be the same in the first two parts. Under this condition we have $r_2^t(0) \oplus r_2^t(i) = 0$. Therefore we can compute an estimation of $u_7^t(0)$ denoted by $\hat{u}_7^t(0)$ where $u_7^t(0) = \hat{u}_7^t(0) \oplus w_7^t(0)$ and $w_7^t(0)$ is the estimation error whose correlation coefficient is equal to $\varepsilon'' = 1 - 2Q(\sqrt{l}\varepsilon^7)$. Using this estimation, the relation (9) for $i = 0$ turns into

$$Z_1^t(0) = r_2^t(0) \oplus r_2^{t+8}(0) \oplus r_2^{t+14}(0) \oplus \hat{u}_7^t(0) \oplus w_7^t(0)\,. \qquad \textbf{(12)}$$

Now the equation (12) can be used in a correlation attack to recover the initial state of R_2 for IV^0 and key segment k_1. The required key-stream length and computational complexity are $T = 30/C(0.5(1-\varepsilon''))$ and $2^{30}T$ respectively (see [7] for details). The total computational complexity of initial state recovery of R_2 and key segment k_1 is equal to $C = T(l + 2^{30})$.

For every value of l between 2^{35} and 2^{55}, the minimum amount of the computational complexity is obtained which is equal to $C = 2^{66}$. The required key-stream length from each IV is equal to $T = 2^{66}/l$, where an attacker can choose the parameter l on his/her fitness. Similarly these parameters can be computed for other registers and key parts. These parameters are summarized in Table 1 for the initial state recovery of R_1 to R_5 and key segments k_1 to k_4.

Table 1. Different parameters of finding different sections of Original Pomaranch

Recovered Sections	l	T	Complexity	Number of fixed part of IV's
R_1	$2^{18} \le l \le 2^{64}$	$2^{73.5}/l$	$2^{73.5}$	1
R_2, k_1	$2^{35} \le l \le 2^{55}$	$2^{66}/l$	2^{66}	2
R_3, k_2	$2^{31} \le l \le 2^{51}$	$2^{57.5}/l$	$2^{57.5}$	3
R_4, k_3	$2^{30} \le l \le 2^{35}$	$2^{41}/l$	2^{41}	4
R_5, k_4	$l = 2^{28}$	$2^{5.3}$	$2^{35.8}$	5

After finding key parts k_1 to k_4, the rest part of the key can be found by exhaustive search with computational complexity $O(2^{64})$. Therefore the total computational complexity of our key-recovery attack is

O($2^{73.5}$), and the required number of IV's, the imposed condition on IV's and the required number of key-stream bits from each IV are determined by Table 1 which provides many tradeoffs.

3.2 Application to Tweaked Pomaranch

Following the recently published key-recovery attack [7], the authors changed the configuration of jump registers. The Linear Equivalent Bias (LEB) of new configuration of jump registers is equal to $\varepsilon = 124/2^{14}$ [5] which effectively counters the attack introduced in [7]. For JR's of Tweaked Pomaranch the LEB value is held for the linear relation $r^{t} \oplus r^{t+5} \oplus r^{t+6} \oplus r^{t+10} \oplus r^{t+14}$. Although the change in configuration counters the attack in [7], the chosen IV attack introduced in section 3.1 is still applicable to the first key-IV setup of Tweaked Pomaranch. A similar procedure to what explained in Section 3.1 leads to the following numbers.

Table 2. Different parameters of finding different sections of Tweaked Pomaranch

Recovered Sections	l	T	Complexity	Number of fixed part of IV's
R_1	$2^{28} \le l \le 2^{78}$	$2^{117.7}/l$	$2^{117.7}$	1
R_2, k_1	$2^{35} \le l \le 2^{52}$	$2^{104.7}/l$	$2^{104.7}$	2

After finding key part k_1, the rest part of the key can be found by exhaustive search with computational complexity $O(2^{112})$. Therefore the total computational complexity of our key-recovery attack is O($2^{117.7}$), and the required number of IV's, the imposed condition on IV's and the required number of key-stream bits from each IV are determined by Table 2 which provides many tradeoffs.

5. Conclusion

In this paper we presented a chosen IV key-recovery attack on Original Pomaranch. In our attack we used the idea of Linear Equivalence Bias which was introduced in [7, 5]. The complexity of our chosen IV attack is $O(2^{73.5})$ on Original Pomaranch which is not less than $O(2^{52})$, the complexity achieved in [1]. However, our attack is applicable to the first version of proposed key-IV setup of Tweaked Pomaranch with computational complexity of $O(2^{117.7})$ while the attack of [1] is not applicable. The second version of the proposed key-IV setup for Tweaked Pomaranch is immune against our attack.

References

1. Cid, C., Gilbert, H., Johansson, T.: Cryptanalysis of Pomaranch. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/060 (2005) `http://www.ecrypt.eu.org/stream/papersdir/060.pdf`.
2. eSTREAM, the ECRYPT Stream Cipher Project `http://www.ecrypt.eu.org/stream.`
3. Jansen C. J.A.: Stream cipher Design: Make your LFSRs jump! In SASC, Workshop Record, ECRYPT Network of Excellence in Cryptology, pages 94:108, 2004.
4. Jansen, C.J.A., Helleseth, T., Kholosha, A.: Cascade jump controlled sequence generator (CJCSG). In: Symmetric Key Encryption Workshop, Workshop Record, ECRYPT Network of Excellence in Cryptology (2005) `http://www.ecrypt.eu.org/stream/ciphers/pomaranch/pomaranch.pdf`.
5. Jansen, C.J.A., Kholosha, A.: Countering the Correlation Attack on Pomaranch. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/070 (2005) `http://www.ecrypt.eu.org/stream/papersdir/070.pdf`.
6. Jansen, C.J.A., Kholosha, A.: Pomaranch is Sound and Healthy. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/074 (2005) `http://www.ecrypt.eu.org/stream/papersdir/074.pdf`.
7. Khazaei, S.: Cryptanalysis of Pomaranch (CJCSG). eSTREAM, ECRYPT Stream Cipher Project, Report 2005/065 (2005) `http://www.ecrypt.eu.org/stream/papersdir/065.pdf`.