Scan-based Attacks on Linear Feedback Shift Register Based Stream Ciphers

Yu Liu, Kaijie Wu, and Ramesh Karri

***Abstract*—In this paper, we present an attack on stream cipher implementations by determining the scan chain structure of the linear feedback shift registers in their implementations. Although scan Design-for-Test (DFT) is a powerful testing scheme, we show that it can be used to retrieve the information stored in a crypto chip thus compromising its theoretically proven security.**

***Index Terms*—Stream Ciphers, LFSRs, Scan DFT, Side-channel attack**

I. INTRODUCTION

Stream ciphers are an important class of encryption algorithms. They encrypt individual characters of a plaintext message one bit at a time. In contrast block ciphers operate on large blocks of data. Consequently, stream ciphers have simple hardware circuitry, are generally faster and consume very low power. Stream ciphers are deployed in applications where buffering is limited or characters are processed individually such as in wireless telecommunications applications. Stream ciphers have limited or no error propagation and hence are advantageous in noisy environments where transmission errors are highly probable. Stream ciphers are being widely implemented in Radio Frequency Identification (RFID) tags. RFID tags are made up of a microchip with some data storage and an antenna. Tag readers broadcast an RF signal to access information stored on the tags. RFIDs are an important cross-section technology whose potential application can be found in practically all areas of daily life and business.

Scan-based attacks exploit the scan chains that are inserted into devices for the purpose of test. Until now, scan attacks have been demonstrated on DES and AES block ciphers [9][13]. By loading pairs of known plaintexts that are different in a single bit position in the normal mode and then scanning out the internal state in the test mode, the positions of all scan elements in the scan chain can be determined. Then, based on a systematic analysis of the modules in the block cipher the secret key is easily discovered.

Countermeasures against scan-based attacks have also been proposed. These include secure scan [13], scan chain scrambling [1] and lock and key techniques [2],[3]. The secure scan architecture ensures a reset/clear to all the register bits in a scan chain when the device switches from the secure mode to the non-secure mode. The scrambling technique randomizes the order of bits in a scan chain and only the authorized tester knows the secret order. The lock and key techniques implement key checking logic into the chip. Upon detecting a wrong test key, the internal states of the chip are scrambled. However, since the secret order of scan flip flops of [1] or the key checking logic of [2] and [3] is common to all chips produced in a batch, maintaining the privacy of these secrets becomes an additional security concern for mass-produced products.

In this paper we will propose a scan-based attack on LFSR-based stream ciphers. The improved attack does NOT require the attacker to scan in any vectors, nor provide any input to the design as required by the scan-attacks on block ciphers. We will introduce the general technique to determine the scan chain structure of several types of LFSRs and follow it up with demonstrating this attack on six LFSR-based stream ciphers DECIM [4], Pomaranch [5], A5/1, A5/2 [6], w7 [7], and LILI II [8].

II. GENERAL DESCRIPTION OF THE ATTACK

We assume that the attacker

- knows the Cipher-Under-Attack (CUA) since all stream ciphers discussed in this paper are public;
- can run the Device-Under-Attack (DUA) for a certain

number of clock cycles;

• can scan out the states of internal registers of DUA via scan chains after each clock cycle;

• does NOT scan in vectors and does not apply chosen inputs to the DUA.

The last assumption makes the proposed attack different and more powerful than the one proposed in [9]. After each scan out operation, the attacker will obtain a bit vector that includes all bits of the LFSR and all bits of the architectural registers. We define architectural registers as those that are not in the CUA specification but are in the DUA implementation. Since LFSRs are initialized by the secret key and an initial vector, a stream-cipher-based DUA can be reproduced if the initial states of all the LFSRs are recovered even though the actual secret key itself may not be known. The goal of the attacker is to discover the correspondence between the bits of the *N*-bit scan-out vector and the bits in the LFSRs in the stream cipher.

The attacker will scan out the internal registers at the time when the DUA is initialized and records the scan-out vector V_0. He then clocks the DUA by one cycle and records the scan-out vector as V_1. In this manner, the attacker repeats this procedure for a certain number of rounds for the DUA and uses all the recorded vectors to reconstruct the state information of the DUA.

III. SCAN ATTACK ON LFSRS

In the following subsections we will describe several attacks that target simple but general LFSR structures. The attack on a specific CUA is a combination of some or all of these attacks. We will analyze the case where the scan-out vector consists of the bits from an LFSR and architectural registers. The states of the architectural registers are assumed to be random. Let N denote the length of the scan-out vector and L denote the size of the LFSR, $N \geq L$. The size of the architectural registers is then N-L.

A. Scan Attack on External (Fibonacci) LFSRs

Figure 1 shows two L-bit external LFSRs. One has no input and the other has one. Since the attack on both are the same, we will only illustrate the attack on the former.

The bits of an external LFSR without an input have the following relations:

$$S_i(t) = S_{i-1}(t\text{-}1) \quad \text{for } 1 \leq i \leq L\text{-}1 \qquad (1)$$

$$S_0(t) = \sum_{0 \leq i \leq L-1} (C_i \times S_{L-1-i}(t\text{-}1)),\ C_i = 0 \text{ or } 1 \qquad (2)$$

$S_i(t)$ is the state of i^{th} stage at clock cycle t (scanned out as part of the vector V_t) and $S_{i-1}(t\text{-}1)$ is the state of $(i\text{-}1)^{th}$ stage at cycle $t\text{-}1$ (scanned out as part of the vector V_{t-1}). C_i $(1 \leq i \leq L\text{-}1)$ could be 1 or 0 depending on the characteristic polynomial of the LFSR.

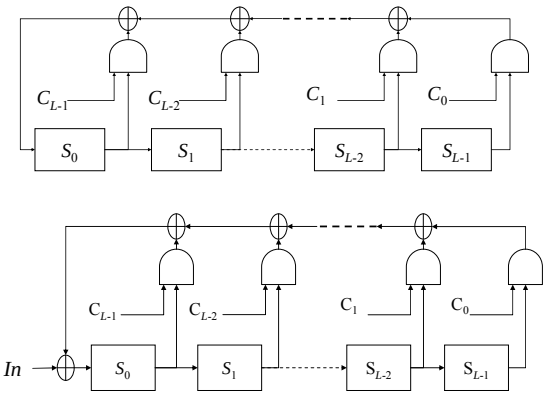

Figure 1: An L-bit external LFSR (a) without an input and (b) with an input

To discover the bit-by-bit correspondence between the scan-out vectors and the LFSR, the attacker randomly picks a bit X from one of the N-bit scan-out vectors, and performs an α-search defined below, to discover if X belongs to the LFSR:

α-leftward-search: For a given bit X , this search looks for another bit W where $W(t\text{-}1)=X(t)$.

α-rightward-search: For a given bit X , this search looks for another bit Y where $X(t)=Y(t+1)$.

(a)

Vector	1	2	3	4	5	6	7	8	9	10	Suspect set of W
V_0	0	0	0	0	0	0	0	1	1	0	All bits except 9
V_1	1	0	0	0	0	0	0	0	0	0	1, 2, 3, 4, 5, 6, 7, 10
V_2	0	1	0	0	0	0	0	0	0	1	2, 3, 4, 5, 6, 7, 10
V_3	0	0	1	0	0	0	0	0	1	1	2, 10
V_4	1	0	0	1	0	0	0	0	0	0	2
V_5	0	1	0	0	1	0	0	0	0	1	2
V_6	0	0	1	0	0	1	0	0	1	0	2
V_7	1	0	0	1	0	0	1	0	1	1	Miss

(b)

Vector	1	2	3	4	5	6	7	8	9	10	Suspect set of W
V_0	0	0	0	0	0	0	0	1	1	0	All bits except 8
V_1	1	0	0	0	0	0	0	0	0	0	1, 2, 3, 4, 5, 6, 7, 10
V_2	0	1	0	0	0	0	0	0	0	1	2, 3, 4, 5, 6, 7, 10
V_3	0	0	1	0	0	0	0	0	1	1	3, 4, 5, 6, 7
V_4	1	0	0	1	0	0	0	0	0	0	4, 5, 6, 7
V_5	0	1	0	0	1	0	0	0	0	1	5, 6, 7
V_6	0	0	1	0	0	1	0	0	1	0	6, 7
V_7	1	0	0	1	0	0	1	0	1	1	7

Figure 2 A leftward-search returns (a) a miss or (b) a hit

Let's consider an 8-bit external LFSR with feedback polynomial $1+x^3+x^8$. To remove any ambiguity we also give the corresponding update function of the LFSR as $S_0(t) = S_2(t\text{-}1)+ S_7(t\text{-}1)$. For simplicity, we assume that the bits of the scan-out vectors are in the same sequence as the bits in the LFSR, i.e. the 1st bit is S_0, the 8th bit is S_7, and the 9th and 10th bits are architectural registers. The α-leftward-search on this example is shown in Figure 2. Figure 2(a) assumes that the 9th bit of the scan-out vectors is chosen to be X. The attacker finds its left neighbor W by pruning the "suspect set" of bits which initially includes all bits except X. The attacker prunes this set by eliminating those bits whose values in V_i are different from the bit X in V_{i+1}. **This is one round of checking**. Since the bits in the LFSR are pseudorandom and the bits in the architectural registers are assumed to be random, each bit in the scan-out vector has about 50% chance to be 1 or 0. The probability of eliminating a bit from the suspect set at the n^{th} round equals

$$\prod_{1 \le t \le n\text{-}1} Pb_{W(t\text{-}1)=X(t)} \times Pb_{W(n\text{-}1)\neq X(n)} = 0.5^n$$

where $Pb_{W(t\text{-}1)=X(t)}$ stands for the probability of $W(t\text{-}1)=X(t)$.

On average a bit can be eliminated from the suspect set in 2 rounds (average number of rounds, denoted as ANR thereafter), and more than 99.99% bits can be eliminated from the suspect set in 15 rounds (the maximum number of rounds, denoted as MNR thereafter). A bit that survives MNR rounds of checking is the W bit with very-close-to-1 probability, and the search is said to return a **hit**, as shown in Figure 2 (b). If the search returns an empty suspect set (a miss), the attacker randomly picks another bit and repeats this procedure until he gets a hit.

A hit discovers two bits, X and W, out of the LFSR. W is the left neighbor of X. A discovered bit is called the Left Boundary Bit (LBB) if its left neighbor is undiscovered. Similarly, a discovered bit is the Right Boundary Bit (RBB) if its right neighbor is undiscovered. In this example, W is an LBB and X is an RBB. Repeatedly applying the α-leftward-search on LBB grows the discovered bits until we discover the left-most bit S_0 of the LFSR. Similarly, repeatedly applying the α-rightward-search on an RBB will discover all the bits until the right most bit S_{L-1}. When all bits are discovered, the structure of the LFSR is automatically identified. It is important to note that the scan-out vectors used to discover the first bits can be reused to discover the rest of bits. Therefore for each DUA the attacker only needs to scan out MNR+1 vectors.

B. Scan Attack on Internal (Galois) LFSRs

A general structure of an internal LFSR is shown in Figure 3. The bits in an internal LFSR have the following relations:

$$S_i(t) = S_{i-1}(t\text{-}1) \oplus (C_i \cdot S_{L-1}(t\text{-}1)) \quad (1 \le i \le L\text{-}1) \qquad (3)$$

$$S_0(t) = S_{L-1}(t\text{-}1) \qquad (4)$$

C_i $(1 \le i \le L\text{-}1)$ could be 1 or 0 depending on the characteristic polynomial implemented by the LFSR.

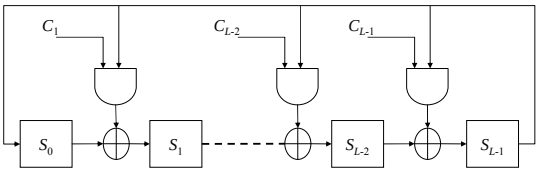

Figure 3 A L-bit internal LFSR

When $C_i = 1$, $S_i(t) = S_{i-1}(t\text{-}1) \oplus S_{L-1}(t\text{-}1)$, which is referred to as a tap bit. When $C_i = 0$, there is no feedback involved and $S_i(t) = S_{i-1}(t\text{-}1)$ $(1 \le i \le L\text{-}1)$, which is referred to as a non-tap bit. The non-tap bits can be discovered by the α-search described above. Discovering and identifying the tap-bits need a new type of search, called β-search:

β-leftward-search: For a selected bit X, this search looks for a 2-tuple (W, Z) where $W(t\text{-}1)=X(t)$ when $Z(t\text{-}1) = 0$, or $W(t\text{-}1)=X(t)$' when $Z(t\text{-}1) = 1$.

β-rightward-search: For a selected bit X, this search looks for a 2-tuple (Y, Z) where $X(t)=Y(t+1)$ when $Z(t) = 0$, or $X(t)=Y(t+1)$' when $Z(t) = 1$.

The β-search is based on the observation that $S_i(t)=S_{i-1}(t\text{-}1)$ or $S_i(t)=S_{i-1}(t\text{-}1)$' when $S_{L-1}(t\text{-}1) = 0$ or 1 respectively. For the first β-search, the attacker has to guess two bits, the neighbor bit of X and S_{L-1}. The number of possible 2-tuples for a given bit X is $P(U, 2)$, where U is the number of undiscovered bits. However, after the first β-search returns a hit, bit S_{L-1} is identified. To discover the remaining bits using β-searches, the

attacker needs to guess only one bit. This greatly reduces the suspect set of 2-tuples to $P(U, 1)$. The probability of eliminating a 2-tuple from the suspect set in the n^{th} round equals:

$$\prod_{1 \le t \le n-1} (Pb_{Z(t-1)=1} \times Pb_{W(t-1)=X(t)'} + Pb_{Z(t-1)=0} \times Pb_{W(t-1)=X(t)}) \times [Pb_{W(n-1) \neq X(n)'} \times Pb_{Z(n-1)=1} + Pb_{W(n-1) \neq X(n)} \times Pb_{Z(n-1)=0}] = 0.5^n.$$

On average, a 2-tuple can be eliminated from the suspect set in 2 rounds (ANR for this search) and more than 99.99% of false 2-tuples will be eliminated from the suspect set in 15 rounds (MNR for this search). The attack procedure combining the α and β searches is as follows:

1) The attacker randomly picks a bit and applies the α-leftward-search. If the search returns a miss, the attacker randomly picks another bit and continues with α-leftward-searches until he gets a hit. The hit discovers a non-tap bit and its left neighbor which are the RBB and LBB of the discovered bits respectively.

2) The attacker applies the α-leftward-search on LBB and the α-rightward-search on RBB. This step is repeated to grow the discovered section until either the whole LFSR is discovered (LBB = RBB), or both the leftward and the rightward searches return misses. The latter case indicates that the attacker has reached tap-bits.

3) The attacker applies the β-leftward-search on the LBB and the β-rightward-search on the RBB to locate the left neighbor of LBB and right neighbor of RBB, which will become the new LBB and RBB respectively. Then go to step 2.

Steps 2) and 3) are repeated until all the bits in the LFSR are discovered. Since the β-search will identify bit S_{L-1}, this bit can be used to identify all the other tap bits in this LFSR.

C. Scan Attack on Internal LFSRs with Inputs

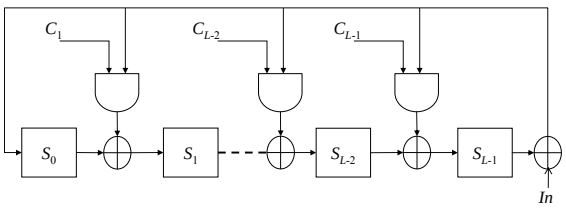

Figure 4 An internal LFSR of length L with an input

An internal LFSR with input is shown in Figure 4. The bits have the following relations where *In* stands for the input:

$$S_i(t) = S_{i-1}(t-1) \oplus (C_i \cdot (S_{L-1}(t-1) \oplus In)),\ 1 \le i \le L-1 \quad (5)$$

$$S_0(t) = S_{L-1}(t-1) \oplus In \quad (6)$$

C_i ($1 \le i \le L-1$) could be 1 or 0 depending on the characteristic polynomial implemented by the LFSR. When $C_i = 0$, there is no feedback involved and $S_i(t) = S_{i-1}(t-1)$ ($1 \le i \le L-1$). These non-tap bits can be discovered by α-searches. When $C_i = 1$, $S_i(t) = S_{i-1}(t-1) \oplus S_0(t-1) \oplus In$. Determining these tap bits, however, requires extra effort since *In* is not accessible by the attacker. The attacker considers two tap bits, LBB S_i and the RBB S_j at the same time. Since $S_i(t) = S_{i-1}(t-1) \oplus S_0(t-1) \oplus In$ and $S_{j+1}(t) = S_j(t-1) \oplus S_0(t-1) \oplus In$, and $S_i(t) \oplus S_{j+1}(t) = S_{i-1}(t-1) \oplus S_0(t-1) \oplus In \oplus S_j(t-1) \oplus S_0(t-1) \oplus In = S_{i-1}(t-1) \oplus S_j(t-1)$

A new γ-search for such a discovery may be defined as follows:

γ-search: Given a pair of bits X and F, this search looks for a 2-tuple (W, G) where $W(t-1) \oplus F(t-1) = X(t) \oplus G(t)$. The attacker has to guess two bits W and G, therefore the suspect set of 2-tuples for bits X and F is $P(U, 2)$ where U is the number of undiscovered bits. The probability of eliminating a 2-tuple from the suspect set at the n^{th} round is:

$$(\prod_{1 \le t \le n-1} Pb_{W(t-1) \oplus F(t-1) = X(t) \oplus G(t)}) \times Pb_{W(n-1) \oplus F(n-1) \neq X(n) \oplus G(n)} = 0.5^n$$

On average, a 2-tuple can be eliminated from the suspect set in 2 rounds (ANR for this search) and more than 99.99% of 2-tuples can be eliminated in 15 rounds (MNR for this search). An attack procedure combining the α and γ searches is similar to the one described earlier for combining α and β searches except for the following procedure in step 3.

1) The attacker randomly picks a bit and applies the α-leftward-search. If the search returns a miss, the attacker randomly picks another bit and continues with α-leftward-searches until he gets a hit. The hit discovers a non-tap bit and its left neighbor which are the RBB and LBB of the discovered bits respectively.

2) The attacker applies the α-leftward-search on LBB and an α-rightward-search on RBB. This step is repeated

to grow the discovered section until either the whole LFSR is discovered (LBB = RBB), or both the leftward and the rightward searches return misses. The latter case indicates that the attacker has reached tap-bits.

3) The attacker applies the γ-search on the bits LBB and RBB to locate the left neighbor of LBB and right neighbor of RBB, which will become the new LBB and RBB respectively. This γ-search always returns a hit. Go to step 2.

Steps 2) and 3) are repeated until all the bits in the LFSR are discovered.

D. Scan Attack on LFSRs with Jump Registers

A modified jump cell, has been proposed to replace the normal register cell used in LFSRs in [10]. Figure 5 shows a register cell that can work in two modes controlled by the Jump Control switch. When the switch is open, it works as a normal register cell and when the switch is closed, it works as a jump cell. Clocking an LFSR using normal cells J times produces the same result as clocking once the same LFSR that uses jump cells instead. J is the jump index derived from the characteristic polynomial.

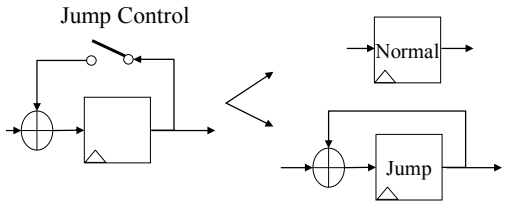

Figure 5 A Jump Cell

Attacking an LFSR using jump cells does not require any new types of searches. However, the number of rounds to eliminate a bit/tuple from the suspect set increases. The bits in an external LFSR using jump cells have the following relations where *JC* stands for Jump Control:

$$S_i(t) = S_{i-1}(t\text{-}1) \oplus JC \cdot S_i(t\text{-}1), \text{ for } 1 \le i \le L\text{-}1, \quad (7)$$

$$S_0(t) = \sum_{0 \le i \le L-1}(C_i \cdot S_{L-1-i}(t\text{-}1)) \oplus JC \cdot S_0(t\text{-}1), \; C_i = 0 \text{ or } 1 \quad (8)$$

Observe that when $S_i(t\text{-}1) = 0$, $S_i(t) = S_{i-1}(t\text{-}1)$. The jump version of α-search defined below can discover the bits in an external LFSR using jump cells:

α-leftward-search-J: Given a bit X, this search looks for another bit W where $W(t\text{-}1) = X(t)$ when $X(t\text{-}1) = 0$.

α-rightward-search-J: Given a bit X, this search looks for another bit Y where $X(t\text{-}1) = Y(t)$ when $Y(t\text{-}1) = 0$.

The probability of eliminating a bit from the suspect set at the n^{th} round is

$$\prod_{1 \le t \le n-1} (Pb_{X(t-1)=1} + Pb_{X(t-1)=0} \times Pb_{W(t-1)=X(t)}) \times (Pb_{X(n-1)=0} \times Pb_{W(n-1) \ne X(n)}) = (3/4)^{n-1} \times 1/4$$

On average it needs 4 rounds to eliminate a bit from the suspect set and more than 99.99% bits will be eliminated from the suspect set within 33 rounds (MNR). The ANR to eliminate a suspect tuple and the MNR to eliminate more than 99.99% tuples from the suspect set are all doubled. The attack procedure of the LFSRs using normal cells can also be applied to the LFSRs using jump cells.

E. Scan Attack on irregular clock controlled LFSRs

Irregularly stepping the LFSR through successive states is a method to increase the linear complexity of an LFSR while preserving a large period and good statistical properties. Stream ciphers based on regularly clocked LFSRs are susceptible to basic and fast correlation attacks [11][14]. Irregular clocking limits the possibilities for mounting classical correlation attacks.

For chips using irregular clocked LFSRs, consecutive scan-out vectors could be partly or completely the same. These redundant vectors should be abandoned. To determine if a vector is redundant, the attacker can check if the bits of interest to the search (i.e. the (X W) in an α-search, the (X W Z) in a β-search, and the (X W F G) in a γ-search) have different values from the bits in the previous vector. The ANR and MNR to eliminate the bits/tuples from a suspect set increases inversely with the probability of the LFSR being clocked in a cycle.

F. Scan Attacks on Stream Ciphers with Multiple LFSRs

Most LFSR-based stream ciphers use more than one LFSR. To determine the scan chain structure, we need to tell them apart. If the LFSRs have different lengths, we can easily tell them apart. If the LFSRs have the same length, we can still tell them apart by matching their unique feedback functions.

IV. PUTTING IT ALL TOGETHER: ATTACKS ON SELECTED STREAM CIPHERS

A. DECIM

DECIM is a stream cipher submitted to the ECRYPT eSTREAM project [11]. It uses an 80-bit key and a 64-bit IV, and a 192-bit external LFSR. The key stream generation mechanism is shown in Figure 6. The bits of the external LFSR are numbered from 0 to 191. The Boolean function f is a 13-variable quadratic symmetric function. ABSG is an irregular decimation mechanism. DECIM uses a 32-bit key buffer to maintain a constant throughput for the key stream. The LFSR is regularly clocked.

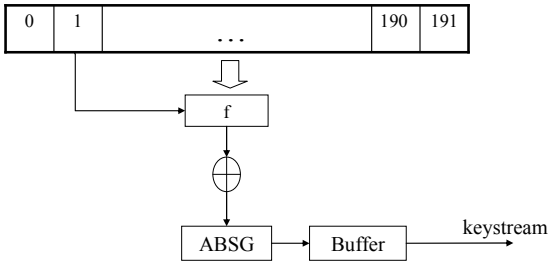

Figure 6 DECIM Stream Cipher

To attack DECIM, the α-search suffices. While we have estimated that the MNR=15 is sufficient for an α-search, we used MNR=17 in our simulation to ensure a high level of confidence. The total number of rounds of comparisons performed is around 17×N=17×228 = 3876.

B. A5/1

A5/1 is a stream cipher used for encrypting over the air transmissions in the GSM standard. A GSM conversation is transmitted as a sequence of 228-bit frames (114 bits in each direction) every 4.6 milliseconds. To ensure privacy, each frame is XORed with a 228-bit keystream produced by the A5/1. As shown in Figure 7, A5/1 cipher uses three external LFSRs – R1, R2, and R3 of lengths 19, 22, and 23 bits, respectively. At each cycle, after the initialization phase, the left-most bits of the LFSRs are XORed to produce one bit key. The three LFSRs are irregularly clocked depending on the output of a majority function M. M computes the majority of S_8 of R1, S_{10} of R2 and S_{10} of R3. An LFSR will shift only when the state of its selected bit equals M.

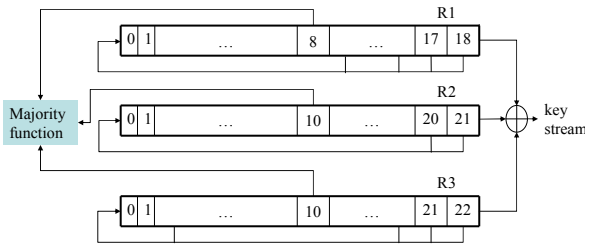

Figure 7: A5/1 Stream Cipher

To attack A5/1, we need to apply α-searches to determine the three register segments. After that, we can tell them apart by their lengths. The number of vectors used by our simulation is 32 (ie. MNR =31) and the total number of rounds of comparisons is about 31×N=31×64=1984.

C. A5/2 Stream Cipher

A5/2 is a stream cipher used to provide voice privacy in the GSM cellular telephone protocol. A5/2 is built from four external LFSRs of lengths 19, 22, 23, 17 bits denoted by R1, R2, R3 and R4 respectively, shown in Figure 8. Each LFSR is updated by its own primitive feedback polynomial. Clocking of R1, R2 and R3 is controlled by R4 and R4 is regularly clocked in each clock cycle. A majority function is attached to an LFSR and outputs the majority of three selected bits from the attached LFSR. The outputs of all the majorities and the right most bit from each register are XORed to form the output bit.

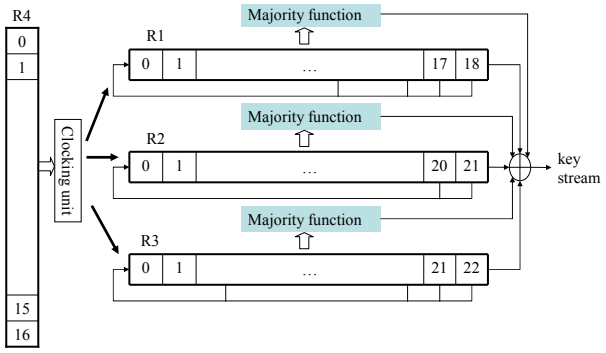

Figure 8: A5/2 Stream Cipher

The procedure to attack A5/2 is basically the same as that of A5/1. The number of scan vectors used in our simulation is 42 (i.e MNR = 41) and the total number of rounds of comparisons is approximately 41×81=3321.

D. W7

W7 is a synchronous stream cipher optimized for

efficient hardware implementation [7]. W7 cipher contains eight similar modules each of which consists of three external LFSRs and one majority function as shown in Figure 9. The majority function in a module controls the clocking of the LFSRs in the module and the clocking principle is the same as that of A5/1. The outputs of cells compose a byte of key stream.

Since all the LFSRs are external, applying α-search is sufficient to discover the bits of all the LFSRs. The identity of a LFSR can be told by matching the unique lengths and feedback functions of discovered LFSRs. The MNR used in our simulation is 83 and the total number of rounds of comparisons is around 83×N=83×1024=84992.

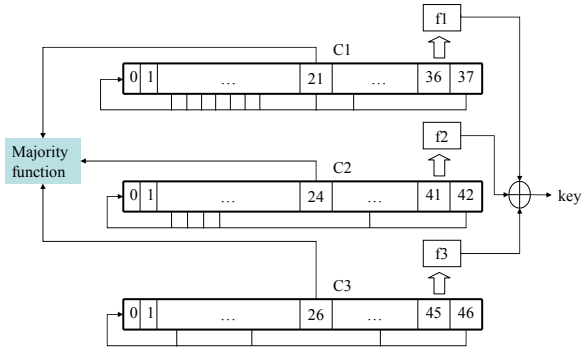

Figure 9: A module in W7 stream cipher

E. LILI II Stream Cipher

LILI-II is a simple and fast stream cipher that uses two internal LFSRs. As shown in Figure 10, there are two subsystems: one subsystem is for generating an irregular clock that controls the other subsystem that produces key stream. The LFSR in the clock-control subsystem is regularly clocked. The Fc function in the system takes S_0 and S_{126} and computes Fc = $2S_0 + S_{126} + 1$. Since the possible output of Fc could be 1, 2, 3, or 4, the LFSR used in data-generation subsystems is clocked 1, 2, 3, or 4 times respectively between two consecutive key bits.

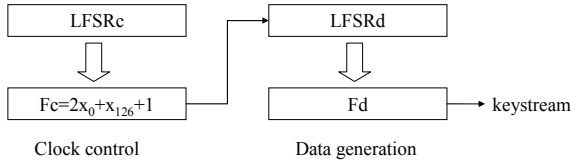

Figure 10: LILI II Stream Cipher

Since both of the two LFSRs are internal, applying α-search and β-search can discover all the bits. The identities of the two LFSRs can be told by matching the lengths. The MNR used by our simulation is 62 and the number of rounds of comparisons is about 62×N=62×255=15810.

F. Pomaranch Stream Cipher

Pomaranch is a hardware oriented stream cipher submitted to the ECRYPT eSTREAM project. The keystream generator of Pomaranch is called Cascade Jump Controlled Sequence Generator consisting of 9 modules as shown in Figure 11 (a). Each module has an 18-bit shift register using a bunch of F and S cells as shown in Figure 11 (b). An F cell works like a jump cell when JC_i of this module is 1, or like a regular shift cell when JC_i is 0. Oppositely, an S cell works like a jump cell when JC_i of this module is 0, or like a regular shift cell when JC_i is 1. All cells are regularly clocked. The inputs to the Key Map comprises of 9 bits from the LFSR and 16 bits of the secret key. The 1-bit output of the Key Map is sent to the next module as the JC_i. For the 9 modules, all the even numbered modules share a configuration of F and S cells and a feedback function, while all the odd numbered modules share another configuration of F and S cells and another feedback function.

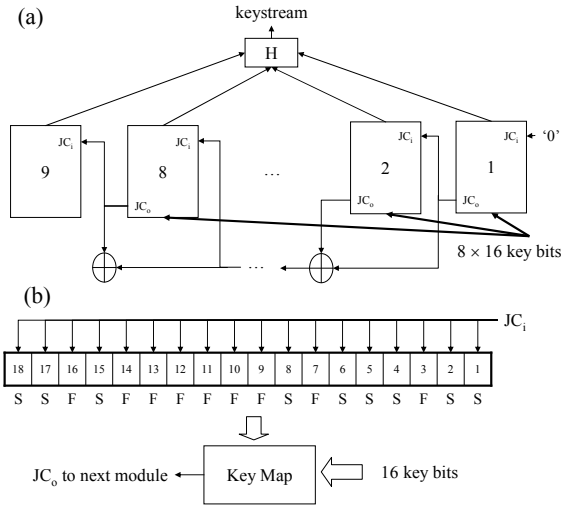

Figure 11: Pomaranch stream cipher (a) Top level (b) The odd module

The attack steps are as follows:

1) Discover the 9 LFSRs by applying the jump version of α-search;

2) Identify if a discovered LFSR belongs to an odd

module or an even module by matching its feedback function;

3) Since the JC_i of the first module is 0 always, it will match the discovered LFSR whose F and S cells never switch modes.

4) For the remaining modules, the cell working modes depend on the JC_i of the module that in turn depends on the LFSR and the 16-bit key used by the previous module. Since the LFSR of the first module is identified in step 3), we can guess JC_i by simulating every possible 16-bit key (from 0x0000 up to 0xFFFF) and see if it agrees with the cell working modes of any discovered LFSR that belongs to an even module.

5) Repeat step 4 above to attack remaining modules.

Overall, the MNR used in our simulation is 42. The total number of rounds of comparisons is about $42 \times N = 42 \times 162 = 6804$

Table 1 summarizes the simulation results of all the ciphers we have attacked. Note that the number of scan vectors needed to launch such scan-based attack is just MNR + 1. The total number of round comparisons equals MNR × N, which takes negligible time in a modern computer.

Table 1 The summary of all the ciphers

Cipher	LFSR Type	MNR	N	Clock Control
DECIM	External	17	228	Regular
A5/1	External	32	64	Irregular
A5/2	External	41	81	Irregular
W7	External	83	1024	Irregular
LILI II	Internal	62	255	Irregular
Pomaranch	Jump	42	162	Regular

V. CONCLUSIONS

In this paper we have proposed an improved scan-based attack to LFSR-based stream ciphers. The attack analyzes the scan-out vectors to discover the internal states of DUA. The number of scan-out vectors required is less than 20 for some ciphers and is less than 100 for all the ciphers attacked in this paper. The CPU time for processing the vectors and identifying the bits of LFSR is negligible. With the knowledge of the LFSRs used in stream cipher devices, an attacker can clone an authentication device, eavesdrop a private conversation, and etc.

It is reorganized that not all DFT techniques may introduce security vulnerabilities. Even a given DFT technique may not introduce security vulnerabilities in all chips and in all embedded processors. For example, in large chips and processors with over tens of thousands of flip flops, test data is typically compressed on chip. This adds an additional layer of security that prevents the attacker from recovering the bit-by-bit information of the scan chains – the attacker would have to work on compressed scan-out vectors. However, in embedded processors and crypto accelerators used in low end smart cards, test data is not compressed owing to the limited number of flip flops. With debug becoming mandatory, test inputs and test outputs are not compressed even in some large processors [15].

REFERENCES

[1] D. Hély, F. Bancel, M. L. Flottes, B. Rouzeyre, M. Renovell, and N. Bérard, “Scan Design and Secure Chip,” Proceedings of IEEE Int. On-Line Testing Symp., Funchal, 2004, pp. 219–226.

[2] J. Lee, M. Tehranipoor, C. Patel, and J. Plusquellic, “Securing Scan Design Using Lock and Key Technique,” Proceedings of 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2005, pp. 51-62.

[3] J. Lee, M. Tehranipoor, and J. Plusquellic, “A Low-Cost Solution for Protecting IPs Against Scan-Based Side-Channel Attacks,” IEEE VLSI Test Symposium, 2006, pp. 94-99.

[4] C. Berbain, O. Billet, A. Canteaut, N. Courtois, B. Debraize, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin, and H. Sibert, “DECIM,” http://www.ecrypt.eu.org/stream/ decimp3.html

[5] C.J.A. Jansen, T. Helleseth, and A. Kholosha, “Cascade jump controlled sequence generator and Pomaranch stream cipher,” http://www.ecrypt.eu.org/stream/pomaranchp3.html

[6] I. Erguler and E. Anarim, “A Modified Stream Generator for the GSM Encryption Algorithms A5/1 and A5/2,” EUSIPCO 2005

[7] S. Thomas, D. Anthony, T. Berson, and G. Gong, “The W7 Stream Cipher Algorithm,” *Internet Draft*, April 2002.

[8] A. Clark, E. Dawson, J. Fuller, J. Golic, H-J Lee, W. Millan, S-J. Moon, and L. Simpson, “The LILI-II Keystream Generator”,

Proceedings of the 7th Australian Conference on Information Security and Privacy, volume 2384 of Lecture Notes In Computer Science, Springer-Verlag, 2002, pp. 25 – 39.

[9] B. Yang, K. Wu, and R. Karri, "Secure Scan: A Design-for-Test Architecture for Crypto Chips", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), Vol. 25, No. 10, Oct 2006, pp 2287- 2293.

[10] C. Jansen, "Stream cipher design: Make your LFSRs jump!" Workshop of the State of the Art of Stream Ciphers, 2004, pp. 94–108.

[11] A. Menezes, P. van Oorschot, and S. Vanstone, Handbook of Applied Cryptography, CRC Press, ISBN 0849385237,1996

[12] eStream, Stream cipher project of the European Network of Excellence in Cryptology ECRYPT. http://www.ecrypt.eu.org/stream/

[13] Bo Yang, Kaijie Wu, and Ramesh Karri, "Secure Scan: A Design-for-Test Architecture for Crypto Chips", Design Automation Conference (DAC) 2005

[14] T. Siegenthaler, "Decrypting a class of stream ciphers using ciphertext only", IEEE Transactions on Computers, Vol. 34, No. 1, 1985, pp. 81–85.

[15] D, Josephson, S. Poehhnan, "Debug methodology for the McKinley processor," International Test Conference (ITC) 2001, pp451-460