A Time-Memory Tradeoff Attack Against LILI-128

Markku-Juhani Olavi Saarinen

Helsinki University of Technology
Laboratory for Theoretical Computer Science
P.O. Box 5400, FIN-02015 HUT, Finland
mjos@tcs.hut.fi

Abstract. In this note we discuss a novel and simple time-memory tradeoff attack against the stream cipher LILI-128. The attack defeats the security advantage of having an irregular stepping function. The attack requires 2^{46} bits of keystream, a lookup table of 2^{45} 89-bit words and computational effort which is roughly equivalent to 2^{48} DES operations.

1 Introduction

The LILI-128 keystream generator [5] is a LFSR-based synchronous stream cipher with a 128 bit key. It was accepted as one of six candidate stream ciphers for NESSIE, but was rejected from the second round.

In the original LILI-128 specification, the authors conjecture that the complexity of divide and conquer attacks is "at least 2^{112} operations, requiring knowledge of at least 1700 known keystream bits".

In this paper, we use the approximate number of "equivalent DES operations" as a measure of computational efficiency. While the number of bit operations is an useful measure in asymptotic analysis of algorithms, we feel that our approach is more appropriate for the purposes of this paper, since it allows easy comparison of security level to other ciphers (esp. block ciphers).

1.1 Previous Work

After its initial release, some cryptanalytic results on LILI-128 has been published [6,9]. The best known attacks are:

- In [8], Jönsson and Johansson describe an attack requiring 2^{71} bit operations (2^{79} in precomputation phase), 2^{30} keystream bits and an off-line precomputed table with 2^{40} entries.
- In [1], Babbage discusses a rekeying attack and generic time-memory tradeoff attacks.

J. Daemen and V. Rijmen (Eds.): FSE 2002, LNCS 2365, pp. 231–236, 2002.

1.2 Time/Memory/Data Tradeoffs

In 1980 Hellman introduced a technique for breaking block ciphers using time-memory tradeoffs [7]. An analogous (but very different) technique for stream ciphers was proposed by Babbage in 1995 [2], although the underlying idea is not algorithmically new. More recently, Biryukov, Shamir and Wagner combined these approaches in work related to the A5/1 cipher [3,4].

The basic idea of Time/Memory/Data tradeoff attacks against stream ciphers is as follows. Most stream ciphers can be described in terms of a state x_i (which is characterized by N, its size) and two functions, *Step* and *Output*. The initial state x_0 is derived from a secret key. To generate one bit of keystream $z(i)$, *Output* is invoked, followed by an update of the internal state using *Step*:

$$z(i) = Output(x_{i-1})$$
$$x_i = Step(x_{i-1})$$

The attack consists of two stages:

1. **Off-line preprocessing stage.** We pick random x_i states and compute the bit sequence $z(i), z(i+1), \ldots, z(i+O(\log N))$. This output bit sequence is stored in a table together with the random state x_i. We sort the list in increasing order of the output bit sequence.
2. **On-line computation phase.** Each $O(\log N)$-bit window of the known keystream is considered. For each window we perform a table lookup to determine whether or not this state was one of the states considered in the preprocessing stage. If the state is found, we can compute keystream bits after or (if *Step* is invertible) before the known keystream segment.

It can be shown that $O(\sqrt{N})$ bits of known keystream, $O(\log N\sqrt{N})$ bits of memory, and $O(\log N\sqrt{N})$ time is sufficient to find one internal state x_i. From this internal state we can derive future bits or possibly even the original session key. It turns out that in case of LILI-128, further improvements are possible over this generic attack.

2 Description of LILI-128

LILI-128 uses two LFSRs, $LFSR_c$ and $LFSR_d$. $LFSR_c$ has an internal state of 39 bits and is clocked once for each output bit. $LFSR_d$ has an internal state of 89 bits and is clocked 1 to 4 times, depending on two bits in $LFSR_c$. During key setup phase a $128 = 39 + 89$ bit cryptovariable is directly loaded into these two registers.[1]

In the following, we let $t_0, t_1, \ldots, t_{38}$ denote the individual bits of $LFSR_c$, t_0 being the most significant bit in the register and t_{38} being the least significant

[1] In [6] the authors also discuss other keying methods for LILI-128.

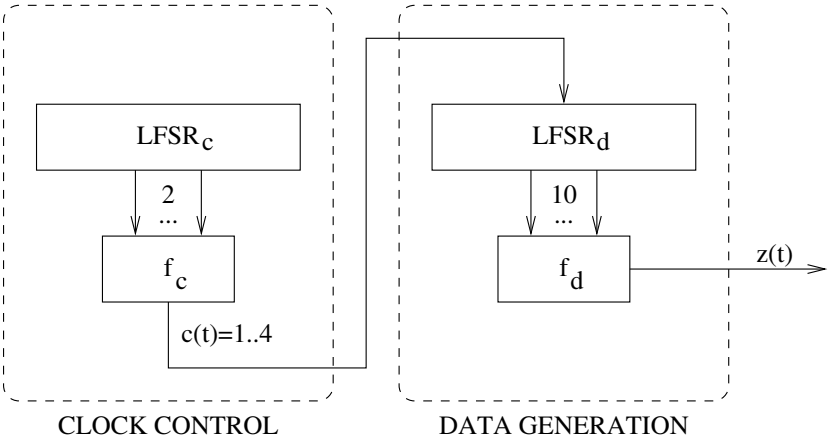

Fig. 1. Overview of the LILI-128 keystream generator.

bit. Similarly we use $u_0, u_1, \ldots, u_{88}$ to denote the individual bits of $LFSR_d$. The primitive polynomial for $LFSR_c$ is

$$x^{39} + x^{35} + x^{33} + x^{31} + x^{17} + x^{15} + x^{14} + x^2 + 1$$

while $LFSR_d$ uses the primitive polynomial

$$x^{89} + x^{83} + x^{80} + x^{55} + x^{53} + x^{42} + x^{39} + x + 1.$$

The procedure for generating keystream is as follows:

1. Ten bits from $LFSR_d$ are fed to a highly nonlinear function f_d, $f_d : \mathbb{F}_2^{10} \to \mathbb{F}_2$ to generate one output bit $z(t)$. [2]

$$z(t) = f_d(u_0, u_1, u_3, u_7, u_{12}, u_{20}, u_{30}, u_{44}, u_{65}, u_{80}).$$

2. Two bits from $LFSR_c$ are fed to a "linear" clock control function $f_c : \mathbb{F}_2^2 \to \mathbb{F}_2$ to form the clocking amount $c(t)$:

$$c(t) = f_c(t_{12}, t_{20}) = 2t_{12} + t_{20} + 1$$

3. The clock control register $LFSR_c$ is clocked once and the data generation register $LFSR_d$ is clocked $c(t)$ times (i.e. 1, 2, 3, or 4 times).

Note that the output bit is indeed generated *before* the LFSRs are clocked, hence effectively halving the key search effort in some applications.

[2] The f_d function is specified as a 1024-entry table in the original specification [5], and is excluded from this paper since it is irrelevant to the present attack.

Lemma 1. *For each $\Delta_c = 2^{39} - 1$ times $LFSR_c$ is clocked, $LFSR_d$ is clocked exactly $\Delta_d = 5 * 2^{38} - 1$ times.* [3]

Proof. We claim that for all cryptovariables and all values t:

$$\sum_{i=1}^{2^{39}-1} c(t+i) = \Delta_d$$

Since the polynomial of $LFSR_c$ is primitive, it's period is $2^{39} - 1 = \Delta_c$ and thus the internal state of the register goes through all possible values except the all zero state $t_0 = t_1 = \ldots = t_{38} = 0$. During this cycle the control bits (t_{12}, t_{20}) have value $(0, 0)$ exactly $2^{37} - 1$ times and values $(0, 1)$, $(1, 0)$, and $(1, 1)$ exactly 2^{37} times, bringing the total sum to $1*(2^{37}-1)+(2+3+4)*2^{37} = 1374389534719 = \Delta_d$.

Lemma 2. *$LFSR_d$ can be stepped by Δ_d number of positions forward or backward by performing a vector-matrix multiplication with a precomputed 89×89 bit matrix over $GF(2)$. The matrix can be constructed with roughly 2^{28} bit operations using a binary matrix exponentiation algorithm.*

Proof. Trivial.

This could also be achieved using a multiplication algorithm in $GF(2^{89})$, but for a constant Δ_d, vector-matrix multiplication actually appears to be slightly faster (and functionally equivalent). The Hamming weight of the matrix is $3949 \approx 2^{11.4}$.

3 The Attack

Although many tradeoffs are possible, we will present a concrete version of the attack where we have tried to minimize the amount of keystream required. The amount of off-line and on-line computation are approximately the same.

3.1 Constructing the Lookup Table

A table of 2^{45} 89-bit words is set up by iterating the following procedure 2^{46} times. A pseudorandom 89-bit value is loaded into $LFSR_d$ and 45 bits from f_d are sampled Δ_d steps apart (see Lemma 2). This 45-bit vector is used as an index in the table to store the original 89-bit register value.

Analysis. The expected proportion of filled slots in the table is $1-e^{-2} = 0.8647$. The table size is $2^{51.48}$ bits. The computational effort required to construct the table is roughly equivalent to 2^{48} DES operations.

[3] This lemma follows implicitly from Theorem 2 in [5]

3.2 Lookup Stage

We have 2^{46} bits of keystream $z(0), z(1), \ldots, z(2^{46}-1)$.

1. For i = 0 to $2^{46} - 44\Delta_c - 1$ Do:
2. idx = $z(i) \mid z(i+\Delta_c) \mid \ldots \mid z(i+44\Delta_c)$
3. Load Table[idx] to $LFSR_d$.
4. Rewind $LFSR_d$ back $\Delta_d \lfloor \frac{i}{\Delta_c} \rfloor$ positions.
5. For j = 0 to 127 Do:
6. If $(f_d(LFSR_d) \neq z(j\Delta_c + (i \bmod \Delta_c))$ break loop.
7. Advance $LFSR_d$ by Δ_d positions.
8. If previous loop was not broken, return $LFSR_d$.

In line 2, a 45-bit index to the lookup table is constructed. In line 3, this index is used to fetch a 89-bit candidate value for $LFSR_d$. In line 4, this guess for $LFSR_d$ is rewinded back a multiple of Δ_d steps so that its output bit should match with $z(i \bmod \Delta_d)$ (see Lemma 1). The loop in lines 5 to 7 compares 128 bits sampled from $LFSR_d$ Δ_d steps apart to keystream bits sampled Δ_c bits apart. If all 128 bits match, the correct value for $LFSR_d$ has been found with high probability and is returned on line 8. This guess can be furthermore verified by performing an exhaustive search for the 39-bit value of $LFSR_c$.

Analysis. The probability of finding the correct value for $LFSR_d$ at least once in the main loop is

$$1 - \left(1 - \frac{0.8647 * 2^{45}}{2^{89}}\right)^{2^{46}-44\Delta_c} \approx 90\%.$$

The inner loop on lines 5 to 7 breaks with high probability after only few tries. The main loop runs for about 2^{45} iterations (before a correct value is found). Therefore we claim that the computational effort is roughly equivalent to 2^{48} DES operations.

4 Conclusions

We have presented a novel time-memory tradeoff attack against LILI-128, which greatly improves the time complexity required to break this cipher over previous published results. The attack requires 2^{46} bits (8 terabytes) of keystream, and therefore is not usable in many applications. We conjecture that LILI-128 can be broken using a lookup table of 2^{45} 89-bit words ($2^{51.48}$ bits) and computational effort equivalent to 2^{48} DES operations.

We therefore feel that the security of LILI-128 is not as high as suggested by the designers. We do not recommend the use of this encryption algorithm for high volumes of data, or as a general-purpose standard for high security applications.

Acknowledgments. The author would like to thank Steve Babbage, Kaisa Nyberg, and anonymous program committee members for very helpful input.

References

1. S. Babbage. *Cryptanalysis of LILI-128*. NESSIE Public Report, `https://www.cosic.esat.kuleuven.ac.be/nessie/reports`, 2001.
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, 1995.
3. A. Biryukov and A. Shamir, *Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers*, Proceedings of ASIACRYPT 2000, LNCS 1976, pp. 1–13, Springer-Verlag, 2000.
4. A. Biryukov, A. Shamir, and D. Wagner, *Real Time Cryptanalysis of A5/1 on a PC*, Proceedings of FSE '2000, LNCS 1978, pp. 1–18, Springer-Verlag, 2001.
5. E. Dawson, J. Golić, W. Millan and L. Simpson, *The LILI-128 Keystream Generator*, Proceedings of the Seventh Annual Workshop on Selected Areas in Cryptology – SAC 2000, LNCS 2012, Springer-Verlag, 2000.
6. E. Dawson, J. Golić, W. Millan and L. Simpson, *Response to Initial Report on LILI-128*, Submitted to Second NESSIE Workshop, 2001.
7. M. E. Hellmab, *A Cryptanalytic Time-Memory Trade-Off*, IEEE Transactions on Information Theory, Vol. IT-26, N 4, pp. 401–406, 1980.
8. F. Jönsson and T. Johansson, *A Fast Correlation Attack on LILI-128.*, Information Processing Letters Vol 81, N. 3, Pages 127–132, 2001.
9. J. White, *Initial Report on the LILI-128 Stream Cipher*, NESSIE Public Report, `https://www.cosic.esat.kuleuven.ac.be/nessie/reports`, 2001.