The LILI-II Keystream Generator

A. Clark[1], E. Dawson[1], J. Fuller[1], J. Golić[2], H-J. Lee[3], W. Millan[1], S-J. Moon[4], and L. Simpson[1]

[1] Information Security Research Centre,
Queensland University of Technology,
GPO Box 2434, Brisbane, Queensland 4001, Australia
{aclark, dawson, fuller, millan, simpson }@isrc.qut.edu.au
[2] GEMPLUS, Rome CryptoDesign Center, Technology R&D,
Via Pio Emanuelli, 00413 Rome, Italy
jovan.golic@gemplus.com
[3] School of Internet Engineering, Dongseo University
San 69-1, Churye-2Dong, SaSang-Ku, Pusan 617-716, Korea
hjlee@dongseo.ac.kr
[4] School of Electronic and Electrical Engineering
Kyungpook National University, 1370,
Sankyuk-dong, Taegu 702-701, Korea.
sjmoon@knu.ac.kr

Abstract. The LILI-II keystream generator is a LFSR based synchronous stream cipher with a 128 bit key. LILI-II is a specific cipher from the LILI family of keystream generators, and was designed with larger internal components than previous ciphers in this class, in order to provide increased security. The design offers large period and linear complexity, is immune to currently known styles of attack, and is simple to implement in hardware or software. The cipher achieves a security level of 128 bits.

1 Introduction

Many keystream generator designs are based on shift registers, both for the simplicity and speed of Linear Feedback Shift Register (LFSR) implementation in hardware and for the long period and good statistical properties LFSR sequences possess. To make use of the good keystream properties while avoiding the inherent linear predictability of LFSR sequences, many constructions introduce nonlinearity, by applying a nonlinear function to the outputs of regularly clocked LFSRs or by irregular clocking of the LFSRs [17].

However, keystream generators using regularly clocked LFSRs are susceptible to correlation attacks, including fast correlation attacks, a concept first introduced in [16]. As a means of achieving immunity to these correlation attacks, keystream generators consisting of irregularly clocked LFSRs were proposed. These keystream generators are also susceptible to certain correlation attacks, such as the generalised correlation attack proposed in [8]. As correlation attacks have been successful against keystream generators based on the single design

principles of either a nonlinear function of regularly clocked LFSR sequences [19, 18] or on irregular clocking of LFSRs [8, 20], both of these approaches are combined for the LILI keystream generators, a family of keystream generators first introduced in [21].

LILI-II is a specific cipher from the LILI family of keystream generators. The development of LILI-II was motivated by the response to the LILI-128 keystream generator, included as a stream cipher candidate for NESSIE [6]. Although the design for the LILI keystream generators is conceptually simple, it produces output sequences with provable properties with respect to basic cryptographic security requirements. Hypothesised attacks on LILI-128, and the request for a re-keying proposal prompted a review of the LILI-128 parameters to ensure provable security properties could be maintained while achieving an effective key size of 128 bits. LILI-II is slightly less efficient in software than LILI-128 mainly due to the larger LFSRs and larger Boolean function which are used in the design to increase security. However, in hardware LILI-II offers the same high speed as LILI-128.

We now briefly summarise the security claims for LILI-II. Firstly, the period at around $2^{128} \cdot 2^{127} = 2^{255}$ greatly exceeds 2^{128}, and exceeds the length of any practical plaintext, rendering any attacks based on the period infeasible. Secondly, the linear complexity is conjectured to be at least 2^{175}, so that at least 2^{176} consecutive bits of known plaintext are required for the Berlekamp-Massey attack. This is an infeasible amount of text to collect. Thirdly, the $127 + 128 = 255$ bit state size renders any of the general time-memory-data tradeoff attacks infeasible. Additionally, we conjecture that the complexity of divide and conquer attacks on LILI-II exceeds 2^{128} operations, and requires a substantial amount of known keystream bits. Taken together, these results indicate that LILI-II is a secure synchronous stream cipher, in that there are no currently known attacks on the cipher which are more feasible than exhaustive key search.

2 Description of LILI-II Keystream Generator

The LILI-II keystream generator is a simple and fast keystream generator using two binary LFSRs and two functions to generate a pseudorandom binary keystream sequence. Both of the LFSRs in LILI-II use the Galois configuration rather than the more common Fibonacci style, although there is no security difference as the state cycle structures of these two LFSR styles have identical properties. This design decision is motivated by the observation that although the two LFSR styles are equally efficient in hardware, the Galois style LFSR is faster in software.

The structure of the LILI keystream generators is illustrated in Figure 1. The components of the keystream generator can be grouped into two subsystems based on the functions they perform: clock control and data generation. The LFSR for the clock-control subsystem is regularly clocked. The output of this subsystem is an integer sequence which controls the clocking of the LFSR

within the data-generation subsystem. If regularly clocked, the data-generation subsystem is a simple nonlinearly filtered LFSR [17] (nonlinear filter generator).

The state of LILI-II is defined to be the contents of the two LFSRs. The functions f_c and f_d are evaluated on the current state data, and the feedback bits are calculated. Then the LFSRs are clocked and the keystream bit is output. At initialisation the 128 bit key and a publicly known 128 bit initialisation vector are combined to form the initial values of the two shift registers. For efficiency,this initialisation process uses the LILI-II structure itself, and can also be used for re-keying. All valid keys produce different keystreams and there are no known weak keys.

The LILI keystream generators may be viewed as clock-controlled nonlinear filter generators. Such a system, with the clock control provided by a stop-and-go generator, was examined in [7]. However, the use of stop-and-go clocking produces repetition of the nonlinear filter generator output in the keystream, which may permit attacks. This system is an improvement on that proposal, as stop-and-go clocking is avoided. For LILI keystream generators, $LFSR_d$ is clocked at least once and at most d times between the production of consecutive keystream bits. For LILI-II, $d = 4$, so $LFSR_d$ is clocked at most four times between the production of consecutive keystream bits.

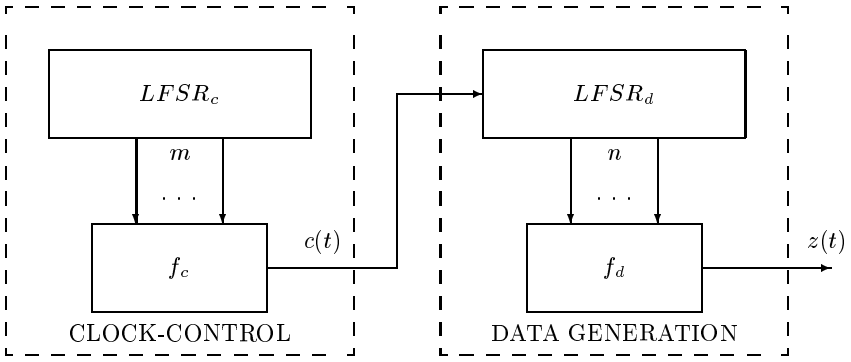

Fig. 1. Overview of LILI keystream generators.

2.1 Clock Control Subsystem

The clock-control subsystem of LILI-II uses a pseudorandom binary sequence produced by a regularly clocked LFSR, $LFSR_c$, of length 128 and a function, f_c, operating on the contents of $m = 2$ stages of $LFSR_c$ to produce a pseudorandom integer sequence, $c = \{c(t)\}_{t=1}^{\infty}$. The feedback polynomial of $LFSR_c$ is chosen to be the primitive polynomial

$$x^{128} + x^{126} + x^{125} + x^{124} + x^{123} + x^{122} + x^{119} + x^{117} + x^{115} + x^{111} + x^{108}$$
$$+x^{106} + x^{105} + x^{104} + x^{103} + x^{102} + x^{96} + x^{94} + x^{90} + x^{87} + x^{82} + x^{81}$$
$$+x^{80} + x^{79} + x^{77} + x^{74} + x^{73} + x^{72} + x^{71} + x^{70} + x^{67} + x^{66} + x^{65} + x^{61}$$
$$+x^{60} + x^{58} + x^{57} + x^{56} + x^{55} + x^{53} + x^{52} + x^{51} + x^{50} + x^{49} + x^{47} + x^{44}$$
$$+x^{43} + x^{40} + x^{39} + x^{36} + x^{35} + x^{30} + x^{29} + x^{25} + x^{23} + x^{18} + x^{17} + x^{16}$$
$$+x^{15} + x^{14} + x^{11} + x^{9} + x^{8} + x^{7} + x^{6} + x^{1} + 1$$

and the initial state of $LFSR_c$ can be any state except the all zero state. It follows that $LFSR_c$ produces a maximum-length sequence of period $P_c = 2^{128} - 1$.

At time instant t, the contents of stages 0 and 126 of $LFSR_c$ are input to the function f_c and the output of f_c is an integer $c(t)$, such that $c(t) \in \{1, 2, 3, 4\}$. The function f_c is given by

$$f_c(x_0, x_{126}) = 2(x_0) + x_{126} + 1.$$

This function was chosen to be a bijective mapping so that the distribution of integers $c(t)$ is close to uniform. Thus $c = \{c(t)\}_{t=1}^{\infty}$ is a periodic integer sequence with period equal to $P_c = 2^{128} - 1$.

2.2 Data Generation Subsystem

The data-generation subsystem of LILI-II uses the integer sequence c produced by the clock-control subsystem to control the clocking of a binary LFSR, $LFSR_d$, of length $L_d = 127$. The contents of a fixed set of $n = 12$ stages of $LFSR_d$ are input to a specially constructed Boolean function, f_d. The truth table for this function is given in the Appendix. The binary output of f_d is the keystream bit $z(t)$. After $z(t)$ is produced, the two LFSRs are clocked and the process repeated to form the keystream $z = \{z(t)\}_{t=1}^{\infty}$.

The feedback polynomial of $LFSR_d$ is chosen to be the primitive polynomial

$$x^{127} + x^{121} + x^{120} + x^{114} + x^{107} + x^{106} + x^{103} + x^{101} + x^{97} + x^{96} + x^{94}$$
$$+x^{92} + x^{89} + x^{87} + x^{84} + x^{83} + x^{81} + x^{76} + x^{75} + x^{74} + x^{72} + x^{69} + x^{68}$$
$$+x^{65} + x^{64} + x^{62} + x^{59} + x^{57} + x^{56} + x^{54} + x^{52} + x^{50} + x^{48} + x^{46} + x^{45}$$
$$+x^{43} + x^{40} + x^{39} + x^{37} + x^{36} + x^{35} + x^{30} + x^{29} + x^{28} + x^{27} + x^{25} + x^{23}$$
$$+x^{22} + x^{21} + x^{20} + x^{19} + x^{18} + x^{14} + x^{10} + x^{8} + x^{7} + x^{6} + x^{4} + x^{3} + x^{2} + x + 1$$

and the initial state of $LFSR_d$ is not permitted to be the all zero state. Then $LFSR_d$ produces a maximum-length sequence of period $P_d = 2^{127} - 1$, which is a Mersenne Prime.

The 12 inputs to f_d are taken from the $LFSR_d$ positions (0, 1, 3, 7, 12, 20, 30, 44, 65, 80, 96, 122), which form a full positive difference set (see [11]). The function selected for f_d is balanced, highly nonlinear and has first order correlation immunity. The function f_d has nonlinearity of 1992, and an algebraic

order of 10. The truth table of the Boolean function f_d is given in Hex in the Appendix. A function with these properties was selected in order to provide greater security against possible attacks (see Section 4).

2.3 Key loading and re-keying

In some communication systems, errors occur which require that the entire message be re-sent. When a synchronous stream cipher such as LILI-II is used, then security requires that a different keystream sequence be used. To achieve this, a re-keying algorithm is used to combine the secret key, k, with v_i, the 128-bit initialisation vector for the i^{th} re-keying. If the initialisation vector has length less than 128 bits, then multiple copies of the vector will be concatenated, repeated and truncated as required, to form a 128-bit vector. Typically the v_i sequence is publicly known, thus introducing security concerns. The existence of re-keyed messages allows the possibility for attacks based on the re-keying. These scenarios have subtle differences as described below.

If only a single segment of keystream is known, (no re-keying occurs) then to break a particular instance of the cipher, the cryptanalyst must recover the initial internal state S_0, using knowledge of the structure of the keystream generator and some amount of the keystream, z. Attacks based on this scenario are discussed in Section 4. No attack which is better than exhaustive key search has been identified.

In contrast, with resynchronisation occurring, the cryptanalyst has access to related keystreams produced under the same k and for different but known v_i, typically sequential or differing in only a few bits. The cryptanalyst's task is then to recover k, given a set of (v_i, z_i) pairs. For security in this scenario, it is required that the re-keying process should does not leak information about the key k.

We now describe the proposed method for initial key loading and for re-keying of the LILI-II keystream generator. The process to generate the initial state for the keystream generator uses the generator itself twice. The starting state of $LFSR_c$ is obtained by XORing the two 128-bit binary strings k and v_i. The starting state of $LFSR_d$ is obtained by deleting the first bit of k and the last bit of v_i, and XORing the two resulting 127-bit binary strings. Now the cipher is run to produce an output string of length 255 bits. For the second application of the cipher, the first 128 bits of this output string are used to form the initial state of $LFSR_c$, and the remaining 127 bits are used to form the initial state of $LFSR_d$. The cipher is run again to produce an output string of length 255 bits. The output from this second application is used to form the initial state of the keystream generator when we begin keystream production. As previously, the first 128 bits form the initial state of $LFSR_c$, and the remaining 127 bits form the initial state of $LFSR_d$.

By employing the LILI-II algorithm itself, we take advantage of both the known security properties of the algorithm and also its fast implementation. Due to the high security of LILI-II we conclude that the best attack in the re-keying scenario is exhaustive key search.

3 Keystream Properties

Several properties of pseudorandom binary sequences are considered basic security requirements: a sequence that does not possess these properties is generally considered unsuitable for cryptographic applications. Basic requirements for pseudorandom binary sequences are a long period, high linear complexity and good statistics regarding the distribution of zeroes and ones in the output. High linear complexity avoids an attack using the Berlekamp-Massey [15] algorithm.

Using the results from [21] it can be shown that for the keystream of LILI-II:
· the period is $(2^{128}-1)*(2^{127}-1)$,
· the linear complexity is at least 2^{175}, and
· the ratio of ones to zeroes is $\frac{2^{126}}{2^{126}-1} \approx 1$.

4 Possible Attacks

A number of attacks should be considered with respect to the LILI-II keystream generator. These are known-plaintext attacks conducted under the standard assumption that the cryptanalyst knows the complete structure of the generator, and the secret key is only the initial states of the component shift registers. The attacks we consider here are only in the no-rekeying scenario. In the re-keying scenario, there may be some related key attack like that suggested in [3], however the high security of the re-keying algorithm we have proposed prevents any rekeying attack from being effective.

Firstly, a general cryptanalytic attack on stream ciphers known as a T/M/D (time/memory/data) tradeoff attack is discussed in relation to LILI-II. Alternatively, for keystream generators based on more than one LFSR where the key consists of the initial states of the LFSRs, such as the LILI-II generator, divide-and-conquer attacks targeting individual LFSRs should be considered. For these attacks, the given keystream is viewed as an irregularly decimated version of a nonlinearly filtered $LFSR_d$ sequence, with the decimation under the control of $LFSR_c$. For divide and conquer attacks, we deal firstly with attacks that target $LFSR_d$, and then with those attacks that target $LFSR_c$. We shall describe these attacks in relation to the general LILI keystream generator as described in [21], and point out why such attacks are not feasible for LILI-II.

4.1 Time/Memory/Data Tradeoff Attacks

The objective of the time-memory tradeoff attacks is to recover the internal state at a known time. The attacks are conducted in two stages. During a preprocessing phase, the cryptanalyst constructs a lookup table, mapping possible internal states to prefixes of the corresponding output keystreams. In the real time phase of the attack, the cryptanalyst takes a segment of known keystream and tries to find the corresponding internal state, by searching through the lookup table.

In [3] Babbage described time-memory tradeoff attacks against LILI-128 with complexity less than 2^{128}. As mentioned in Section 1, LILI-II is designed to

overcome these attacks through the use of longer LFSRs providing a 255 bit internal state, rather than the 128 bits of LILI-128. We analyse LILI-II in relation to the time-memory tradeoff attacks below.

Let S, M, T, P and D denote the cardinality of the internal state space, the memory (in binary words of size equal to $\log_2 S$), the computational time (in table lookups), the precomputation time (in table lookups), and the amount of data (without re-keying, this is the length of known keystream), respectively.

For the time-memory attacks described in [2, 12] $T \cdot M = S$, $P = M$ and $D = T$. For example, a $2^{128}.2^{127} = 2^{255}$ tradeoff could be used although, as this requires time equivalent to exhaustive key search, and an excessive amount of memory and known keystream, such an attack is certainly not feasible. The more general time-memory-data tradeoff [4] asserts that $T \cdot M^2 \cdot D^2 = S^2$, $P = S/D, D^2 \leq T$. This decreases D at the cost of increasing P. For example, one may choose $M = D = S^{1/3}$ and $T = P = S^{2/3}$, but for LILI-II, with $S = 255$, this gives $M = D = 2^{85}$ and $T = P = 2^{170}$, clearly worse than exhaustive key search. Alternatively, to reduce the time required (with a corresponding reduction in D), we can increase the amount of memory required, and obtain, for example, $M = 2^{127}$, $D = 2^{64}$ and $T = 2^{128}$, although this is still no better than exhaustive search, and requires an excessive amount of memory. The tradeoffs permitted by this attack result in either M or T being in excess of 2^{128}, when applied to LILI-II.

In any case, the use of the initialisation scheme (the key-loading/re-keying algorithm) to expand the 128-bit secret key into a 255 bit initial state renders the time-memory attacks on LILI-II infeasible, as their performance is at best, no better than exhaustive key search.

4.2 Attacks on Irregularly Clocked $LFSR_d$

Suppose a keystream segment of length N is known, say $\{z(t)\}_{t=1}^{N}$. This is a decimated version of a segment of length M of the underlying regularly clocked nonlinearly filtered $LFSR_d$ sequence, $g = \{g(i)\}_{i=1}^{M}$, where $M \geq N$. The objective of correlation attacks targeting $LFSR_d$ is to recover the initial state of $LFSR_d$ by identifying the segment $\{g(i)\}_{i=1}^{M}$ that $\{z(t)\}_{t=1}^{N}$ was obtained from through decimation, using the correlation between the regularly clocked sequence and the keystream, without knowing the decimating sequence.

For clock-controlled shift registers with constrained clocking (so that there is a fixed maximum number of times the data shift register may be clocked before an output bit must be produced), correlation attacks based on a constrained Levenshtein distance and on a probabilistic measure of correlation are proposed in [8] and [9], respectively, and further analysed in [10]. These attacks could be adapted to be used as the first stage of a divide-and-conquer attack on LILI-II. The rest of this section describes how such an attack would be performed.

For a candidate initial state of $LFSR_d$, say $\{\hat{d}(i)\}_{i=1}^{L_d}$, use the known $LFSR_d$ feedback function to generate a segment of the $LFSR_d$ sequence, $\{\hat{d}(i)\}_{i=1}^{M+L_d-1}$, for some $M \geq L_d$. Then use the known filter function f_d to generate a segment of

length M of the output of the nonlinear filter generator when regularly clocked, $\{\hat{g}(i)\}_{i=1}^{M}$. A measure of correlation between $\{\hat{g}(i)\}_{i=1}^{M}$ and $\{z(t)\}_{t=1}^{N}$ is calculated (either the Constrained Levenshtein Distance (CLD) [8], or the Probabilistic Constrained Edit Distance (PCED) [9]) and the process repeated for all $LFSR_d$ initial states.

In either case, the attack is considered successful if only a few initial states are identified. As the correlation attack based on the PCED takes into account the probability distribution of the decimating sequence, it is statistically optimal and may be successful in cases where the embedding attack based on the CLD is not, such as for larger values of m. The value of M is a function of N and m. If $M = 2^m \times N$, then the probability of not identifying the correct $LFSR_d$ initial state is zero.

The second stage of a divide-and-conquer attack on the generator is the recovery of the initial state of the second shift register. This can be performed as in [20]. From the calculation of the edit distance (either CLD or PCED) between $\{\hat{g}(i)\}_{i=1}^{M}$ and $\{z(t)\}_{t=1}^{N}$, form the edit distance matrix, and use this to find possible edit sequences. From each possible edit sequence, form a candidate integer sequence $\{\hat{c}(t)\}_{t=1}^{N}$. From this, the underlying binary sequence $\{\hat{a}(t)\}_{t=1}^{N}$ and hence the candidate initial state of $LFSR_c$ can be recovered. To determine whether the correct initial states of both LFSRs have been recovered, use both candidate initial states to generate a candidate keystream and compare it with the known keystream segment.

To conduct either of these correlation attacks requires exhaustive search of the $2^{127} - 1$ $LFSR_d$ initial states. For each $LFSR_d$ initial state, the attacks require calculation of either the CLD or the PCED, with computational complexity $O(N(M - N))$. Finally, further computational complexity is added in finding the corresponding $LFSR_c$ initial state. For either correlation attack, the minimum length of keystream required for a successful attack on $LFSR_d$ is linear in L_d, but exponential or even super-exponential in 2^m (see [10]). For $m = 2$, the required keystream length [23] is prohibitively large.

This is supported by the work of Chambers and Gollmann [5] on embedding, which indicates embeddings for $d > 3$ require impractically long output sequences. The LILI class of ciphers uses $d = 4$. The complexity of such an attack on LILI-II is $(2^{127} - 1)$ multiplied by the complexity of computing the correlation measure, with the additional complexity of recovering the corresponding $LFSR_c$ state. That is, the performance of divide and conquer attacks which target $LFSR_d$ are much worse than exhaustive key search.

4.3 Attacks Targeting $LFSR_c$

A possible approach to attacking the proposed generator is by targeting the clock-control sequence produced by $LFSR_c$. Guess an initial state of $LFSR_c$, say $\{\hat{a}(t)\}_{t=1}^{L_c}$. Use the known $LFSR_c$ feedback function and the function f_c to generate the decimating sequence $\{\hat{c}(t)\}_{t=1}^{N}$ for some $N \geq L_c$. Then position the known keystream bits $\{z(t)\}_{t=1}^{N}$ in the corresponding positions of $\{\hat{g}(i)\}_{i=1}^{\infty}$, the nonlinear filter generator output when regularly clocked. At this point we

have some (not all consecutive) terms in the nonlinear filter generator output sequence and are trying to reconstruct a candidate initial state for $LFSR_d$.

Note that the amount of trial and error involved in guessing the initial state of $LFSR_c$ is the same as for guessing the secret key. Thus, the performance of any divide and conquer attack targeting $LFSR_c$ will be worse than exhaustive key search. Nevertheless, we outline several ways such an attack could proceed.

Consistency Attack One method is to use the known filter function f_d to write equations relating terms in the underlying $LFSR_d$ sequence to terms in $\{\hat{g}(i)\}_{i=1}^{\infty}$. Reject the guessed initial state $\{\hat{c}(t)\}_{t=1}^{L_c}$ when the equations are inconsistent. This is a generalisation of the linear consistency test [22]. The feasibility of such an approach depends on the number of inputs to f_d, on the tap positions producing these inputs and on some properties of f_d such as its nonlinearity and order of correlation immunity. For example, this attack is complicated if the tap positions are chosen according to a full positive difference set (see [11]), as in the case of LILI-II.

Attacks on Regularly Clocked $LFSR_d$ An alternative approach is to use a correlation attack on the nonlinear filter generator [18] to recover a linear transform of the $LFSR_d$ sequence, and then recover the $LFSR_d$ initial state. However, this is complicated by not having consecutive terms in the regularly clocked nonlinear filter generator sequence. The feasibility of such an attack primarily depends on the use of a feedback polynomial of $LFSR_d$ that is of low weight or has low weight polynomial multiples and on the nonlinearity of f_d.

The feedback polynomial of $LFSR_d$ has been selected so that it does not have low-weight polynomial multiples of relatively small degrees, in order to avoid the vulnerability to fast correlation attacks on $LFSR_d$ when regularly clocked.

An alternative correlation attack on a (regularly clocked) nonlinear filter generator which could be applied at this point is the conditional correlation attack [1], with a difference that the known output bits are not consecutive. The feasibility of such an attack depends on the number of inputs to the filter function and on the tap positions. The use of a full positive difference set for the tap positions, as suggested in [11], and of a filter function with correlation-immunity order greater than zero renders this attack infeasible. The number and positions of taps for the filter function, f_d, have been chosen to ensure resistance to the attacks discussed in Section 4.3. This was the motivation for our choice of 12 inputs from tap positions which do form a full positive difference set.

Finally, the inversion attack [11] can be adapted to deal with the case of non-consecutive output bits, but the associated branching process is then supercritical, because more than one bit has to be guessed at a time. As a consequence, the computational complexity may be prohibitively high even if the tap positions are not spread across the $LFSR_d$ length.

Jönsson and Johansson's Attack In [13], a divide and conquer attack on the LILI-128 stream cipher was proposed. The attack involved guessing an ini-

tial state for $LFSR_c$, and then solving the remaining $LFSR_d$ with the clocking known. This is a fast correlation attack, that is not based on iterative probabilistic decoding, and as such does not require the available keystream bits to be consecutive.

Applying any of the approaches discussed above requires exhaustive search over the $LFSR_c$ initial state space and additional computation for each candidate $LFSR_c$ state. Thus, all of these attacks are worse than exhaustive search on the 128-bit secret key.

4.4 Summary of Security Claims

In this section we summarise the claims we make for the security of LILI-II. Firstly, the period at around 2^{255} is sufficiently large. Secondly the linear complexity is conjectured to be at least 2^{175}, so that at least 2^{176} consecutive bits of known plaintext are required for the Berlekamp-Massey attack. This is an infeasible amount of text to collect. Thirdly, we conjecture that the complexity of divide and conquer attacks on LILI-II is in excess of 2^{128} operations, and additionally these attacks require knowledge of a large amount of keystream. The best known attack is therefore conjectured to be exhaustive search on the 128-bit key. This indicates that LILI-II is a secure synchronous stream cipher.

5 Efficiency and Implementation

5.1 Software Efficiency

The current software implementation of LILI-II runs at 6.07 Mbps on a 300MHz Pentium III using Borland C++ 5.5, and 3.75 Mbps on a 450MHz SPARCv9 using gcc 2.95.3. In comparison, implementations of LILI-128 achieved speeds of 6.65 MHz and 4.12 MHz under the respective conditions.

There are several internal differences between the implementation of LILI-128 and that of LILI-II. These differences include using Galois structure (rather than traditional Fibonacci style), for the LFSR state transitions, the increase in the size of the registers and increasing the number of inputs to the filter function from 10 to 12. These aspects have different effects on the speed of the design in software.

The use of Fibonacci style LFSRs is traditional in stream ciphers. In this style, the feedback polynomial selects a set of bits which are added mod 2 (XOR) to create the feedback bit which is shifted in to the LFSR. In contrast, the Galois style checks the value of the bit shifted out and if it is 1 then a constant vector (the feedback polynomial) is XORed in to the LFSR. These two styles can be seen as the time reversal of each other. Their state sequences have the same period. We believe that there is no security difference between the methods, so we choose to use the Galois style as it is faster in software.

The length of the $LFSR_c$ and $LFSR_d$ are increased from 39 and 89 bits to 128 bits and 127 bits, respectively. These structures now take 4 words each

(on 32-bit processors). The clocking of the LFSRs will take more operations due to the extra words. This slight speed reduction compared to LILI-128 is unavoidable, but will be less of a problem when implemented on processors with 64 or 128 bit words. LILI-II will be slightly faster on those processors.

The most efficient way to implement the shifting of multiple-word LFSRs requires every second word to be physically reversed and shifted the other way. This removes the need for an extra shift per word to properly place the bit shifting between words. This reversal was taken into account during the design of the boolean function input assembly.

An interesting part of the implementation is the selection of bits from $LFSR_d$ as input to the Boolean function. We have chosen a full positive difference set (FPDS) to describe the bit positions selected, as this maximised resistance to correlation attacks. However, each bit must be selected by a logical mask and shifted to the desired position for input to a look-up-table. This means up to two operations per bit are required to assemble the 12-bit boolean function input value. By careful analysis of the chosen FPDS and given the abovementioned word reversal implementation style, we were able to slightly reduce the number of operations to assemble the value for input to the filter function. However the overall speedup from this optimisation is only about 2%.

5.2 Hardware Efficiency

The basic approach to the hardware design of LILI-II does not differ significantly from that proposed for LILI-128. Both ciphers can run at the clock speed with very small space required. The timing is simulated using a Max+plus II from the ALTERA Co., the logic circuit is implemented for an FPGA device (EPF10K20RC240-3), and the throughput stability is analysed up to a rate of 50 Mbps (ie. higher than the T3 rate at 45 Mbps, plus the maximum delay routine in the proposed design was below 20ns) with a 50MHz system clock, We have translated/simulated our VHDL design for Lucent's ASIC device (LV160C, 1.3 micrometer CMOS and 1.5V technology) and it can achieve a throughput of about 500 Mbps with a 0.13 micrometer semiconductor for the maximum path delay below 1.8ns.

Since the LILI-II cipher is a clock-controlled keystream generator, the keystream data rate is degraded in a clock-synchronised hardware logic design. Basically the clock-controlled $LFSR_d$ in the LILI-II cipher requires clocking at up to 4 times the rate of the $LFSR_c$. If the same clock is selected for both then the system throughput will be lowered. Accordingly, we propose a 4-bit parallel $LFSR_d$ where each register bit includes four variable data routines for feedback or shifting within the $LFSR_d$. After shifting $LFSR_d$, output sequences are generated using the nonlinear filter function from the 12 input taps in FPDS structure. The primitive polynomials require only some XOR operations. The data required to store the 12-input Boolean function is four times the space required for the 10-bit function of LILI-128, however at 512 bytes, this is still no problem even for memory-tight applications.

In the design for hardware, we have used an idea about the parallel structure of LFSR, from [14]. In most clock-synchronised logic designs, the feedback/shift in each register is implemented by a synchronised system clock to stabilise the hardware. However, since $LFSR_d$ in LILI-II requires many (1 to 4) clocks within a system clock period, this is a serious drawback. A frequency multiplier has been suggested to solve this problem, yet this is inappropriate for high-speed communications due to the small margin in a clock time interval. Accordingly, a 4-bit parallel $LFSR_d$ where each register has four variable data routines within the $LFSR_d$ was used. Note that the $LFSR_c$ and f_c can be easily implemented using a general shift register with feedback and a full adder device. Whereas each register in the 127-stage $LFSR_d$ represented by $d_0, d_1, ...d_{126}$ can randomly jump 1-4 registers from right to left according to the output of f_c. This $LFSR_d$ can be implemented using 127 D flip-flops and 127 multiplexors (4-1 MUX). For example the d_{122} register can select a 1-bit input d_{123+f_c} from the 4-bit registers d_{123} through d_{126}, where the selection signal is the output of f_c, implemented using a 1-bit full adder. As the $LFSR_d$ feedback polynomial is fixed, the four feedback logics can be precalculated from that vector, as linear functions of the current state.

6 Conclusion

In this paper, the LILI-II keystream generator, intended for use in stream cipher applications, is proposed. The design is simple: the generator is based on two binary LFSRs and uses two combining functions. The security of this keystream generator has been investigated with respect to currently known styles of attack. With the chosen parameters, LILI-II provides the basic security requirements for cryptographic sequences, such as a long period and high linear complexity. Also, LILI-II is immune to current known-plaintext attacks, conducted under the assumption that the cryptanalyst knows the entire structure of the generator and also the 128-bit initialisation vector. The 128-bit secret key is used only to form the initial states of the two LFSRs, using the re-keying algorithm outlined. We conjecture that the fastest possible attack on LILI-II is exhaustive key search.

The use of both nonlinear combining functions and irregular clocking in LFSR based stream ciphers is not a novel proposal, and has been employed in previous constructions. However, in this proposal the two approaches are combined in a manner that produces output sequences with provable properties with respect to basic cryptographic security requirements and also provides security against currently known cryptanalytic attacks.

The design is transparent, relying on basic known results in LFSR theory. In addition, LILI-II is easy to implement in software or hardware and, as it employs only simple components, can be implemented efficiently on any platform. The cipher is especially efficient in hardware. Our demonstration software implementation of LILI-II runs at approximately ninety percent of the speed that LILI-128 achieves, while giving far greater security. The speed we obtain in soft-

ware is still fast enough for most applications, and the very fast hardware speed is sufficient even for applications requiring both high speed and high security.

Finally, the designers would like to state that no weakness has been deliberately inserted into the LILI-II design.

References

1. R. Anderson. Searching for the Optimum Correlation Attack. In *Fast Software Encryption - Leuven'94*, volume 1008 of *Lecture Notes in Computer Science*, pages 137–143. Springer–Verlag, 1995.
2. S. Babbage. A space/time tradeoff in exhaustive search attacks on stream ciphers. European Convention on Security and Detection, IEE Conference Publication No. 408, May 1995.
3. S. Babbage. Cryptanalysis of LILI-128. Available at *https://cosic.esat.kuleuven.ac.be/nessie/reports/extwp3-001-2.pdf*
4. A. Biryukov and A. Shamir. Cryptanalytic time/memory/data tradeoffs for stream ciphers. In *Advances in Cryptology - ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 1–13. Springer–Verlag, 2000.
5. W.G. Chambers and D. Gollmann. Embedding attacks on step[1..D] clock-controlled generators. Electronics Letters, vol.36 pp.1771-1773, 2000.
6. E. Dawson, A. Clark, J. Golić, W. Millan, L. Penna and L. Simpson. The LILI-128 Keystream Generator. Available at *https://www.cosic.esat.kuleuven.ac.be/nessie/workshop/submissions.html.*
7. C. Ding, G. Xiao and W. Shan. *The Stability Theory of Stream Ciphers.* Volume 561 of *Lecture Notes in Computer Science*. Springer–Verlag, 1991.
8. J. Dj. Golić and M. J. Mihaljević. A Generalised Correlation Attack on a Class of Stream Ciphers Based on the Levenshtein Distance. *Journal of Cryptology*, vol. 3(3), pp. 201–212, 1991.
9. J. Dj. Golić and S. Petrović. A Generalised Correlation Attack with a Probabilistic Constrained Edit Distance. In *Advances in Cryptology - EUROCRYPT'92*, volume 658 of *Lecture Notes in Computer Science*, pages 472–476. Springer–Verlag, 1992.
10. J. Dj. Golić and L. O'Connor. Embedding and Probabilistic Correlation Attacks on Clock-Controlled Shift Registers. In *Advances in Cryptology - EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 230–243. Springer–Verlag, 1994.
11. J. Dj. Golić. On the Security of Nonlinear Filter Generators. In *Fast Software Encryption - Cambridge'96*, volume 1039 of *Lecture Notes in Computer Science*, pages 173–188. Springer–Verlag, 1996.
12. J. Dj. Golić. Cryptanalysis of Alleged A5 stream cipher. In *Advances in Cryptology - EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 239–255. Springer–Verlag, 1997.
13. F. Jönsson and T. Johansson A Fast Correlation Attack on LILI-128. *http://www.it.lth.se/thomas/papers/paper140.ps*
14. H-J. Lee and S-J. Moon Parallel Stream Cipher for Secure High-Speed Communications. *Signal Processing*, vol. 82, no. 2, pp. 137-143, 2002.
15. J. Massey. Shift-Register Synthesis and BCH Decoding. *IEEE Trans. Inform. Theory*, IT-15:122-127, January 1969.

16. W. Meier and O. Staffelbach. Fast Correlation Attacks on Certain Stream Ciphers. *Journal of Cryptology*, vol. 1(3), pp. 159–167, 1989.
17. R. Rueppel. *Analysis and design of stream ciphers*. Springer–Verlag, Berlin, 1986.
18. M. Salmasizadeh, L. Simpson, J. Dj. Golić and E. Dawson. Fast Correlation Attacks and Multiple Linear Approximations. In *Information Security and Privacy - Nepean '97*, volume 1270 of *Lecture Notes in Computer Science*, pages 228–239. Springer–Verlag, 1997.
19. T. Siegenthaler. Decrypting a Class of Stream Ciphers Using Ciphertext Only. *IEEE Trans. Computers*, vol. C-34(1), pp. 81–85, 1985.
20. L. Simpson, J. Dj. Golić and E. Dawson. A Probabilistic Correlation Attack on the Shrinking Generator. In *Information Security and Privacy - Brisbane '98*, volume 1438 of *Lecture Notes in Computer Science*, pages 147–158. Springer–Verlag, 1998.
21. L. Simpson, E. Dawson, J. Dj. Golić and W. Millan. LILI Keystream Generator. *Proceedings of the Seventh Annual Workshop on Selected Areas in Cryptology - SAC'2000*, volume 2012 of *Lecture Notes in Computer Science*, pages 248–261, Springer–Verlag, 2000.
22. K. C. Zeng, C. H. Yang and T. R. N. Rao. On the Linear Consistency Test (LCT) in Cryptanalysis with Applications. In *Advances in Cryptology - CRYPTO'89*, volume 434 of *Lecture Notes in Computer Science*, pages 164–174. Springer–Verlag, 1990.
23. M. Živković. An Algorithm for the Initial State Reconstruction of the Clock-Controlled Shift Register. *IEEE Trans. Inform. Theory*, vol. IT-37, pp. 1488–1490, Sept. 1991.

APPENDIX

Output Boolean Function for LILI-II Stream Cipher

This is the truth table (in hex) of the output function f_d:

C22C2CC22CC2C22CC22C2CC2C22C2CC2
C22CC22C2CC22CC2C22CC22CC22CC22C
C2C22C2C2C2CC2C2C2C22C2CC2C22C2C
C2C2C2C22C2C2C2CC2C2C2C2C2C2C2C2
CC2222CC22CCCC22CC2222CCCC2222CC
CC22CC2222CC22CCCC22CC22CC22CC22
CCCC22222222CCCCCCCC2222CCCC2222
CCCCCCCC22222222CCCCCCCCCCCCCCCC
A44A4AA44AA4A44AA44A4AA4A44A4AA4
A44AA44A4AA44AA4A44AA44AA44AA44A
A4A44A4A4A4AA4A4A4A44A4AA4A44A4A
A4A4A4A44A4A4A4AA4A4A4A4A4A4A4A4
AA4444AA44AAAA44AA4444AAAA4444AA
AA44AA4444AA44AAAA44AA44AA44AA44
AAAA44444444AAAAAAAA4444AAAA4444
AAAAAAAA4444444468868668866868866
688686686886866868686886868866868
688668866886686868686868868668668
686886866868868668686868868668686
686868686868686866868866886666688
668888666688886666868668888668866
668866886688668866666888888866666
666688886666688886666666688888888
0EE0E00EE00E0EE00EE0E00E0EE0E00E
0EE00EE0E00EE00E0EE00EE00EE00EE0
0E0EE0E0E0E00E0E0E0EE0E00E0EE0E0
0E0E0E0EE0E0E0E000EEEE00EE0000EE
00EEEE0000EEEE0000EE00EEEE00EE00
4000AEEEE6EE0800CAA028CAC642424E
C2CA26C88C626C842206C26CC4AAAC84
22C8EA0A2866404E2286286868668628
EC84022E84642EA8C86422C42A2C8AC6

The Boolean Function has 12 inputs and these properties:
Balanced, CI(1), Order=10, Nonlinearity=1992, No Linear Structures.