Cryptanalysis of a Stream Cipher due to Lin and Shepherd

Simon R. Blackburn*
Karl Brincat†
Fauzan Mirza
Sean Murphy

Department of Mathematics
Royal Holloway, University of London
Egham, Surrey TW20 0EX, United Kingdom

August 19, 1998

Abstract

The paper cryptanalyses the stream cipher 'Labyrinth', a cipher recently proposed by Bo Lin and Simon Shepherd. Given only 2^{30} known bits of keystream, the 119 bit key of Labyrinth is recovered in under a second of computation using a DEC Alpha.
Key words: Stream cipher, cryptanalysis, Labyrinth.

1 Introduction

The stream cipher 'Labyrinth' has recently been proposed by Bo Lin and Simon Shepherd [2]. They suggest using Labyrinth in bulk data encryption applications, in particular for ATM packet–switching networks. The cipher

*This author is supported by an E.P.S.R.C. Advanced Fellowship
†This author is supported by an E.P.S.R.C. Research Assistantship

has a 119 bit key and Lin and Shepherd report encryption speeds of 1.6 Gbs^{-1} when the cipher is implemented in hardware.

This paper contains a cryptanalysis of Labyrinth. This requires approximately 2^{30} bits of known keystream (about one second of the cipher's output); simulations indicate that a DEC Alpha would recover the cipher's key in under one second using this attack.

2 The Cipher

This section briefly describes Labyrinth. There are some misprints[1] and ambiguities in the original paper [2], and so we have additionally relied on a software simulation of the algorithm (written in C, and kindly provided by Bo Lin) in producing this description.

Labyrinth may be thought of as comprising two main components.

The first component is a linear finite state machine; its current state may be thought of as a binary vector v of length 107, the next state is obtained by multiplying the current state by a fixed (public) 107×107 invertible matrix M. The output of the finite state machine consists of a binary vector r of length 48. This vector is obtained from the current state v by calculating $r = \phi(v)$, where $\phi : \mathbb{Z}_2^{107} \rightarrow \mathbb{Z}_2^{48}$ is a fixed (public) linear function. The initial state of the finite state machine is controlled by 96 bits of the key — these bits determine 96 of the 107 components of the initial state v_0 of the machine, the remaining components of v_0 are key independent (and public).

The second component is a key- and time-dependent non-linear function f. The 23 key bits of the cipher not involved in determining the initial state of the linear finite state machine are used to determine this non-linear function. The function f takes the 48 bit vector r produced by the finite state machine as input and produces a 32 bit output. We describe f in more detail below.

The 23 bits of key which influence f are broken up into three parts. The first part determines an odd integer t such that $1 \leq t \leq 31$ (and so this part

[1] Three misprints are: 1. Register A should have feedback polynomial $x^{43}+x^6+x^4+x^3+1$ (the polynomial given in the paper is not primitive). 2. In the definition of Register B, the equations should read $y_i = x_{i-1} \oplus (x_{i-1} >> 15)$ and $x_i = y_i \oplus (y_i << 17)$. 3. The first line of the description of the key controlled hash function in Section 2.4 should read $Y = (U \oplus V, V)$.

consists of 4 bits). The second part determines an odd integer p such that $1 \leq p \leq 15$ (and so this part consists of 3 bits). The remaining 16 bits of key material determine a 16 bit string K. Given t, p, K and the 48 bit input vector r, the output of f at time i is calculated as follows. The 48 bit vector r is regarded as eight 6-bit quantities $r_1, r_2, \ldots, r_8$. For each n such that $1 \leq n \leq 8$, r_n is fed into the nth S-Box as detailed in the Data Encryption Standard (DES) [1]; each S-Box produces a 4 bit output x_n. The outputs x_n are concatenated to form a 32 bit quantity x. The 16 most significant bits of x are modified by XORing them with the 16 least significant bits of x, to produce the 32 bit quantity y. The quantity y is then rotated t bits to the left to form the 32 bit quantity y'; we regard y' as a 32 bit integer. Let K'_i be the 16 bit quantity formed by rotating K a total of ip times. This is the only time dependent part of the function f; note that $K'_i = K'_{i+16}$ for all i. The output of f is defined to be the sum modulo 2^{32} of y' and the 32 bit integer ℓ whose 16 least significant bits are equal to the 16 most significant bits of y' and whose 16 most significant bits are equal to K'_i.

The cipher operates as follows. The initial state v_0 of the linear finite state machine is determined by 96 bits of the key. At time i, the cipher outputs a block of 32 bits of keystream given by $f(\phi(vM^i))$, where f depends on the remaining 23 bits of key material and on the time i. The keystream block is then XORed with a 32 bit plaintext block in the classic stream cipher fashion.

3 The Cryptanalysis

We cryptanalyse the cipher in two stages. In Stage I, we obtain most of the key material that is used to control the non-linear function f. More precisely, we obtain the 16 bit string K and the integer p by using the fact that for a fixed key the function f is not quite surjective. We then guess the 4 bits of the key that determine the integer t used in the function f. In Stage II, we use outputs of the S-Box stage of the function f to determine a collection of linear relations that the initial state v_0 of the linear finite state machine should satisfy. If our guess for the integer t in Stage I was correct, these linear relations determine the initial state v_0 (and hence the remainder of the key). If our guess for the integer t was incorrect, the linear relations quickly become inconsistent and so we repeat the process with a different

guess.

Stage I: Key Material Controlling the Non-Linear Function We recover the key material associated with the quantities K and p by using the following lemma.

Lemma 1 *Let A and B be the 16 bit integers formed respectively from the most significant and least significant 16 bits of the 32 bit output of f at time i. If $B \neq 2^{16} - 1$, then the quantity K_i' used in the computation of f has the property that*

$$K_i' \neq A - B - 1 \bmod 2^{16}.$$

Proof. Let W and Z be the integers formed respectively from the most significant and least significant 16 bits of the quantity y' given in the description of f. The final stage of the calculation of f gives us, for some $\epsilon \in \{0, 1\}$,

$$2^{16}K_i' + W + 2^{16}W + Z = 2^{32}\epsilon + 2^{16}A + B.$$

Suppose for a contradiction that $K_i' = A - B - 1 \mod 2^{16}$, so for some $\delta \in \{0, 1\}$, $K_i' = A - B - 1 + \delta 2^{16}$. Then

$$Z = \{(2^{16} - 1)(\epsilon - \delta) + (B - W + 1)\}(2^{16} + 1) + (\epsilon - \delta - 1).$$

If $\epsilon = \delta$, then $Z = -1 \bmod (2^{16} + 1)$, a contradiction.

If $\epsilon = 1, \delta = 0$, then $Z = 0 \bmod (2^{16}+1)$, so $Z = 0$. Thus $2^{16}+B-W = 0$, and so $W - B = 2^{16}$, a contradiction.

If $\epsilon = 0$, $\delta = 1$, then $Z = -2 \bmod (2^{16} + 1)$, so $Z = 2^{16} - 1$. Thus $\{B - W - (2^{16} - 1)\}(2^{16} + 1) = 0$, so $B - W = 2^{16} - 1$. Therefore $W = 0$ and $B = 2^{16} - 1$, a contradiction.

Since all three cases lead to a contradiction, $K_i' \neq A - B - 1 \bmod 2^{16}$ when $B \neq 2^{16} - 1$, as required. ∎

The rotation K_i' of K is used in the production of every sixteenth output block. If we therefore decimate an output stream of blocks by sixteen (consider every sixteenth block), we obtain a sequence of blocks produced using K_i'. Almost every one of those blocks ($B \neq 2^{16} - 1$) rules out a possible value for K_i'. A sequence of 2^{20} such blocks (2^{24} blocks before decimation) would usually rule out all values for K_i' except the true value. This is an incredibly quick method for finding part of the key, as we only have to perform one

16-bit subtraction to eliminate a possible value for K_i'. The value p for the rotation of K can easily be found by finding the value of K_j' for an adjacent decimation and comparing the two values K_i' and K_j'. Simulations indicate that the whole process will take well under a second to complete using a DEC Alpha.

Stage II: The Initial State of the Finite State Machine Suppose that we have determined K and p using the method of Stage I. We now aim to recover the initial state v_0 of the linear finite state machine. We begin by guessing a value for the integer t; there are 16 possible guesses.

Assume that our guessed value of t is correct. Given a sequence of 32 bit keystream blocks, we are able to determine almost all of the corresponding values x that occurred as the output of the S-Box layer during the calculation of the function f; we are able to do this since the transformation that maps the S-Box output x to the output of f is almost always uniquely invertible. By examining the output x_n associated with each S-Box, we may deduce many linear relations that the vector r must satisfy, where r is the 48 bit vector used as input to the S-Box layer.

As an example, suppose that the binary string 1101 is observed at the output of S-Box 1. This implies that the input to S-Box 1 is one of the strings 000100, 001101, 101000 and 111111. All these strings satisfy the relations $b_2 \oplus b_5 = 0$, $b_1 \oplus b_4 \oplus b_5 = 1$ and $b_3 \oplus b_4 \oplus b_6 = 1$, where b_i is the ith input bit of the S-Box.

A fixed output to any S-Box implies at least 3 linear relations of this type, because the four possible inputs to an S-Box for a given output are always contained in an affine subspace of dimension 3. Hence we may deduce at least 24 independent linear relations that the vector r must satisfy by observing the output x of the S-Box layer for a single 32 bit block. (We may calculate these relations either directly or by means of a precomputed look-up table for each S-Box.) Since $r = \phi(v_0 M^i)$, these linear relations may be converted to linear relations that the initial state v_0 must satisfy. Once 107 independent linear relations have been obtained, by calculating enough elements x, the initial state v_0 may be recovered. A simple argument shows that calculating the relations associated with 6 blocks should be more than enough to recover v_0. In this case, v_0 is obtained by row reducing a 108×144 binary matrix.

If our guessed value of t is incorrect, the linear relations that v_0 should

satisfy will be inconsistent, and so we try another value of t until we succeed in reconstructing v_0.

The whole of this procedure is very fast. The whole of Stage II should take a fraction of a second to complete on a DEC Alpha.

4 Final Remarks

We remark that the cipher may be cryptanalysed even if less than 2^{24} blocks of keystream are known, simply by carrying out Stage II of the cryptanalysis above for each guess as to the form of the non-linear function f. We would expect such an attack to be completed in a matter of hours on a DEC Alpha. Note that this attack also works if the non-linear function f is altered so that the transformation from the S-Box output x to the cipher's output becomes invertible.

References

[1] NBS Federal Information Processing Standard Publication 46, *Data Encryption Standard*, US National Bureau of Standards, US Department of Commerce, January 1977.

[2] Bo Lin and Simon Shepherd, 'LABYRINTH: a new ultra high speed stream cipher', in: *Cryptography and Coding, 6th IMA International Conference, Cirencester, UK, December 1997* (M. Darnell, Ed.), Lecture Notes in Computer Science 1355, Springer, Berlin, 1997, pp. 192-198.