Linearisation Attacks on FCSR-based Stream Ciphers

Arshad Ali
Information Security Group
Royal Holloway, University of London
Egham, Surrey, TW20, 0EX, UK

Abstract

This paper presents a new class of cryptanalytic attacks, which are applicable against those binary additive synchronous stream ciphers, whose analysis theory is based on the properties of 2-adic numbers. These attacks are named as 'Linearisation Attacks'. Linearisation attacks consist of three variants, which are referred as 'Conventional Linearisation Attacks (CLAs)', 'Fast Linearisation Attacks (FLAs)' and 'Improved Linearisation Attacks (ILAs)'. The paper demonstrates, these new attacks against F-FCSR-H and an eSTREAM candidate F-FCSR-H v2 stream ciphers by presenting state recovery algorithms based on CLAs, FLAs and ILAs. The paper analyses the efficiency of these attacks in terms of data, time and memory complexities. The paper also presents a comparative analysis of these attacks, which shows that these variants provide a tradeoff of various complexity measures, i.e. data, time and memory complexities when compared with each other. Further more, the paper also presents a key recovery algorithm, which works in conjunction with the state recovery algorithms to recover the effective key used in these ciphers.

Keywords - Cryptanalysis, Most Efficient, Linearisation Attacks, CLAs, FLAs, ILAs, eSTREAM, F-FCSR-H v2.

I. INTRODUCTION

This paper is an extended version of [1] and presents a new class of crypt-analytical attacks, which are applicable against those ciphers, whose analysis theory is based on the properties of 2-adic numbers. This new class of crypt-analytic attacks is named as 'Linearisation Attacks' and it consists of three variants, which are named as 'Conventional Linearisation Attacks (CLAs)', 'Fast Linearisation Attacks (FLAs)' and 'Improved Linearisation Attacks (ILAs)'. The paper demonstrates the Linearisation attacks with reference to F-FCSR [3,4] family of binary additive synchronous stream ciphers. The ciphers of this family consist of a Galois Feedback with Carry Shift Register (FCSR) [11], whose states are filtered by using linear Boolean functions. In this paper, we crypt-analyse an eSTREAM candidate F-FCSR-H v2 [5], and another variant of this cipher, which is known as F-FCSR-H. These ciphers are considered important members of the F-FCSR family of stream ciphers. It is pertinent to mention that, F-FCSR-H v2 was initially included in the eSTREAM portfolio for hardware profile [13], but after discovery of some weaknesses [8,9] in the key-stream generator, it was removed from the revised portfolio [14].

Linearisation attacks on such ciphers are based on a new observation, which is also described in this paper. To cryptanalyse F-FCSR-H v2 and F-FCSR-H stream ciphers, we present state recovery algorithms based on CLAs, FLAs and the ILAs for these ciphers. A comparative analysis of these attacks is also presented, which shows that these attacks are the most efficient attacks on these ciphers. Further more, the comparative analysis also shows that CLAs, FLAs and ILAs provide a tradeoff between data, time and memory complexities, when compared with each other.

The rest of the paper is organised as follows. In Section 2, we briefly describe the operation of a Galois FCSR and the construction of the F-FCSR-H v2. Sections 3 to 8 contain the main results of this paper. In Section 3, we present the new class of cryptanalytic attacks, which is named as 'Linearisation Attacks'. In this section, we present three variants of linearisation attacks, i.e. CLAs, FLAs and the ILAs. In Sections 4,5,6, we describe the applications of linearisation attacks on F-FCSR-H v2 and present state recovery algorithms based on CLAs, FLAs and ILAs to recover the internal states of F-FCSR-H v2 and F-FCSR-H stream ciphers. In Section 7, we present a comparative analysis of these attacks on F-FCSR-H v2 key-stream generator, in terms of data, time and memory complexities. In Section 8, we present a key recovery algorithm, which works in conjunction with the state recovery algorithms and recovers the effective key used in these ciphers. Finally, we conclude this paper in Section 9.

II. BRIEF DESCRIPTION OF F-FCSR-H V2

In this section, we give a brief description of F-FCSR-H and F-FCSR-H v2 key stream generators. These key-stream generators use the same filtered Galois FCSR [11] to produce the key stream for encryption or decryption. A Galois FCSR is a finite state machine, which consists of a main register, a carry register and a feedback function. These devices, i.e. FCSRs are used to compute the binary expansions of 2-adic numbers of the form $\frac{p}{q}$, $p, q \in \square$ and $0 < p < |q|$. Consider an odd integer q and let $r = \lfloor \log_2(q+1) \rfloor$. Let q be represented in the binary form as follows.

Copyright © 2012, Infonomics Society

$$q = 2q_1 + 2^2 q_2 + 2^3 q_3 + \cdots + 2^r q_r - 1 \qquad (1)$$

Where $q_i \in \{0,1\}$ for every $i \in [0, r-1]$ and $q_{r-1} = 1$. Let $2d = 1 + |q|$. Then r defines the bit-length of d. The binary representation of d is given by $d = \sum_{i=0}^{r-1} d_i 2^i$, where $d_i \in \{0,1\}$ for all $i \in [0, r-2]$ and $d_{r-1} = 1$. Let ω denotes one less than the Hamming weight (wt) of the binary representation of d, i.e. $\mathrm{wt}(d) = \omega + 1$.

Define $\mathrm{Id} = \{i \mid (0 \le i < r-1) \wedge (d_i = 1)\}$. Then a Galois FCSR consists of two registers, namely, the main and the carry registers, which are denoted by M and C, respectively. The main register consists of r binary cells, which are denoted by $\{m_i\}_{i=0}^{r-1}$, and at time t, the contents of these cells are denoted by $\{m_i(t)\}_{i=0}^{r-1}$. Similarly, the carry register consists of $r-1$ binary cells, which are denoted by $\{c_i\}_{i=0}^{r-2}$, and at time t, the contents of these cells are denoted by $\{c_i(t)\}_{i=0}^{r-2}$. The content of a cell c_i is always 0, if $d_i = 0$, such cells are considered in active. This means that the carry register contains only ω active cells.

Suppose at time t, the FCSR is in state $(m(t), c(t))$. The next state, which is denoted by $(m(t+1), c(t+1))$, is computed from the current state according to the following rules:

1. For every integer $i \in [0, r-2] \wedge i \notin \mathrm{Id}$; $\mathrm{m}_i(t+1) = m_{i+1}(t)$
2. For every integer $i \in [0, r-2] \wedge i \in \mathrm{Id}$; $\mathrm{m}_i(t+1) = m_{i+1}(t) \oplus c_i(t) \oplus m_0(t)$ and $c_i(t+1) = m_{i+1}(t)c_i(t) \oplus c_i(t)m_0(t) \oplus m_0(t)m_{i+1}(t)$
3. For $i = r-1$; $m_i(t+1) = m_0(t)$

The number q, which is associated with every FCSR as shown in Eq.1 is also known as connection integer of the FCSR. The connection integer of a FCSR determines the periodicity of the FCSR. If q is prime such that 2 is a primitive element in the ring of units $\mathbb{Z}/(q)^*$, then the sequence generated by the FCSR has maximum period and is known as l-sequence.

Definition 2.1 Consider a sequence $S = (s_0, s_1, \ldots)$ of period n, and let $b = (b_0, b_1, \ldots, b_{k-1})$ denotes a block of size k defined over the same alphabet. The occurrence of block b in the sequence S, means there exists an integer $i \in [0, n-1]$ such that $s_i = b_0, s_{i+1} = b_1, \ldots, s_{i+k-1} = b_{k-1}$.

Following theorem describes the distribution of blocks in l-sequences, which is useful to estimate the complexities of various state recovery algorithms presented in the following sections to recover the internal states of F-FCSR-H v2 and F-FCSR-H stream ciphers.

Theorem 1 (Klapper and Goresky, [7], Theorem 2) Let $S = (s_0, s_1, \ldots)$ be an l-sequence generated by a FCSR with connection integer q. Then the number of occurrences of any block $b = (b_0, b_1, \ldots, b_{k-1})$ of size k in S varies at most by 1as the block varies over all 2^k possibilities. That is, there is an integer ω so that every block of length k occurs either ω times or $\omega+1$ times in S. The number of blocks of length k that occur $\omega+1$ times are $(q-1)(\mathrm{mod}\, 2^k)$, and the number of blocks of length k that occur ω times is $(2^k - q + 1)(\mathrm{mod}\, 2^k)$.

F-FCSR-H v2 generator filters the main register states of a Galois FCSR with a linear Boolean function to generate the pseudo-random key-stream. The linear Boolean function used in these schemes is denoted by F and is of the form: $F : F_2^r \to F_2$. This filter can be viewed as a bit-string of length r, which can be expressed as:

$$F = \sum_{i=0}^{r-1} f_i 2^i \quad \text{where} \quad f_{r-1} = 1$$

To produce the key-stream, F is applied on the main register states:

$$M = (m_0(t), m_1(t), \ldots, m_{r-1}(t)) \quad \text{for} \quad t \ge 0$$

A generalised expression for computing the output sequence S of such a generator using a one-bit filter is given below:

$$S = (s(t))_{t \ge 0} \quad \text{where} \quad s(t) = \langle F \cdot M(t) \rangle \quad \text{for} \quad t \ge 0$$

The prime number q, which is used as connection integer in the Galois FCSR of F-FCSR-H cipher is given below.

$$q = 1993524591318275015328041611344215036460140087963$$

In this case $\omega = 82$. The parameter d is used to filter the main register states of the Galois FCSR for producing the key-stream. This filter is split into eight sub-filters, which are denoted by $F_0, F_1, \ldots, F_7$. These sub-filters are publicly known as part of the algorithm and are given below (in hex form).

Copyright © 2012, Infonomics Society

$$F_0 = 374AA, F_1 = 9ADC1, F_2 = BBAEF, F_3 = F2389$$
$$F_4 = 7223C, F_5 = 9C48A, F_6 = 35265, F_7 = D3BB4$$

With these sub filters, the F-FCSR-H and F-FCSR-H v2 generators produce one byte of key-stream in such a way that each sub-filter generates 1 bit per clock cycle.

We consider a cryptanalytic attack as an algorithm, which exploits some bias in the function of the generator. To estimate the time complexity of such an algorithm, we consider one step of the algorithm as one primitive operation and define the worst and the average case running time complexities as follows.

Definition 2.2 The worst-case running time complexity of an algorithm expressed, as a function of the input is an upper bound on the running time for any input. Similarly, the average case running time complexity of an algorithm is the average running time expressed as a function of the input over all inputs of a fixed size.

III. LINEARISATION ATTACKS

Linearisation attacks are based on the theory of linearisation intervals (See Definition 3.1) to exploit the time dependent inherent non-linearity of the primitives such as FCSRs, which are used as pseudo-random sequence generators in the cryptographic schemes. These attacks use correlations between various parts of the internal states of the ciphers and thus have close resemblance with the correlation attacks. Further more, these attacks are different from linear cryptanalysis, which mainly rely on time independent statistical linear relations between the input and output of the non-linear primitives such as S-boxes used in block ciphers. These attacks also differ from techniques of linearising the attack equations. These types of techniques are more effectively used in algebraic attacks and are usually referred to as linearisation methods or linearisation techniques to solve algebraic equations in finite fields. In such methods, linearisation of the attack equation is achieved by replacing a monomial of degree more than 1 with a new independent variable of degree 1. Study of linear structures in block ciphers [6] and cryptanalysis of syndrome-based hashes [12] are two more examples, which use techniques based on linearisation but in a different context, than our approach used to define the linearisation attacks. In general, such techniques are probabilistic and time independent, which transform non-linear transformations into linear or affine transformations.

The linearisation attacks described in this paper combine a variety of concepts relating to deterministic and probabilistic techniques, which exploit particular cipher structures with the generalized techniques such as linear, correlation and guess-and-determine attacks. The combinations of various techniques make these attacks more successful against those ciphers whose analysis theory depends on the properties of 2-adic numbers. In the context of FCSR-based ciphers, these attacks exploit the inherent non-linearity of the FCSRs in a time dependent fashion. These attacks are particularly more effective against the class of FCSR-based ciphers, which use, linear Boolean functions to filter the internal states of the key-stream generator. In such cipher secret key and IV are used to define the initial state of the ciphers.

Linearisation attacks are known plaintext attacks. The attack scenario of linearisation attacks is based on Kerkhoff's principle [10], which means that the attacker knows full details of the cipher and its implementation. The goal of the cryptanalyst is to re-construct the whole key-stream by recovering the internal state of the cipher given the key-stream $\{z_i\}_{i=0}^{N-1}$ for some $N \in \square$. The integer N represents the length of the known key-stream required for the attack to succeed, and hence it determines the computational effort required for a successful cryptanalytic attack. Therefore, the parameter N is useful to determine various complexity measures of the attack.

Definition 3.1[2] Consider a binary Galois FCSR, consisting of r binary cells in its main register. Let $M_h(t)$ denote the Hamming weight of the main register M at time t. Then during the operation of such a FCSR, the integers $I \in \square$, $I < r$ such that either of the following two equations holds for all integers $i \in [0, I[$

$$M_h(t+i+1) = M_h(t+i) \ \forall i \in \square \text{ s.t } 0 \le i < I \cdots (1)$$

$$M_h(t+i+1)+1 = M_h(t+i) \ \forall i \in \square \text{ s.t } 0 \le i < I \cdots (2)$$

Such contiguous intervals of duration I are defined as linearisation intervals and are denoted by LI.

There are three types of linearisation attacks, which are tradeoffs between data, time and memory complexities when compared with each other. In the context of FCSR-based ciphers, these attacks are briefly described in the following sections.

A. *Conventional Linearisation Attacks*

The CLAs on FCSR-based ciphers exploit only the inherent non-linearity of the FCSRs and use only the value of the feedback bit m_0 during the linearisation intervals (See Definition 3.1). This means, CLAs use the correlations between the feedback bit and the carry register of the FCSR. Further more, these attacks do not require any additional information relating to any other part of the internal state of the ciphers.

B. *Fast Linearisation Attacks*

The FLAs on FCSR-based ciphers exploit the correlations between the feedback bits; the carry register and some portions of the main register cells. This means, the FLAs in addition to using the linearisation intervals also make use of the 3-dimensional correlation between the feedback bit, the carry register and the Hamming weight of the main register of Galois FCSRs. These correlations show that the FCSR sequences observed in

Copyright © 2012, Infonomics Society

various main register cells are dependent, correlated and identically distributed (See [2] for more details). As shown in [2] that during the linearisation intervals of duration n (say), n cells on both edges of the FCSR main register are correlated. This enables us to determine exactly n cells on both edges of the main register during the linearisation intervals. The FLAs make use of this additional information. Further more, these attacks are more successful against those ciphers, which use linear Boolean functions to filter the main register states of Galois FCSRs.

C. *Improved Linearisation Attacks*

The ILAs combine the correlations between the feedback bits, the carry register and the Hamming weight of the main register with the guess-and-determine approach. The main goal of this novel approach used in ILAs is to reduce the upper bound for the known key-stream. This results in increasing the efficiency of the linearisation attacks. In ILAs, we guess a small portion of the internal state of the cipher and determine its accuracy by using exhaustive search technique. This approach of combining various pre-determined and guessed parameters, results in improving the data and memory complexities at the expense of some extra computations, which slightly increase the running time complexities of these attacks when compared with CLAs and FLAs. However, from the cryptanalytic point of view, the gain in the data complexity is more valuable, because otherwise, the attack fails to recover the internal state of the cipher.

IV. State Recovery Algorithm Based on CLAs for F-FCSR-H v2 and F-FCSR-H

In this section, we present Algorithm 1, which is a state recovery algorithm based on the CLAs and recovers the internal states of the F-FCSR-H v2 and F-FCSR-H stream ciphers by using some known key-stream.

Algorithm 1 is an iterative procedure, which requires some known key-stream and the linear Boolean function, which is used to filter the main register states, as inputs and gives as output the internal state of the FCSR. The linear Boolean function, which is used in F-FCSR-H and F-FCSR-H v2 ciphers, extracts one byte during each clock cycle of the generator in such a way that each 20-bit sub filter extracts one bit. Therefore, if we have a linearisation interval of duration 20 contiguous clock cycles of the generator, then the systems of equations formulated in step-1.3 of Algorithm 1 become consistent and hence gives unique solution. Assume that, during these 20 clock cycles, the FCSR states be denoted by $p(t), p(t+1),\ldots, p(t+19)$, and then Algorithm 1 recovers the internal state $p(t)$. In the verification step, we use these state variables to initialize the FCSR and compare the output with the known values of key-stream bytes at times $t, t+1, t+2$.

Algorithm 1 requires 8 pre-computed P-matrices, which are computed by simulating the behavior of the filter for 20 contiguous clock cycles under the assumption that the generator functions in the linearisation interval for these 20 contiguous clock cycles.

Algorithm 1. State recovery algorithm based on CLAs for F-FCSR-H v2 and F-FCSR-H

Input 1) F, $N \in \mathbb{Z}$ such that $N \geq 20$

2) $z_0, z_1, \ldots, z_N$

Output $p(t)$

Set $T \leftarrow 0, n \leftarrow 0, m_0 \leftarrow 0$

Off-line a) formulate 1 x 20 matrices

$M_j = \{m_i \mid 0 \leq i \leq 159 \wedge i \equiv j \pmod 8\}$

b) compute $P_0, P_1, \ldots, P_7$

On-line

1. while $(n < N)$ do

1.1) take $z(n+m)$ such that $0 \leq m \leq 19$

1.2) formulate 20x1 matrices $W_0, W_1, \ldots, W_7$

1.3) formulate $M_i \cdot P_i = W_i \quad \forall i \in [0,7]$

1.4) solve the systems of equations (step-1.3) to recover

m_i for $1 \leq i \leq 159$

1.5) load the recovered values in M and C

1.6) generate KSB_i for $i = 0,1,2$

1.7) if $z(i+n) == KSB_i \quad \forall i \in \{0,1,2\}$

3.7.1) $T \leftarrow 1$

3.7.2) goto 4

1.8) $n \leftarrow n+1$

2. if $T == 1$ then return $p(t)$

else return no solution.

It is pertinent to mention that the online phase of Algorithm 1 is computationally most expensive step of this attack. Therefore, analysis of this phase determines the data, time and memory complexities of the attack, which are considered as main metrics to determine the efficiency of any cryptanalytic attack.

Theorem 1 shows that the worst-case data complexity of Algorithm 1 is equal to the key-stream generated during $2^{\lfloor \log_2(82) \rfloor + 19} = 2^{25}$ advances of the generator. Note that during these 2^{25} advances of the generator, the generator functions in the linearisation interval for at least 20 contiguous advances of the generator. These 20 contiguous advances of the generator, during the linearisation interval produce 20 bytes of key-stream. These 20 bytes are used to formulate eight 20x20 systems of equations. Since the F-FCSR-H and F-FCSR-H v2 generators produce one byte of key-stream per clock cycle, therefore, the worst-case data complexity of

Copyright © 2012, Infonomics Society

Algorithm 1 is 2^{25} key-stream bytes. The average case data complexity is therefore, $2^{24.5}$ key-stream bytes.

V. State Recovery Algorithm Based on FLAs for F-FCSR-H v2 and F-FCSR-H

In this section, we present a state recovery algorithm based on FLAs for the F-FCSR-H v2 and the F-FCSR-H stream ciphers. As shown in [2], during the linearisation interval of length n (say), $n+\lfloor \log_2(\omega) \rfloor$ cells on both edges of the FCSR are correlated. This enables us to determine exactly, n cells on both edges of the main register. The FLAs make use of this observation and are applicable against the FCSR-based ciphers in general, and particularly against those ciphers, which use linear Boolean functions to filter the main register states of the FCSR to produce the key-stream.

Algorithm 2. State recovery algorithm based on FLAs for F-FCSR-H v2 and F-FCSR-H

Input 1) F and known values of the main register cells

2) $z_0, z_1, \ldots, z_N$

Output $p(t)$

Set $T \leftarrow 0, n \leftarrow 0, m_0 \leftarrow 0$

$m_i \leftarrow 1 \quad \text{for} \quad 1 \le i \le 15$

Off-line a) formulate 1 x 20 matrices

$$M_j = \{m_i \mid 0 \le i \le 159 \wedge i \equiv j (\text{mod}\, 8)\}$$

b) compute $P_0, P_1, \ldots, P_7$

On-line

1. While $(n < N)$ do

1.1) take $z(n+m) \quad \text{such that} \quad 0 \le m \le 17$

1.2) formulate 18x1 matrices $W_0, W_1, \ldots, W_7$

1.3) formulate $M_i \cdot P_i = W_i \quad \forall i \in [0,7]$

1.4) solve the systems of equations (step-1.3) to recover

m_i for $16 \le i \le 159$

1.5) load the recovered values in M and C

1.6) generate KSB_i for $i = 0,1,2$

1.7) if $z(i+n) == KSB_i \quad \forall i \in \{0,1,2\}$

1.7.1) $T \leftarrow 1$

1.7.2) goto 4

1.8) $n \leftarrow n+1$

2. if $T == 1$ then return $p(t)$

else return no solution.

Algorithm 2 is based on the technique of FLAs and recovers the internal states of the F-FCSR-H v2 and the F-FCSR-H stream ciphers. In the initialisation step of Algorithm 2, we inject the known values of some of the binary cells of the internal state into the algorithm. The Off-line phase of the algorithm can be considered as a pre-processing step. In this step we divide the main register states into 8 disjoint sets according to the filtering mechanism. The P-matrices are computed by simulating the behavior of the filter for 18 contiguous clock cycles assuming the generator functions in the linearisation interval during these 18 contiguous clock cycles. Note that each of these matrices is of dimension 18x20.

The step-1 of online phase of Algorithm 2 is considered as computationally most expensive step of the attack. In this step, we take 18 contiguous bytes of the key-stream, which are split into eight 18x1 matrices $W_0, W_1, \ldots, W_7$. Each of these matrices is used to formulate eight systems of equations, which combine unknown internal state with the known key-stream values. Each such system consists of 18 equations in 20 variables. Using the known values of the main register cells, we can reduce each of these systems in to a system consisting of 18 equations in 18 variables.

These systems of equations are consistent, if the key-stream bytes belong to the linearisation interval of the generator. The solutions (if any) of these systems of equations give the values of the internal state variables m_i for $16 \le i \le 159$. These recovered values are then used to regenerate three key-stream bytes and compared with the known values. If these re-generated values match the known values, then the Boolean variable T is set to true.

Theorem 1 shows that the maximum known key-stream required for Algorithm 2 to succeed is equal to the key-stream generated during $2^{\lfloor \log_2(82) \rfloor + 17} = 2^{23}$ advances of the generator. The integer 17 in this expression denotes the duration of the linearisation interval. During these 18 advances of the generator in the linearisation interval, the generator produces 18 bytes of key-stream, which are used to formulate eight 18x20 systems of equations. Since, the F-FCSR-H generator produces one byte of key-stream during each clock cycle, therefore, the worst-case data complexity of this algorithm is 2^{23} key-stream bytes and the average case data complexity is $2^{22.5}$ key-stream bytes. The worst -case running time complexity of Algorithm 2 is 2^{23} primitive operations and the average case running time complexity of Algorithm 2 is $2^{22.5}$ primitive operations.

VI. State Recovery Algorithm Based on ILAs for F-FCSR-H v2 and F-FCSR-H

In this section, we present a state recovery algorithm based on ILAs to recover the internal states of F-FCSR-H v2 and the F-FCSR-H stream ciphers. These attacks combine the correlations of $n+\lfloor \log_2(\omega) \rfloor$ cells on both edges of the FCSR with the guess-and-determine

Copyright © 2012, Infonomics Society

approach to improve the data and memory complexities of the attack. Additional calculations required in this attack to verify the guessed bits increase the running time complexity of the attack. Therefore, there is a tradeoff between the data and the running time complexities of the attack.

Algorithm 3. State recovery algorithm based on ILAs for F-FCSR-H v2 and F-FCSR-H

Input 1) F ; known and guessed values of the main register cells

2) $z_0, z_1, \ldots, z_N$

Output $p(t)$

Off-line a) formulate 1 x 20 matrices

$$M_j = \{m_i \mid 0 \le i \le 159 \wedge i \equiv j (\mathrm{mod}\, 8)\}$$

b) compute $P_0, P_1, \ldots, P_7$

Set $c \leftarrow 0$

On-line

1. a) $T \leftarrow 0, n \leftarrow 0, m_0 \leftarrow 0$

   b) $m_i \leftarrow 1$ for $1 \le i \le 15$

   c) $m_i \leftarrow 0$ for $154 \le i \le 159$

   d) $m_{16} \leftarrow g_1, m_{154} \leftarrow g_2$

2. While $(n < N)$ do

   2.1) take $z(n+m)$ such that $0 \le m \le 16$

   2.2) formulate 17x1 matrices $W_0, W_1, \ldots, W_7$

   2.3) formulate $M_i \cdot P_i = W_i \quad \forall i \in [0,7]$

   2.4) solve the systems of equations (step-2.3).

   2.5) load the recovered values in M and C

   2.6) generate KSB_i for $i = 0,1,2$

   2.7) if $z(i+n) == KSB_i \quad \forall i \in \{0,1,2\}$

   3.7.1) $T \leftarrow 1$

   3.7.2) goto 4

   2.8) $n \leftarrow n+1$

3. if $T == 1$ then return $p(t)$

   else return no solution.

Algorithm 3 shows the applications of ILAs on the F-FCSR-H v2 and the F-FCSR-H stream ciphers. This algorithm requires as input, the known values of main register cells as explained above. This additional information about the internal state of the FCSR-based ciphers, results in improving the data and memory complexities of the attack. In addition, this algorithm also uses guess-and-determine approach for two binary cells of the main register, which increases the running time complexity of the attack. Therefore, there is a slight trade-off between the data and running time complexities of this attack. This algorithm requires eight 17x20 P-matrices, which are denoted by $P_0, P_1, \ldots, P_7$. As in the case of CLAs and the FLAs, these P-matrices are computed by simulating the behavior of the Filter F under the assumption that the generator functions in the linearisation interval for 17 contiguous clock cycles.

To estimate various complexity measures of Algorithm 3, we note that the online phase of this algorithm also includes the initialisation step. The guess-and-determine approach used in this step of the algorithm, increase the maximum number of iterations in the online phase of the attack. Theorem 1 shows that the maximum known key-stream required for Algorithm 3 to recover the internal states of the F-FCSR-H v2 and the F-FCSR-H stream ciphers is equal to the key-stream generated during $2^{\lfloor \log_2(82) \rfloor + 16} = 2^{22}$ advances of the generator. During this period, the key-stream generator functions in the linearisation interval for at least 17 contiguous clock cycles of the generator. These 17 advances of the generator, during the linearisation interval produce 17 contiguous advances of the key-stream, which are used to formulate eight systems of equations, where each such system relates the known key-stream with the unknown internal state variables. Therefore, the worst-case data complexity of the attack is 2^{22} key-stream bytes, and the average case data complexity is $2^{21.5}$ key-stream bytes.

VII. COMPARATIVE ANALYSIS

This section is devoted to present a comparative analysis of results of CLAs, FLAs and ILAs on the F-FCSR-H v2 and the F-FCSR-H key-stream generators as described in the preceding sections. In the three state recovery algorithms presented in the preceding sections, computationally most expensive step is the on-line phase. In this phase of these state recovery algorithms, we formulate and solve the systems of equations, which relate internal unknown state variables with the known values of the key-stream. Therefore, analysis of the on-line phase of the algorithms determines various performance metrics of the attack.

Table 1. Comparison of results of CLAs, FLAs and ILAs to recover the internal states of F-FCSR-H v2 and the F-FCSR-H

	Data Complexity	Time Complexity	Memory Complexity
CLAs	$2^{24.5}$	$\approx 2^{31.85}$	$\approx 2^{27.31}$
FLAs	$2^{22.5}$	$\approx 2^{29.67}$	$\approx 2^{25.17}$
ILAs	$2^{21.5}$	$\approx 2^{30.08}$	$\approx 2^{24.09}$

In Table 1, we present such an analysis, which compares the three different complexity measures, i.e. required length of known key-stream, number of equations to be solved and the number of solutions to be stored for the three types of linearisation attacks.

The results presented in Table 1, show that ILA is the most efficient attack in terms of data and memory

Copyright © 2012, Infonomics Society

complexities. These results also show that FLA is the most efficient attack in terms of time complexity. CLAs are easy to implement in software as compared to FLAs and the ILAs. In summary, this analysis shows that these variants provide a trade-off between various complexity measures, when compared with each other.

VIII. KEY RECOVERY

In this section, we present a key recovery algorithm, which recovers the effective key used in the F-FCSR-H v2 and the F-FCSR-H stream ciphers. The key-recovery algorithm works in conjunction with the state recovery algorithms and recovers the effective key (See Definition 8.1) used in these ciphers.

Definition 8.1 An effective key for an FCSR-based cipher, which filters the main register states of a Galois FCSR by using a linear Boolean function, is defined as the FCSR state, which generates the first key-stream unit, which may or may not be used for encryption or decryption.

Algorithm 4. FCSR key recovery algorithm

Input $N \in \square$, q, $p(N)$

Output $p(0)$ which corresponds to $(Unit)_0$

Set $b \leftarrow q, b_r \leftarrow 0$

Off-line
a) $b = b_{r-1} \cdots b_2 b_1 b_0$
b) $B = b_r b_{r-1} \cdots b_2 b_1 b_0$
c) Represent $p(N)$ as bit-string of length equal to B, concatenate as many zeros on the left as needed

On-line
1. While($N > 0$) do
   1.1 $p \leftarrow p(N) << 1$
   1.2 If ($B <_l p$)
      1.2.1 $p \leftarrow p - B$
   1.3 $N \leftarrow N - 1$
   1.4 $p(N) \leftarrow p$
2. Return p

Let $(Unit)_0$ denote the first key-stream unit generated by the F-FCSR-H generator by using the effective key. Algorithm 4 works in an iterative manner to recover the effective key. This algorithm requires as input, the FCSR state $p(t)$ (say) at time t, and gives as output, the FCSR state $p(0)$, which corresponds to the key-stream $(Unit)_0$ for these ciphers. This state $p(0)$ is known as effective key. The inputs required for the three state recovery algorithms at time t, are the key-stream bytes generated by the states $p(t), p(t+1), \ldots, p(t+n)$, where $n = 19,17,16$ respectively, for algorithms based on CLAs, FLAs and ILAs. On successful termination, these algorithms return the FCSR state $p(t)$. This state $p(t)$ can be considered as the current internal state of the cipher under consideration, and is used to decrypt the cipher-text from this point onward. As a cryptanalyst, we are mainly interested to retrieve the entire plaintext message or as much as possible. The key recovery algorithm is aimed to satisfy this requirement.

Example 8.1 Consider the Key/IV setup procedure of the F-FCSR-H v2 [5], which is given below.

- The carry register C is initialised with 0's. The input key K of size 80 bits and the IV of size $v \leq 80$ bits are used to initialize the main register M as $M := 0^{80-v} \,||\, \text{IV} \,||\, K$
- The key-stream generator is clocked 20 times and using the filter 20 pseudo-random bytes $S_0, S_1, \ldots, S_{19}$ are generated.
- The main register is re-initialised with $S_0, S_1, \ldots, S_{19}$ and the carry register is re-initialised with 0's.
- The FCSR is then clocked 162 times before using the output for encryption or decryption.

In this case, the effective key is the FCSR state in Step-3 of the Key/IV setup procedure.

Algorithm 4 is an iterative procedure, which reverses the direction of the generator to recover the effective key used in the cipher. This algorithm uses the current state and generates previous states up to the initial point and recovers the effective key used by the cipher. In this algorithm, $p(N) << 1$ denotes the left shift of $p(N)$ by 1, inserting a 0 on the right and discarding the overflow bit on the left. The parameters B and p represent bit strings of equal lengths. Suppose, the state recovery algorithm recovers the N^{th} internal state of the cipher, then the key recovery algorithm recovers the effective key by performing $N + 162$ iterations in the 'while loop' of the algorithm.

F-FCSR-H is similar to F-FCSR-H v2 and uses the same filtered FCSR to generate the key-stream. These two constructions differ in only the Key/IV setup procedure. The Key/IV setup procedure for F-FCSR-H [5] is given below.

- The carry register C is initialized with 0's and the main register M is initialised with the key and the IV as: $M := 0^{80-v} \,||\, \text{IV} \,||\, K$
- The generator is clocked 160 times before using the output.

In this case, the secret and the effective keys are the same. This enables us to recover the secret key used in the F-FCSR-H generator by using the key recovery algorithm. Assume that the state recovery algorithm recovers the

Copyright © 2012, Infonomics Society

N^{th} internal state of the cipher, then the key recovery algorithm recovers the secret key by performing $N+160$ iterations in the 'while loop' of the on-line phase of the algorithm.

IX. CONCLUSION

In this paper, we presented a new class of cryptanalytic attacks, which are applicable against those ciphers whose analysis theory depends on the properties of 2-adic numbers. The new class of cryptanalytic attacks is referred to as 'Linearisation Attacks'. This class consists of three variants, namely, 'Conventional Linearisation Attacks', 'Fast Linearisation Attacks' and the 'Improved Linearisation Attacks'. All these variants provide a tradeoff between data, time and memory complexities when compared with each other. These attacks are demonstrated against the F-FCSR family of stream ciphers by cryptanalysing two important members of this family, which are known as F-FCSR-H v2 and the F-FCSR-H. To cryptanalyse these ciphers, we presented state recovery algorithms based on the three different variants of the linearisation attacks. These algorithms recover the internal secret state of the ciphers. A comparative analysis of different attacks on F-FCSR-H v2 and F-FCSR-H stream ciphers is also presented, which shows various tradeoffs between these variants in terms of data, time and memory complexities. This analysis shows that CLA is easy to implement, but requires more resources such as known key-stream, processing time, etc. The FLA is the most efficient attack in terms of running time complexity. The ILA is the most efficient attack in terms of data and memory complexities. The paper also presented a key recovery algorithm, which works in conjunction with the state recovery algorithms and recovers the effective key used in the ciphers.

X. REFERENCE

[1] A.Ali, "New Most Efficient State Recovery Attacks on an eSTREAM Candidate F-FCSR-H v2 and F-FCSR-H Stream Ciphers," In Proceedings of the World Congress on Internet Security (WorldCIS-2011), London, UK, 2011.

[2] A. Ali, "New Results on FCSRs and Their Applications for FCSR-based Stream Ciphers," PhD Annual Review Report, RHUL, UK, 28 October 2010.

[3] F. Arnault and T. Berger, "Design and Properties of a New Pseudo-random Generator Based on a Filtered FCSR Automaton," IEEE Transactions on computers 54(11), 2005, pp.1374–1384.

[4] F. Arnault and T. Berger, "F-FCSR: Design of a new class of stream ciphers," In: H. Gilbert and Fast Software Encryption 2005, Lecture Notes in Computer Science, 3557, 2005, pp. 83-97.

[5] F. Arnault, T. Berger, and C. Lauradoux, "Update on F-FCSR stream cipher," eSTREAM, ECRYPT stream cipher project, Report 2006. http://www.ecrypt.eu.org/stream/, 2006.

[6] J. Evertse, "Linear structures in block ciphers," Advances in Cryptology – EUROCRYPT'87, Lecture Notes in Computer Science 304, 1988.

[7] M. Goresky and A. Klapper, "Periodicity and Distribution Properties of Combined FCSR Sequences," Sequences and Their Applications – SETA'06, Lecture Notes in Computer Science 4086, 2006, pp. 334-341.

[8] M. Hell and T. Johansson, "Breaking the F-FCSR-H Stream Cipher in Real Time," Proceedings of 14th International Conference on the Theory and Applications of Cryptology and Information Security, Advances in Cryptology, Lecture Notes in Computer Science 5350, 2008, pp. 557-569.

[9] M. Hell and T. Johansson, "Breaking the Stream Ciphers F-FCSR-H and F-FCSR-16 in Real Time," Journal of Cryptology, 2009, pp. 1-19.

[10] A. Kerkhoffs, "La cryptographie militaire," Journal des Sciences Militaires, 1883, pp. 161-191.

[11] A. Klapper and M. Goresky, "Fibonacci and Galois Representations of Feedback –With-Carry Shift Registers," IEEE Transactions on Information Theory 48(11), 2002, pp. 2826-2836.

[12] M. Saarinen, "Linearisation attacks against syndrome based hashes," In: K. Srinathan, C. Pandu, M. Yung (eds.), Indocrypt'07, Lecture Notes in Computer Science 4859, 2007, pp. 1-9.

[13] S. Babbage, C.D. Canniere, A. Canteaut, C. Cid, H. Gilbert, T. Johansson, M. Parker, B. Preneel, V. Rijmen, and M. Robshaw, "The eSTREAM Portfolio", April 15, 2008, Available at: http://www.ecrypt.eu.org/stream/portfolio.pdf (Access Date: March 7, 2011).

[14] S. Babbage, C.D. Canniere, A. Canteaut, C. Cid, H. Gilbert, T. Johansson, M. Parker, B. Preneel, V. Rijmen, and M. Robshaw, "The eSTREAM Portfolio (rev. 1)", September 8, 2008, Available at: http://www.ecrypt.eu.org/stream/portfolio-revision1.pdf (Access Date: March 7, 2011).

Copyright © 2012, Infonomics Society