Slid Pairs of the Fruit-80 Stream Cipher

Kok-An Pang[1], Shekh Faisal Abdul-Latip[1] and Hazlin Abdul Rani[2]

[1]INSFORNET, Centre for Advanced Computing Technology (C-ACT),
Fakulti Teknologi Maklumat dan Komunikasi, Universiti Teknikal Malaysia Melaka.
[2]Cryptography Development Department, CyberSecurity Malaysia.

Abstract: Fruit is a small-state stream cipher designed for securing communications among resource-constrained devices. The design of Fruit was first known to the public in 2016. It was later improved as Fruit-80 in 2018 and becomes the latest and final version among all versions of the Fruit stream ciphers. In this paper, we analyze the Fruit-80 stream cipher. We found that Fruit-80 generates identical keystreams from certain two distinct pairs of key and IV. Such pair of key and IV pairs is known as a slid pair. Moreover, we discover that when two pairs of key and IV fulfill specific characteristics, they will generate identical keystreams. This shows that slid pairs do not always exist arbitrarily in Fruit-80. We define specific rules which are equivalent to the characteristics. Using the defined rules, we are able to automate the searching process using an MILP solver, which makes searching of the slid pairs trivial.

Keywords: cryptanalysis, Fruit-80, MILP, slid pairs, stream ciphers

1. Introduction

Stream ciphers play a prominent role in protecting digital communications. While there is a myriad of stream ciphers available in the literature, considerations of these ciphers for real-world applications are hindered by "security" concerns. This problem can be witnessed from several practical applications such as RC4 [25], A5/1 [9] and E0 [27]. The RC4 stream cipher, which is a well-known primitive for protecting wireless networks, has been used as the underlying algorithm for Wired Equivalent Privacy (WEP) protocol. The aim of this protocol is to provide confidentiality comparable to wired networks. Nowadays, RC4 can no longer provide a sufficient level of security as it is vulnerable to attacks such as in [14, 22]. A5/1, which is another example of stream cipher, adopted in the Global System for Mobile Communications (GSM) protocol, is a well-known primitive for securing telecommunications. Unfortunately, the A5/1 stream cipher can be broken [5, 6, 8, 12] as well, which has become one of the factors that renders GSM insecure. The E0 stream cipher, which had been implemented in Bluetooth networks, has also been practically broken [21]. Although stream ciphers encrypt faster than block ciphers, their security margins are unclear as the design principle of stream ciphers is not well understood as in block ciphers. Thus, in order to understand a “good” design principle for stream ciphers, an extensive effort to analyze existing design structures is required.

Due to limitation of resource in Internet of Things (IoT) devices, implementation of security features have become a challenge to cryptographers [1], especially in this era where IoT is on the rise. Most modern ciphers are designed to be lightweight for resource-constrained devices. However, The design of stream ciphers often involves internal states which are at least twice as large as their key size [2, 7, 10, 11, 19, 20], which requires larger area size. This design strategy is adopted due to its resistance against the time-memory-data tradeoff (TMDTO) attack, which can render the effective key length into half of the original key size. Nevertheless, this does not stop cryptographers in designing stream ciphers with small internal states which are suitable for ultra-lightweight applications such as Internet of Things. In FSE 2015, Sprout [4], a new stream cipher has been proposed with shorter internal state, allowing the construction to have a smaller area size and lower power consumption than ciphers with large internal state size. However, Sprout was later proven to be insecure [13] against the time-memory-data tradeoff attack, which breaks Sprout practically. Nevertheless, studies on small-state stream ciphers remain active, with the emergence of new design proposals such as Fruit [3], Plantlet [23] and Lizard [17] with similar design principle which are based on Grain [19]. Among all these Grain-like ciphers, Fruit has undergone several modifications [3, 16], until being finalized in [15] and has been called Fruit-80.

The Fruit-80 stream cipher has a faster initialization than Plantlet, Sprout and Lizard [15]. It is the lightest stream cipher compared with other Grain-like stream ciphers. Although small-state stream ciphers may incur TMDTO distinguishing attacks, the designers of Fruit-80 ruled out the possibility the cipher to be susceptible to this attack depending on the application scenario [15, 18]. Nevertheless, in order to avoid this attack, one of the countermeasures proposed by the designers is to limit the number of keystream bits to 2^{16}. Recently, Todo et al [31] discovered that Fruit-80 can be broken in a time complexity of $2^{77.8702}$ when 2^{43} keystream bits is allowed to be generated per one key and IV pair. Thus, by limiting the number of keystream bits to 2^{16} as suggested, the attack by Todo on Fruit-80 can be avoided as well. In this paper, we show that the initialization and keystream generation of Fruit-80 is slidable, proving that there are more than one pair of key and IV pairs that can produce the same keystream. Thus, by limiting the number of keystream bits to 2^{16} is not sufficient to strengthen the cipher from its weaknesses.

An ideal stream cipher should produce keystreams which are indistinguishable from truly-random sequence. In this paper we would like to point out that, there exists slid key-IV pairs (alternatively “slid pairs”) in Fruit-80. To begin, denote f_r and f_s as r-clock and s-clock variants of the same cipher respectively which only differ in the number of clocks. Given a key-IV pair, (x, v) and (x', v'), if both $f_{r+t}(x, v)$ and $f_{s+t}(x', v')$ can produce the same keystream for any clock $t > 0$, such key-IV pair is known as a slid pair.

The existence of slid pairs in a stream cipher clearly shows that there are more than one key-IV pair that can produce the same keystream. In principle, each keystream should only be used once. By having different messages being encrypted using the same keystream will cause a catastrophic failure to the cipher system in preserving the confidentiality of the messages. To be more precise, let c and c' be two different ciphertexts obtained by encrypting two different plaintexts m and m' using the same keystream k, such that, $c_i = m_i \oplus k_i$ and $c_i' = m_i' \oplus k_i$. Thus, by XORing c_i and c_i', we have $c_i \oplus c_i' = m_i \oplus m_i'$ which simplifies the recovery of both m_i and m_i'.

OUR CONTRIBUTION. In this paper, we investigate Fruit-80 of such behavior. We found that, by setting specific distinct key and IV pairs, Fruit-80 will generate identical keystreams with clock-shifts. Moreover, finding slid pairs in Fruit-80 is trivial. We propose specific rules for key and IV pairs to become a slid pair in Fruit-80. These rules can be translated into MILP inequalities to be solved automatically by an MILP solver. By the aid of an MILP solver, slid pairs can be generated easily. The result of our work shows that slid pairs in Fruit-80 is easy to be generated, which implies the weakness of the cipher in generating random keystreams. Moreover, by observing the slid pairs, we found that both combination of key and IV pairs in most slid pairs do not have a definite pattern, suggesting that slid pairs can also occur even if keys and IVs are randomly generated.

ORGANIZATION OF THE PAPER. In Section 2, we review the design of the Fruit-80 stream cipher. Section 3 describes the construction of MILP inequalities for MILP solver to find slid pairs. Section 4, shows how to find slid pairs in Fruit-80 and describes the result of our work. Section 5 concludes the paper.

2. A Brief Description of the Fruit-80 Stream Cipher

Fruit-80 [15] is a GRAIN-like stream cipher. It has a 37-bit non-linear feedback shift register (NFSR) n and a 43-bit linear feedback shift register (LFSR) l. It receives a 80-bit secret key k and a 70-bit initialization vector v where $k = (k_1, k_2, \dots, k_{80})$ and $v = (v_1, v_2, \dots, v_{70})$. The feedback function of NFSR is as follow.

$$\begin{aligned} n_{t+37} = \;& k_t' \oplus l_t \oplus n_t \oplus n_{t+10} \oplus n_{t+20} \oplus n_{t+12} \\ & \cdot n_{t+3} \oplus n_{t+14} \oplus n_{t+5} \cdot n_{t+23} \cdot n_{t+31} \\ & \oplus n_{t+8} \cdot n_{t+18} \oplus n_{t+28} \cdot n_{t+30} \cdot n_{t+32} \\ & \cdot n_{t+34} \end{aligned}$$

while the feedback function of LFSR is

$$l_{t+43} = l_t \oplus l_{t+8} \oplus l_{t+18} \oplus l_{t+23} \oplus l_{t+28} \oplus l_{t+37}$$

where $n \in \boldsymbol{n}$ and $l \in \boldsymbol{l}$.

The output function produces an output bit z_t in every clock. However, z_t is discarded when $0 \le t < 160$. Therefore z_{160} is the first output bit in the keystream. The output function is as follows.

$$z_t = h_t \oplus n_t \oplus n_{t+7} \oplus n_{t+19} \oplus n_{t+29} \oplus n_{t+36} \oplus l_{t+38}$$

The output function involves h function and round keys. The h function is as follows.

$$\begin{aligned} h_t = \;& k_t' \cdot (n_{t+36} \oplus l_{t+19}) \oplus l_{t+6} \cdot l_{t+15} \oplus l_{t+1} \\ & \cdot l_{t+22} \oplus n_{t+35} \cdot l_{t+27} \oplus n_{t+1} \cdot n_{t+24} \\ & \oplus n_{t+1} \cdot n_{t+33} \cdot l_{t+42} \end{aligned}$$

There are two round keys, k_t' and k_t^* used in the round function. The generation of k_t' and k_t^* is based on three selected key bits k_r, k_{p+16} and k_{q+48}.

$$\begin{aligned} k_t' = \;& k_r \cdot k_{p+16} \cdot k_{q+48} \oplus k_r \cdot k_{p+16} \oplus k_{p+16} \cdot k_{q+48} \\ & \oplus k_r \cdot k_{q+48} \oplus k_{p+16} \end{aligned}$$

$$\begin{aligned} k_t^* = \;& k_r \cdot k_{p+16} \oplus k_{p+16} \cdot k_{q+48} \oplus k_r \cdot k_{q+48} \oplus k_r \\ & \oplus k_{p+16} \oplus k_{q+48} \end{aligned}$$

The indices of the selected key bits are based on the value of a counter Cr which consists of 7 bits $Cr = (c_t^0 c_t^1 c_t^2 c_t^3 c_t^4 c_t^5 c_t^6)$, where $r = (c_t^0 c_t^1 c_t^2 c_t^3)$, $p = (c_t^1 c_t^2 c_t^3 c_t^4 c_t^5)$ and $q = (c_t^2 c_t^3 c_t^4 c_t^5 c_t^6)$.

The initialization phase is divided into three steps. The first step of the initialization starts from the first clock until the 80-th clock. Both NFSR and LFSR are first initialized with k, such that, $n_i = k_i$ and $l_i = k_{i+37}$ for $0 \le i < 37$. The second step of initialization involves overwriting the counter Cr and the last step of initialization starts from the 81-th clock to the end of initialization.

During the first initialization phase, the counter is initialized such that $c_0^i = 0$ for $0 \le i \le 6$. It is then overwritten in the second step of initialization where $c_{80}^i = n_{80+i}$ for $0 \le i \le 5$ and $c_{80}^6 = l_{80}$. The value of l_{80} is then set to 1. The overwritten counter will then be used in the third step of the initialization.

During the first step of initialization, $z_t \oplus v_t$ is XORed with both n_{t+37} and l_{t+37} respectively before entering as a new bit into NFSR and LFSR. The bit $z_t \oplus v_t$ is then disconnected from NFSR and LFSR during the third step of initialization. Figure 1 illustrates the structure of the Fruit-80 stream cipher.

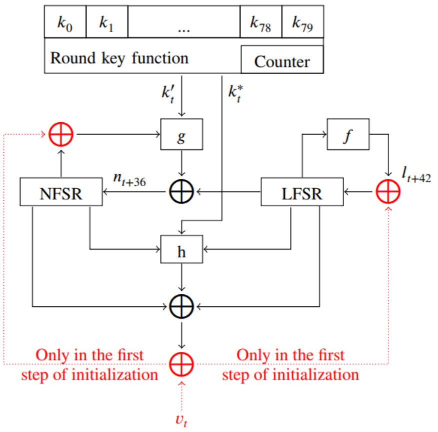

Figure 1. Structure of Fruit-80

3. The Notion of Mixed-Integer Linear Programming (MILP)

Mixed-Integer Linear Programming (MILP) [24] is a constraint programming used to determine the minimum or maximum objective of a set of linear equalitions and inequalities. In this paper, we use Gurobi Optimizer [35], which is an MILP solver, to find the key and IV pairs which can fulfill the sliding property. Gurobi Optimizer is also used in [28–30, 33, 34]. The MILP inequalities which are equivalent to AND, XOR and OR are shown in Lemma 1, 2 and 3 respectively. Note that the MILP inequalities presented in [30, 32, 34] describes the MILP inequalities for propagation of the division property of copy, xor and and. However, in this paper, we show MILP inequalities which are equivalent to the bitwise AND, XOR and OR.

Lemma 1. Let $c = \prod_{i=1}^{j} a_i$ be an AND operation. The equivalent inequalities for the AND operation is as follows.

$$c - a_i \le 0 \text{ where } 1 \le i \le j$$
$$c - a_1 - a_2 - \dots - a_j \ge 1 - j$$

Proof. By limiting c and a_i for all $1 \le i \le j$ such that $c - a_i \le 0$, the possibility that $c = 1$ while $a_i = 0$ for any i is eliminated. The constraint $c - a_1 - a_2 - \ldots - a_j \geq 1 - j$ eliminates the possibility that $c = 0$ when $a_i = 1$ for all $1 \leq i \leq j$. All other possibilities are true in bitwise AND operations. □

Lemma 2. Let $c = a_1 \oplus a_2$ be an XOR operation. The equivalent inequalities for the XOR operation are as follows.

$$\begin{gathered} c - a_1 - a_2 \leq 0 \\ c - a_1 + a_2 \geq 0 \\ c + a_1 - a_2 \geq 0 \\ c + a_1 + a_2 \leq 2 \end{gathered}$$

Proof. The inequality $c - a_1 - a_2 \leq 0$ eliminates the possibility that $c = 1$, $a_1 = 0$ and $a_2 = 0$, which is not in accordance with a bitwise XOR operation. The second inequality $c - a_1 + a_2 \geq 0$ eliminates the possibility that $c = 0$, $a_1 = 1$ and $a_2 = 0$. The third inequality $c + a_1 - a_2 \geq 0$ eliminates the possibility that $c = 0$, $a_1 = 0$ and $a_2 = 1$ while the fourth inequality $c + a_1 + a_2 \leq 2$ eliminates the possibility where $c = 1$, $a_1 = 1$ and $a_2 = 1$. The remaining possibilities are in accordance with bitwise XOR operations. □

Lemma 3. Let $c = \bigvee_{i=0}^{j} a_i$ be an OR operation. The equivalent inequalities for the OR operation are as follows.

$$c - a_i \geq 0 \text{ where } 1 \leq i \leq j$$
$$c - a\_1 - a\_2 - \ldots - a\_j \leq 0$$

Proof. By limiting c and a_i for all $1 \leq i \leq j$ such that $c - a_i \geq 0$, the possibility that $c = 0$ while $a_i = 1$ for any i is eliminated. The constraint $c - a_1 - a_2 - \ldots - a_j \geq 1 - j$ eliminates the possibility that $c = 1$ when $a_i = 0$ for all $1 \leq i \leq j$. All other possibilities are true in a bitwise OR operation. □

Lemma 3 is equivalent to Proposition 2 in [34]. Note that Todo et al [30] claim that $c - a_1 - a_2 - \ldots - a_j \leq 0$ is redundant and does not affect their result even if the inequality is not included. However, we found that there is a possibility for $c = 1\ and\ \sum_{i=1}^{j} a_i = 0$, which does not correspond to a bitwise OR operation. Therefore, we adopt the MILP inequalities introduced in [34] to avoid this possibility.

Both k'_t and k^*_t are determined by three secret key bits k_r, k_{p+16} and k_{q+48}. We show the equations for determining the value of k_r, k_{p+16} and k_{q+48} in Lemma 4.

Lemma 4. Let ℓ be the length of subkey; γ as the starting index of key bits for k_r, k_{p+16} and k_{q+48} respectively. Next, let α be the starting index of the respective counter bits used by either r, p or q; while β representing the last index of the respective counter bits. Then, the value of k_r, k_{p+16} and k_{q+48} can be determined by assigning the respective values of ℓ, α, β and γ to the following expression.

$$\sum_{i=\gamma}^{\gamma+l-1}\left(k_i \cdot \left(1 - \left(\bigvee_{j=\alpha}^{\beta}\left(c_t^j \oplus \left\lfloor \frac{i-\gamma}{2^{\beta-j}} \right\rfloor mod\ 2\right)\right)\right)\right)$$

Proof. From the round key function of Fruit-80 (cf. Section 2), we know that $k_r \in \{k_0, k_1, \ldots, k_{15}\}$. Therefore, we consider $k_r = \sum_{i=0}^{15}(k_i \cdot \omega)$, where $\omega \in \mathbb{F}_2$ takes a value of 1 if and only if $i = (c_t^0 c_t^1 c_t^2 c_t^3)$. The index i can be viewed as a vector $i = (i_0 i_1 i_2 i_3 i_4 i_5 i_6)$, in which $i_j = \left\lfloor \frac{i}{2^{6-j}} \right\rfloor mod\ 2$. Since $r = (c_t^0 c_t^1 c_t^2 c_t^3)$, we focus on obtaining $(i_0 i_1 i_2 i_3)$ with $i_j = \left\lfloor \frac{i}{2^{3-j}} \right\rfloor$. Note that $(c_t^0 c_t^1 c_t^2 c_t^3) \oplus (i_0 i_1 i_2 i_3) = 0$ if and only if $(i_0 i_1 i_2 i_3) = (c_t^0 c_t^1 c_t^2 c_t^3)$. In this case, the bitwise OR operation $\bigvee_{j=0}^{3}\left(c_t^j \oplus \left\lfloor \frac{i}{2^{3-j}} \right\rfloor mod\ 2\right)$ can help in distinguishing whether $(i_0 i_1 i_2 i_3) = (c_t^0 c_t^1 c_t^2 c_t^3)$ by returning value 0 when it is true or value 1 if otherwise. However, ω works the opposite of $\bigvee_{j=0}^{3}\left(c_t^j \oplus \left\lfloor \frac{i}{2^{3-j}} \right\rfloor\ mod\ 2\right)$, thus $\omega = 1 - \bigvee_{j=0}^{3}\left(c_t^j \oplus \left\lfloor \frac{i}{2^{3-j}} \right\rfloor mod2\right)$. Therefore, $k_r = \sum_{i=0}^{15}\left(k_i \cdot \left(1 - \left(\bigvee_{j=0}^{3}\left(c_t^j \oplus \left\lfloor \frac{i}{2^{3-j}} \right\rfloor\ mod\ 2\right)\right)\right)\right)$. We apply the same procedure for k_{p+16} and k_{q+48}. Since $k_{p+16} \in \{k_{16}, k_{17}, \ldots, k_{47}\}$, then $k_{p+16} = \sum_{i=16}^{47}\left(k_i \cdot \left(1 - \left(\bigvee_{j=1}^{5}\left(c_t^j \oplus \left\lfloor \frac{i-16}{2^{5-j}} \right\rfloor\ mod\ 2\right)\right)\right)\right)$; and since $k_{q+48} \in \{k_{48}, k_{49}, \ldots, k_{79}\}$, then $k_{q+48} = \sum_{i=48}^{79}\left(k_i \cdot \left(1 - \left(\bigvee_{j=2}^{6}\left(c_t^j \oplus \left\lfloor \frac{i-48}{2^{6-j}} \right\rfloor\ mod\ 2\right)\right)\right)\right)$. The expressions for finding k_r, k_{p+16} and k_{q+48} can then be generalized to $\sum_{i=\gamma}^{\gamma+l-1}\left(k_i \cdot \left(1 - \left(\bigvee_{j=\alpha}^{\beta}\left(c_t^j \oplus \left\lfloor \frac{i-\gamma}{2^{\beta-j}} \right\rfloor\ mod\ 2\right)\right)\right)\right)$. Hence, we require to give the respective value of α, β, γ and l to the expression, such that, $\alpha = 0$, $\beta = 3$, $\gamma = 0$ and $l = 16$ (for finding k_r); $\alpha = 0$, $\beta = 3$, $\gamma = 0$ and $l = 16$ (for finding k_{p+16}); and, $\alpha = 2$, $\beta = 6$, $\gamma = 48$ and $l = 32$ (for finding k_{q+48}). □

Lemma 4 can be converted into MILP equations by making use of the MILP inequalities shown in Lemma 1, Lemma 2 and Lemma 3 accordingly.

4. Finding Slid Pairs on Fruit-80

A general rule for a slid pair to exist in a stream cipher is to have the first starting state initialized with (k, v) to produce the second starting state which can be initialized with (k', v') at different clock-shifts [26]. The key factor for two key and IV pairs to generate identical keystreams is due to the symmetrical structure between round functions after the first starting state, and round functions after the second starting state. To be more precise, the round function at clock t must be symmetrical to the round function at clock $t + u$ in order to produce keystreams with u clock-shifts. There are several components in Fruit-80 that may remove the symmetry. However, when certain variables (as explained later in Rule 1, 2, 3 and 4) hold specific values, every round function at clock t behaves similarly to the round function at clock $t + u$, thus allowing slidable keystreams to be generated. Moreover, we notice that slid pairs does not always exist arbitrarily. We have found specific rules in which, when all rules being fulfilled by two pairs of key and IV, Fruit-80 generates identical keystreams. We define the rules as follows:

Rule 1. k'_t and k^*_t must be the same for all clocks in the first and third steps of the initialization.

For each clock, k'_t and k^*_t are generated based on the counter Cr (cf. Section 2). The counter is initially set to 0. It is then overwritten after the first step of the initialization based on the value of k. Thus, the variables k_t' and k_t^* may provide assymetry between the first step and the third step of the initialization. Since k_t' and k_t^* are not hardcoded, there are possibilities for the value of k_t' and k_t^* are not changed during the third step of the initialization, even if the counter is updated with different values. Thus, it is not possible for a slid pair to occur when the values of k_t' and k_t^* during the first step of the initialization is different from the values during the third step of the initialization. Therefore, it is required for k_t' and k_t^* to be the same for all clocks in both steps.

Rule 2. l_{80} and l_{80+u} must be set to 1.

After the overwriting of Cr, the value of l_{80} is set to 1 to prevent the LFSR from being all zeros throughout the third initialization step. By default, the value of l_{80} will be set to 1 during the 81-st clock. However, the value of l_{80+u} may differ from l_{80}. If this happens, it will prevent the symmetry between the round function on the 81-st clock and the round function at the $(81+u)$-th clock. Thus, to ensure the symmetry between these two clocks, we set l_{80+u} to 1 as well.

Rule 3. $z_t \oplus v_t = 0$, for all clocks in the first and third steps of the initialization.

During the first step of the initialization, the feedback functions for both NFSR and LFSR are XORed with $z_t \oplus v_t$. After the first step of initialization, $z_t \oplus v_t$ are disconnected from the feedback functions, results in $z_t \oplus v_t = 0$. Thus, the disconnection of $z_t \oplus v_t$ may provide assymetry between the first step and the third step of the initialization. It is not possible for a slid pair to occur when $z_t \oplus v_t = 1$ during the first step of the initialization. Therefore, it is required that $z_t \oplus v_t = 0$ for all clocks in the first step of the initialization.

Rule 4. $v_t = v_{t+u}$, for all $0 \leq t < 10$.

During the key and IV loading, the 70-bit v will be padded with 10 constant bits to form an 80-bit v. The 10 padded bits are used in clock $0 \leq t < 10$. The constant values may induce asymmetry between the first 10 clocks after the starting state and the first 10 clocks after the second starting state. To remove the asymmetry, we set $v_{t+u} = v_t$ for all $0 \leq t < 10$.

Table 1: The number of slid pairs in Fruit-80 based on the number of clock-shifts

Number of clock-shifts	Number of Slid Pairs
20	$2^{12.81}$
40	$2^{13.32}$
60	$2^{12.99}$
80	$2^{12.08}$

With these predefined rules, we can automate the process of finding these slid pairs using an MILP solver. We run our experiment for several weeks using a single PC to find slid pairs applying our predefined rules. From our experiment we have been able to find $2^{12.81}$ slid pairs considering a 20-clock shift. We continue our experiment considering other number of clock-shifts, i.e. 40, 60, and 80 clocks and have been able to find $2^{13.32}$, $2^{12.99}$ and $2^{12.08}$ slid pairs respectively. We summarize our findings in Table 1. Examples of slid pairs obtained through several clock-shifts are listed in Appendix A.

5. Conclusions

We investigated the existence of slid pair in the Fruit-80 stream cipher that can generate an identical keystream. We employed an MILP solver to help us finding these pairs by applying our predefined rules. Our result shows that slid pairs in Fruit-80 can be found with deterministic rules. This proves that it is possible for Fruit-80 to generate the same keystream after resynchronization if the key and IV are not properly selected. Our result should provide insights to designs of more secure and random stream ciphers for constrained environments.

6. Acknowledgement

The authors would like to express sincere appreciation to CyberSecurity Malaysia for supporting this research.

References

[1] Ali M.A. Abuagoub, “Iot security evolution : challenges and countermeasures review,” International Journal of Communication Networks and Information Security, pp. 342–351, 2019.

[2] M. Ågren, M. Hell, T. Johansson, W. Meier, “Grain-128a: a new version of Grain-128 with optional authentication,” International Journal of Wireless and Mobile Computing, pp. 48-59, 2011.

[3] A. Ghafari, H. Hu, “Fruit: ultra-lightweight stream cipher with shorter internal state,” IACR Cryptology ePrint Archive, pp. 1-5, 2016.

[4] F. Armknecht, V. Mikhalev, “On lightweight stream ciphers with shorter internal states,” International Workshop on Fast Software Encryption, Instanbul, Turkey, pp. 451-470, 2015.

[5] E. Barkan, E. Biham, “Conditional estimators: an effective attack on a5/1,” International Workshop on Selected Areas in Cryptography, Kingston, ON, Canada, pp. 1-19, 2005.

[6] E. Barkan, E. Biham, N. Keller, “Instant ciphertext-only cryptanalysis of gsm encrypted communication,” Annual International Cryptology Conference, Santa Barbara, CA, USA, pp. 600-616, 2003.

[7] D.J. Bernstein, “The salsa20 family of stream ciphers,” New Stream Cipher Designs, pp. 84-97, 2008.

[8] E. Biham, O. Dunkelman, “Cryptanalysis of the a5/1 gsm stream cipher,” International Conference on Cryptology in India, Calcutta, India, pp. 43-51, 2000.

[9] A. Biryukov, A. Shamir, D. Wagner, “Real time cryptanalysis of a5/1 on a pc,” International Workshop on Fast Software Encryption, New York, USA, pp. 1-18, 2000.

[10] M. Boesgaard, M. Vesterager, T. Pedersen, J. Christiansen, O. Scavenius, “Rabbit: a new high-performance stream cipher,” International Workshop on Fast Software Encryption, Lund, Sweden, pp. 307-329, 2003.

[11] D. C. Canniere, “Trivium: a stream cipher construction inspired by block cipher design principles,” International Conference on Information Security, Samos, Greece, pp. 171-186, 2006.

[12] P. Ekdahl, T. Johansson, “Another attack on a5/1,” IEEE Transactions on Information Theory, pp. 284-289, 2003.

[13] M.F. Esgin, O. Kara, “Practical cryptanalysis of full sprout with tmd tradeoff attacks,” International Conference on Selected Areas in Cryptography, Sackville, Canada, pp. 67-85, 2015.

[14] S. Fluhrer, I. Mantin, A. Shamir, “Weaknesses in the key scheduling algorithm of rc4,” International Workshop on Selected Areas in Cryptography, Toronto, ON, Canada, pp. 1-24, 2001.

[15] V.A. Ghafari, H. Hu, “Fruit-80: a secure ultra-lightweight stream cipher for constrained environments,” Entropy, pp. 1-3, 2018.

[16] V.A. Ghafari, H. Hu, M. Alizadeh, “Necessary conditions for designing secure stream ciphers with the minimal internal states,” IACR Cryptology ePrint Archive, pp. 1-16, 2017.

[17] M. Hamann, M. Krause, W. Meier, “Lizard – a lightweight stream cipher for power-constrained devices,” IACR Transactions on Symmetric Cryptology, pp. 45-79, 2017.

[18] M. Hamann, M. Krause, W. Meier, B. Zhang, “Design and analysis of small-state grain-like stream ciphers,” Cryptography and Communications, pp. 803-834, 2018.

[19] M. Hell, T. Johansson, A. Maximov, W. Meier, “A stream cipher proposal: grain-128,” 2006 IEEE International Symposium on Information Theory, Seattle, WA, pp. 1-6, 2006.

[20] Y. Lee, K. Jeong, J. Sung, S. Hong, “Related-key chosen iv attacks on grain-v1 and grain-128,” Australasian Conference on Information Security and Privacy, Wollongong, NSW, Australia, pp. 321-335, 2008.

[21] Y. Lu, W. Meier, S. Vaudenay, “The conditional correlation attack: a practical attack on Bluetooth encryption,” Annual International Cryptology Conference, Santa Barbara, CA, USA, pp. 97-117, 2005.

[22] I. Mantin, A. Shamir, “A practical attack on broadcast rc4,” International Workshop on Fast Software Encryption, Yokohama, Japan, pp. 152-164, 2001.

[23] V. Mikhalev, F. Armknecht, C Müller, “On ciphers that continuously access the non-volatile key,” IACR Transactions on Symmetric Cryptology, pp. 52-79, 2017.

[24] N. Mouha, Q. Wang, D. Gu, B. Preneel, “Differential and Linear Cryptanalysis using Mixed-Integer Linear Programming ⋆,” International Conference on Information Security and Cryptology, Beijing, China, pp. 57-76, 2011.

[25] G. Paul, S. Maitra, “RC4 Stream Cipher and Its Variants,” 2011.

[26] D. Priemuth-Schmid, A. Biryukov, “Slid pairs in salsa20 and trivium,” International Conference on Cryptology in India, Kharagpur, India, pp. 1–14, 2008.

[27] Y. Shaked, A. Wool, “Cryptanalysis of the bluetooth e0 cipher using obdd’s,” International Conference on Information Security, Samos, Greece, pp. 187-202, 2006.

[28] Khan, J. Loo, “Milp-aided bit-based division property for arx-based block cipher,” IACR Cryptology ePrint Archive, pp. 1-37, 2016.

[29] L. Sun, W. Wang, M. Wang, “Milp-aided bit-based division property for primitives with non-bit-permutation linear layers,” IET Information Security, pp. 12-20, 2020.

[30] Y. Todo, T. Isobe, Y. Hao, W. Meier, “Cube attacks on non-blackbox polynomials based on division property,” IEEE Transactions on Computers, pp. 1720-1736, 2018.

[31] Y. Todo, W. Meier, K. Aoki, “On the data limitation of small-state stream ciphers: correlation attacks on fruit-80 and plantlet,” International Conference on Selected Areas in Cryptography, Waterloo, ON, Canada, pp. 1-29, 2020.

[32] Y. Todo, M. Morii, “Bit-based division property and application to simon family,” International Conference on Fast Software Encryption, Bochum, Germany, pp. 357-377, 2016.

[33] Q. Wang, Y. Hao, Y. Todo, C. Li, T. Isobe, W. Meier, “Improved division property based cube attacks exploiting algebraic properties of superpoly,” Annual International Cryptology Conference, Santa Barbara, CA, USA, pp. 275-305, 2018.

[34] Z. Xiang, W. Zhang, Z. Bao, D. Lin, “Applying milp method to searching integral distinguishers based on division property for 6 lightweight block ciphers,” International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, pp. 648-678, 2016.

[35] Gurobi - The fastest solver. https://www.gurobi.com/

A. Results on Selected Slid Pairs

We investigated the existence of slid pair in the Fruit-80 stream cipher

A.1. A Slid Pair with a 20 clock-shifts

Key	:	C67BFFED58E7F743695F
IV	:	310800001289621C08
Keystream	:	AA2E4 94FEB25422A1914CA6ADA49…

Key	:	FED5B6F3FC3695FD79D6
IV	:	0001289621C08AD1A1
Keystream	:	94FEB25422A1914CA6ADA49…

A.2. A Slid Pair with a 40 clock-shifts

Key	:	C625020054820218AE54
IV	:	2D369DF8800010A874
Keystream	:	444C31ADDA 498B156E08324F36FC2F4F9…

Key	:	58BB9FF1E406BCBE1C0F
IV	:	0010A874677CD5BBFA
Keystream	:	498B156E08324F36FC2F4F9…

A.3. A Slid Pair with a 60 clock-shifts

Key	:	1012932001519735626A
IV	:	1A9E5A63999BDA0117
Keystream	:	9D6747794A93E12 06F37DEC94E5A5024C41009…

Key	:	7BB65D5F3E7FA3DFFAAB
IV	:	11723DDA7AA63C3C4C
Keystream	:	06F37DEC94E5A5024C41009…

A.4. A Slid Pair with an 80 clock-shifts

Key	:	874F13DB769E49B609F6
IV	:	22611AFA9CAE02E3FF
Keystream	:	3B13EF3DFF53D34E3F8F 9E208C2C18F431D2F415…

Key	:	520AC8AB352ABAFF97AA
IV	:	13A93C3727FFA229C3
Keystream	:	9E208C2C18F431D2F415…