Loosening the KNOT

Antoine Joux and Frédéric Muller

DCSSI Crypto Lab, 18 rue du Docteur Zamenhof
F-92131 Issy-les-Moulineaux Cedex, France
{Antoine.Joux,Frederic.Muller}@m4x.org

Abstract. In this paper, we present differential attacks on the self-synchronizing stream cipher KNOT. Our best attack recovers 96 bits of the secret key with time complexity of 2^{62} and requires 2^{40} chosen ciphertext bits.

1 Introduction

KNOT is a self-synchronizing (also called asynchronous) stream cipher (SSSC) proposed by Joan Daemen, René Govaerts and Joos Vandewalle [4] in 1992. In that paper, the authors discussed the design of hardware and software optimized asynchronous stream ciphers and proposed KNOT as an example of such designs. KNOT uses a 96-bit secret key and has a 128-bit internal state.

This paper presents differential attacks on KNOT. Differential attacks are cryptanalytic tools that can be applied against many kinds of secret key cryptosystems. In particular, it was successfully applied to many block ciphers including DES [2], RC5 [6, 7] and other DES-like cryptosystems [1]. The principle of a differential attack is to introduce controlled perturbations in the input of an encryption or decryption function and to observe the corresponding modifications induced on the output. This might allow an attacker to retrieve secret information (for instance subkeys or internal state values).

In this paper, we will modify the ciphertext fed into the decryption function of KNOT and observe the corresponding plaintext. Hence our attack is a chosen ciphertext attack. In Section 2 we present general concepts about self-synchronizing stream ciphers. Section 3 describes the cipher KNOT. Then, Section 4 presents a basic attack on KNOT with time complexity 2^{69} that requires 2^{39} chosen ciphertext bits. This differential attack is improved in Section 5 in order to obtain time complexity 2^{62} and data complexity 2^{40}.

2 Self-synchronizing Stream Ciphers

A self-synchronizing stream cipher is one in which the keystream bit is a function of the key and a fixed number m of previous ciphertext bits. This parameter m is called the *memory* of the cipher.

Let x_t denote plaintext bit number t, y_t the corresponding ciphertext bit and w_t the corresponding keystream bit. The encryption can be described as

$$y_t = x_t \oplus w_t$$

T. Johansson (Ed.): FSE 2003, LNCS 2887, pp. 87–99, 2003.

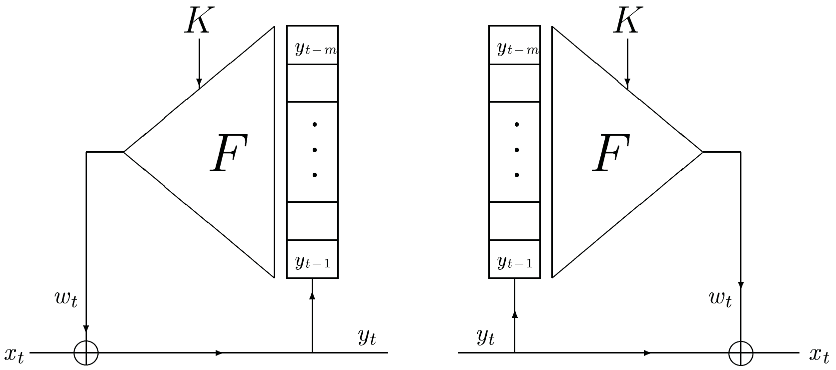

Fig. 1. Canonical representation of a self-synchronizing stream-cipher

where the keystream bit is computed as

$$w_t = F(y_{t-1}, \ldots, y_{t-m}, K)$$

F denotes the keystream function and K the secret key. Without loss of generality, the encryption and decryption function of a self-synchronizing stream cipher can be presented as in Fig. 1.

The basic idea behind SSSCs is to encrypt each plaintext bit with an encryption function depending only on the secret key and the previous m ciphertext bits. Therefore each ciphertext bit can be correctly deciphered as long as the previous m ciphertext bits have been successfully received. This self-synchronization mechanism has many advantages from an engineering point of view. For instance, it may be helpful in contexts where no lower layer of protocol is present to assure error-correction. In particular, it prevents long bursts of error when a bit insertion or deletion occurs during the transmission of the ciphertext, which may be a problem, for instance when using a block cipher. From a security point of view, such ciphers also have some advantages compared to other kind of secret key cryptosystems.

First, since each plaintext bit influences the entire subsequent ciphertext, SSSCs are more likely to be resistant against attacks based on plaintext statistical properties or plaintext redundancies. In the case of synchronous additive stream ciphers, some ciphertext-only attacks have been proposed (see [5]) based only on partial information about the plaintext. In various cases, the entropy of the plaintext is greatly decreased, for instance when a 7-bit ASCII representation is used or when it contains English text.

Furthermore, self-synchronizing stream ciphers are more likely than asynchronous ones to detect single digit modifications in the ciphertext. While such a modification only implies a single digit error in the deciphered plaintext for a synchronous additive stream cipher, up to m ciphertext bits may be incorrectly decrypted in the case of a SSSC. This mechanism provides additional security against active attacks. However, insertion, deletion or replay of ciphertext is still possible against a SSSC and is very difficult to detect. This shows that, in spite of their nice properties, SSSCs cannot guarantee data integrity and authentication without the use of additional mechanisms.

The most commonly used SSSCs are based on block ciphers used in 1-bit Cipher FeedBack (CFB) mode [9]. Such modes are usually quite inefficient in terms of encryption speed, since one block cipher operation is needed in order to encrypt one bit. Moreover, it has been shown that reducing the number of rounds in order to increase encryption speed is not always a good idea [10].

The literature about SSSC is very limited compared to the literature about synchronous additive stream ciphers. However, general properties and design criteria for SSSCs have been studied by Maurer [8]. Besides, in [4], the authors discuss the design of SSSCs from an engineering point of view and propose KNOT as an example of such efficient ciphers. An improved version of KNOT, called $\Gamma\Upsilon$ was also proposed by Daemen in [3]. Unfortunately, the attacks we propose in this paper are based on specific properties of KNOT and hence do not apply to $\Gamma\Upsilon$. Therefore, we will focus only on KNOT in the following sections.

3 Description of KNOT

3.1 Overview of the Cipher

In [4], the authors propose to build efficient SSSCs in analogy with a block cipher function. In order to limit the gate delay during the computation of the keystream function, they suggest to use a construction with several simple rounds, similar to what is widely done in the context of block cipher design. Such a structure should also allow an efficient pipelining. For example, in KNOT, they propose to use a keystream function with 8 rounds that use only basic boolean operations. After each round, the size of the internal state decreases in order to eventually produce 1 output bit starting from the m input bits of the keystream function.

This multi-stage construction is chosen in order to protect the cipher against differential attacks. Hence, these stages must guarantee that no difference in the input of the keystream function will propagate to the outputs with non-negligible probability. These properties concerning difference propagation have been analyzed by the designers of KNOT. They claim that no difference pattern in the inputs of the keystream function will imply a difference pattern in the outputs with probability greater than 2^{-16}.

To improve confusion and diffusion properties on ciphertext and key bits, they propose to use a finite state machine in the upper stage instead of a simple shift register. In KNOT, this machine is based on the structure of a nonlinear register with a key-dependent evolution. This evolution mechanism has some special properties in order for the self-synchronization mechanism to work correctly. Furthermore, the ciphertext bit is introduced at different positions in this machine, in order to modify very quickly the internal state when a difference in the ciphertext is introduced. This machine is described more precisely in Sec-

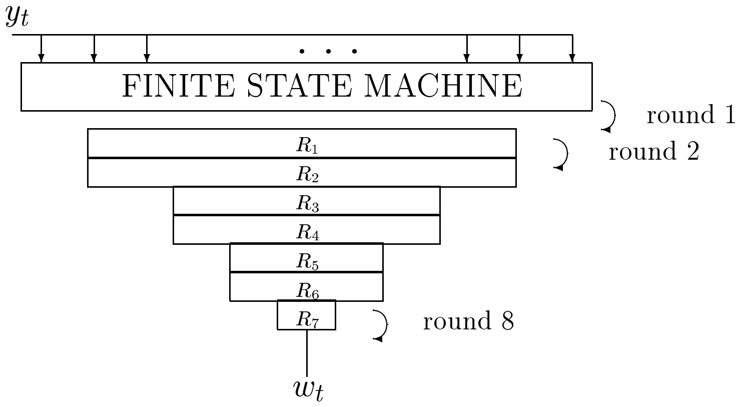

Fig. 2. The KNOT keystream function

tion 3.2. The general structure of KNOT keystream function can be represented as in Fig. 2.

3.2 The Finite State Machine

Let Q denote the finite state machine used in the keystream function of the cipher KNOT. Q has 128 memory cells which values depend on the 96-bit secret key K and the last 96 ciphertext bits introduced in the machine. Thus, the content of Q at time t can be expressed as

$$Q(t) = G(y_t, \ldots, y_{t-95} | k_0, \ldots, k_{95})$$

where y_t denotes the ciphertext bit at time t and k_i denotes the i-th key bit.

The structure of Q is based on a nonlinear shift register. Accordingly, Q may be seen as a 96-bit register where some of the memory cells have been duplicated. Hence, memory cells of Q are sorted into 96 sets denoted as $Q_1, \ldots, Q_{96}$ where each set Q_i may be seen as an extension of the i-th cell of the analog nonlinear shift register (see Fig. 3).

Each set Q_i contains n_i cells. Let $Q_{i,j}$ denote its j-th cell. Thus

$$Q_i = \{Q_{i,j}, 0 \leq i \leq n_i\}$$

If a standard shift register was used, a single digit difference in the input introduced at time $t - 95$ would imply only one differing cell at time t, that is the rightmost cell of the register. This might be a problem when considering differential attacks. This was avoided in KNOT by duplicating the rightmost cells in the register. Therefore Q can be seen as a nonlinear shift register where the rightmost cells have been expanded.

In analogy with a shift register, the value of each memory cell in Q is updated using only memory cells located on the left of this cell. More precisely, Q_i is

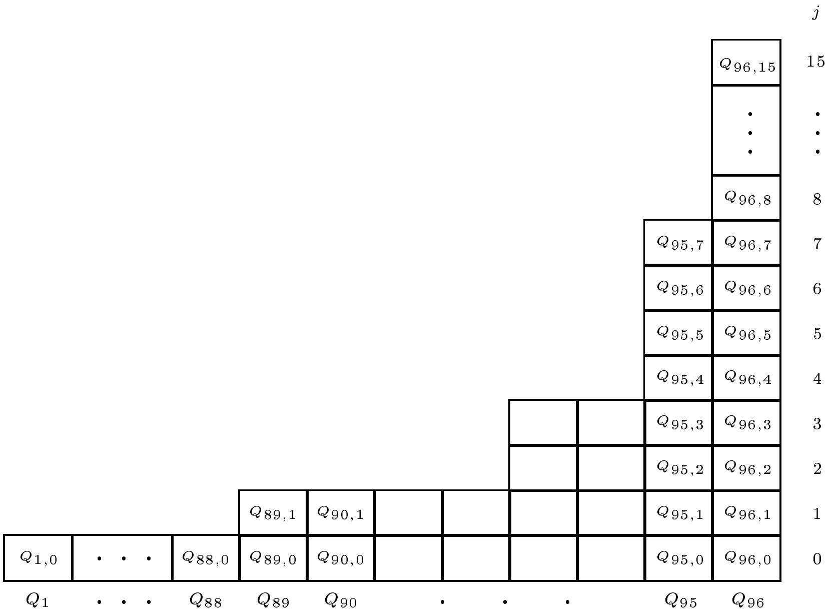

Fig. 3. The finite state machine Q with Expansion of the rightmost cells

updated at each instant using only the value of sets Q_j with $j < i$. This updating is actually key-dependent, which means the new value of Q_i is computed using also several secret key bits. Hence

$$Q_i(t) = H_i(Q_1(t-1), \dots, Q_{i-1}(t-1)|k_0, ..., k_{i-1})$$

Let $z_0^t, \dots, z_{95}^t$ denote the input bits of the keystream function at time t, where z_{95}^t is the oldest input and z_0^t is the latest input. These input bits are related to the ciphertext bits by

$$z_i^t = y_{t-i}$$

When no confusion is possible, we will get rid of the dependence on t and note $z_0, \dots, z_{95}$ these inputs. Thus, the content of the finite state machine Q can also be expressed as

$$Q(t) = G(z_0^t, \dots, z_{95}^t|k_0, \dots, k_{95})$$

In order to have a progressive elimination of oldest input bits, each Q_i does not actually depend on all input bits. In fact, the value of cells in Q_i depends only on

- The first i secret key bits, $k_0, \dots, k_{i-1}$
- The last i ciphertext bits introduced, $z_0, \dots, z_{i-1}$

This can be summarized by

$$Q_i(t) = G_i(z_0^t, \dots, z_{i-1}^t|k_0, \dots, k_{i-1})$$

To guarantee the propagation of differences in KNOT, these functions G_i were chosen of a particular form:

$$G_i(z_0^t, \ldots, z_{i-1}^t | k_0, \ldots, k_{i-1}) = z_{i-1}^t \oplus G_i'(z_0^t, \ldots, z_{i-2}^t | k_0, \ldots, k_{i-1})$$

Thus, if z_{i-1}^t is flipped, all cells in Q_i are also flipped. Actually, this property holds only for $i < 96$. The behavior of the update function G_{96} is different. Indeed, flipping z_{95} does not always cause cells of Q_{96} to flip as well. Thus different ciphertexts may yield the same internal state, which is an essential point of the key recovery attack we propose in this paper.

A precise description of the update function of Q is given in Table 1. The explicit value of each G_i function could be directly derived from this table. For each $Q_{i,j}$, the update function is denoted by

$$Q_{i,j}^t = f_i(a_i, b_i, c_i, d_i)$$

where f_i is a 4-entry boolean functions chosen among

$$g(a,b,c,d) = a + b + c(d+1) + 1$$
$$h(a,b,c,d) = a(b+1) + c(d+1)$$

3.3 The Pipelined Stages

When computing the new keystream bit using the keystream function, the content of Q is updated first. This is followed by 8 rounds of transformations in order to decrease the size of the internal state, until one output bit is produced. In KNOT, these round transformations are not key-dependent. Internal values can be represented by 7 registers $R_1, \ldots, R_7$ which value is computed from Q using the functions described in Table 2.

Update functions ψ and τ are defined by

$$\psi(X) = Y : y_i = g(x_{6i}, x_{6i+3}, x_{6i+1}, x_{6i+2})$$
$$\tau(X) = Y : y_i = g(x_{5i}, x_{5i+3}, x_{5i+1}, x_{5i+2})$$

where the indices of x are taken modulo the length of X and the boolean function g is the same as the one used in the update function of Q. The 8-th round consist in the computation of the keystream bit by

$$w_t = R_7(0) + R_7(1)(R_7(2) + 1) + 1$$

where $R_7(0)$, $R_7(1)$, $R_7(2)$ are the first 3 bits of register R_7.

These register are only used as intermediate values in the computation of the keystream bit. They cannot be considered part of the internal state of the cipher since they are not re-used afterwards. Therefore the effective size of the internal state of KNOT (excluding the secret key bits) is only 128 bits.

Table 1. The update function of Q

$cell$	f_i	a_i	b_i	c_i	d_i
$Q_{96,15}$	h	$Q_{95,7}$	$Q_{76,0}$	$Q_{94,3}$	$Q_{84,0}$
$Q_{96,14}$	h	$Q_{95,6}$	$Q_{75,0}$	$Q_{94,2}$	$Q_{83,0}$
$Q_{96,13}$	h	$Q_{95,5}$	$Q_{74,0}$	$Q_{94,1}$	$Q_{82,0}$
$Q_{96,12}$	h	$Q_{95,4}$	$Q_{73,0}$	$Q_{94,0}$	$Q_{81,0}$
$Q_{96,11}$	h	$Q_{95,3}$	$Q_{72,0}$	$Q_{94,3}$	$Q_{80,0}$
$Q_{96,10}$	h	$Q_{95,2}$	$Q_{71,0}$	$Q_{94,2}$	$Q_{79,0}$
$Q_{96,9}$	h	$Q_{95,1}$	$Q_{70,0}$	$Q_{94,1}$	$Q_{78,0}$
$Q_{96,8}$	h	$Q_{95,0}$	$Q_{69,0}$	$Q_{94,0}$	$Q_{77,0}$
$Q_{96,7}$	g	$Q_{95,7}$	k_{95}	$Q_{95,7}$	$Q_{93,3}$
$Q_{96,6}$	g	$Q_{95,6}$	k_{95}	$Q_{95,6}$	$Q_{93,2}$
$Q_{96,5}$	g	$Q_{95,5}$	k_{95}	$Q_{95,5}$	$Q_{93,1}$
$Q_{96,4}$	g	$Q_{95,4}$	k_{95}	$Q_{95,4}$	$Q_{93,0}$
$Q_{96,3}$	g	$Q_{95,3}$	k_{95}	$Q_{95,3}$	$Q_{91,1}$
$Q_{96,2}$	g	$Q_{95,2}$	k_{95}	$Q_{95,2}$	$Q_{91,0}$
$Q_{96,1}$	g	$Q_{95,1}$	k_{95}	$Q_{95,1}$	$Q_{90,1}$
$Q_{96,0}$	g	$Q_{95,0}$	k_{95}	$Q_{95,0}$	$Q_{90,0}$
$Q_{95,7}$	g	$Q_{94,3}$	$Q_{53,0}$	$Q_{93,3}$	$Q_{89,1}$
$Q_{95,6}$	g	$Q_{94,2}$	$Q_{47,0}$	$Q_{93,2}$	$Q_{89,0}$
$Q_{95,5}$	g	$Q_{94,1}$	$Q_{41,0}$	$Q_{93,1}$	$Q_{92,1}$
$Q_{95,4}$	g	$Q_{94,0}$	$Q_{35,0}$	$Q_{93,0}$	$Q_{92,0}$
$Q_{95,3}$	g	$Q_{94,3}$	k_{94}	$Q_{93,3}$	$Q_{85,0}$
$Q_{95,2}$	g	$Q_{94,2}$	k_{94}	$Q_{93,2}$	$Q_{88,0}$
$Q_{95,1}$	g	$Q_{94,1}$	k_{94}	$Q_{93,1}$	$Q_{87,0}$
$Q_{95,0}$	g	$Q_{94,0}$	k_{94}	$Q_{91,0}$	$Q_{93,0}$
$Q_{94,3}$	g	$Q_{93,3}$	k_{93}	$Q_{92,1}$	$Q_{86,0}$
$Q_{94,2}$	g	$Q_{93,2}$	k_{93}	$Q_{92,0}$	$Q_{88,0}$
$Q_{94,1}$	g	$Q_{93,1}$	k_{93}	$Q_{92,1}$	$Q_{85,0}$
$Q_{94,0}$	g	$Q_{93,0}$	k_{93}	y_t	$Q_{92,0}$
$Q_{93,3}$	g	$Q_{92,1}$	$Q_{65,0}$	$Q_{91,1}$	$Q_{89,1}$
$Q_{93,2}$	g	$Q_{92,0}$	$Q_{59,0}$	$Q_{91,0}$	$Q_{89,0}$
$Q_{93,1}$	g	$Q_{92,1}$	k_{92}	$Q_{91,1}$	$Q_{87,0}$
$Q_{93,0}$	g	$Q_{92,0}$	k_{92}	$Q_{87,0}$	$Q_{91,0}$
$Q_{92,1}$	g	$Q_{91,1}$	k_{91}	$Q_{90,0}$	$Q_{84,0}$
$Q_{92,0}$	g	$Q_{91,0}$	k_{91}	$Q_{88,0}$	$Q_{90,0}$
$Q_{91,1}$	g	$Q_{90,1}$	k_{90}	$Q_{89,0}$	$Q_{83,0}$
$Q_{91,0}$	g	$Q_{90,0}$	k_{90}	$Q_{86,0}$	y_t
$Q_{90,1}$	g	$Q_{89,1}$	k_{89}	$Q_{88,0}$	$Q_{82,0}$
$Q_{90,0}$	g	$Q_{89,0}$	k_{89}	$Q_{84,0}$	$Q_{88,0}$
$Q_{89,1}$	g	$Q_{88,0}$	$Q_{71,0}$	$Q_{87,0}$	$Q_{81,0}$
$Q_{89,0}$	g	$Q_{88,0}$	k_{88}	$Q_{85,0}$	$Q_{87,0}$
$Q_{88,0}$	g	$Q_{87,0}$	k_{87}	y_t	$Q_{86,0}$
$Q_{87,0}$	g	$Q_{86,0}$	k_{86}	$Q_{81,0}$	$Q_{85,0}$
$Q_{86,0}$	g	$Q_{85,0}$	k_{85}	$Q_{82,0}$	$Q_{84,0}$
$Q_{85,0}$	g	$Q_{84,0}$	k_{84}	$Q_{80,0}$	y_t
$Q_{84,0}$	g	$Q_{83,0}$	k_{83}	$Q_{78,0}$	$Q_{82,0}$
$Q_{83,0}$	g	$Q_{82,0}$	k_{82}	$Q_{79,0}$	$Q_{81,0}$
$Q_{82,0}$	g	$Q_{81,0}$	k_{81}	y_t	$Q_{80,0}$
$Q_{81,0}$	g	$Q_{80,0}$	k_{80}	$Q_{75,0}$	$Q_{79,0}$
$Q_{80,0}$	g	$Q_{79,0}$	k_{79}	$Q_{76,0}$	$Q_{78,0}$
$Q_{79,0}$	g	$Q_{78,0}$	k_{78}	$Q_{74,0}$	y_t
$Q_{78,0}$	g	$Q_{77,0}$	k_{77}	$Q_{72,0}$	$Q_{76,0}$
$Q_{77,0}$	g	$Q_{76,0}$	k_{76}	$Q_{73,0}$	$Q_{75,0}$
$Q_{76,0}$	g	$Q_{75,0}$	k_{75}	y_t	$Q_{74,0}$
$Q_{75,0}$	g	$Q_{74,0}$	k_{74}	$Q_{69,0}$	$Q_{73,0}$
$Q_{74,0}$	g	$Q_{73,0}$	k_{73}	$Q_{70,0}$	$Q_{72,0}$
$Q_{73,0}$	g	$Q_{72,0}$	k_{72}	$Q_{68,0}$	y_t
$Q_{72,0}$	g	$Q_{71,0}$	k_{71}	$Q_{66,0}$	$Q_{70,0}$
$Q_{71,0}$	g	$Q_{70,0}$	k_{70}	$Q_{67,0}$	$Q_{69,0}$
$Q_{70,0}$	g	$Q_{69,0}$	k_{69}	y_t	$Q_{68,0}$
$Q_{69,0}$	g	$Q_{68,0}$	k_{68}	$Q_{63,0}$	$Q_{67,0}$
$Q_{68,0}$	g	$Q_{67,0}$	k_{67}	$Q_{64,0}$	$Q_{66,0}$
$Q_{67,0}$	g	$Q_{66,0}$	k_{66}	$Q_{62,0}$	y_t
$Q_{66,0}$	g	$Q_{65,0}$	k_{65}	$Q_{60,0}$	$Q_{64,0}$
$Q_{65,0}$	g	$Q_{64,0}$	k_{64}	$Q_{61,0}$	$Q_{63,0}$
$Q_{64,0}$	g	$Q_{63,0}$	k_{63}	y_t	$Q_{62,0}$
$Q_{63,0}$	g	$Q_{62,0}$	k_{62}	$Q_{57,0}$	$Q_{61,0}$
$Q_{62,0}$	g	$Q_{61,0}$	k_{61}	$Q_{58,0}$	$Q_{60,0}$
$Q_{61,0}$	g	$Q_{60,0}$	k_{60}	$Q_{56,0}$	y_t
$Q_{60,0}$	g	$Q_{59,0}$	k_{59}	$Q_{54,0}$	$Q_{58,0}$
$Q_{59,0}$	g	$Q_{58,0}$	k_{58}	$Q_{55,0}$	$Q_{57,0}$
$Q_{58,0}$	g	$Q_{57,0}$	k_{57}	y_t	$Q_{56,0}$
$Q_{57,0}$	g	$Q_{56,0}$	k_{56}	$Q_{51,0}$	$Q_{55,0}$
$Q_{56,0}$	g	$Q_{55,0}$	k_{55}	$Q_{52,0}$	$Q_{54,0}$
$Q_{55,0}$	g	$Q_{54,0}$	k_{54}	$Q_{50,0}$	y_t
$Q_{54,0}$	g	$Q_{53,0}$	k_{53}	$Q_{48,0}$	$Q_{52,0}$
$Q_{53,0}$	g	$Q_{52,0}$	k_{52}	$Q_{49,0}$	$Q_{51,0}$
$Q_{52,0}$	g	$Q_{51,0}$	k_{51}	y_t	$Q_{50,0}$
$Q_{51,0}$	g	$Q_{50,0}$	k_{50}	$Q_{45,0}$	$Q_{49,0}$
$Q_{50,0}$	g	$Q_{49,0}$	k_{49}	$Q_{46,0}$	$Q_{48,0}$
$Q_{49,0}$	g	$Q_{48,0}$	k_{48}	$Q_{44,0}$	y_t
$Q_{48,0}$	g	$Q_{47,0}$	k_{47}	$Q_{42,0}$	$Q_{46,0}$
$Q_{47,0}$	g	$Q_{46,0}$	k_{46}	$Q_{43,0}$	$Q_{45,0}$
$Q_{46,0}$	g	$Q_{45,0}$	k_{45}	y_t	$Q_{44,0}$
$Q_{45,0}$	g	$Q_{44,0}$	k_{44}	$Q_{39,0}$	$Q_{43,0}$
$Q_{44,0}$	g	$Q_{43,0}$	k_{43}	$Q_{40,0}$	$Q_{42,0}$
$Q_{43,0}$	g	$Q_{42,0}$	k_{42}	$Q_{38,0}$	y_t
$Q_{42,0}$	g	$Q_{41,0}$	k_{41}	$Q_{36,0}$	$Q_{40,0}$
$Q_{41,0}$	g	$Q_{40,0}$	k_{40}	$Q_{37,0}$	$Q_{39,0}$
$Q_{40,0}$	g	$Q_{39,0}$	k_{39}	y_t	$Q_{38,0}$
$Q_{39,0}$	g	$Q_{38,0}$	k_{38}	$Q_{33,0}$	$Q_{37,0}$
$Q_{38,0}$	g	$Q_{37,0}$	k_{37}	$Q_{34,0}$	$Q_{36,0}$
$Q_{37,0}$	g	$Q_{36,0}$	k_{36}	$Q_{32,0}$	y_t
$Q_{36,0}$	g	$Q_{35,0}$	k_{35}	$Q_{30,0}$	$Q_{34,0}$
$Q_{35,0}$	g	$Q_{34,0}$	k_{34}	$Q_{31,0}$	$Q_{33,0}$
$Q_{34,0}$	g	$Q_{33,0}$	k_{33}	y_t	$Q_{32,0}$
$Q_{33,0}$	g	$Q_{32,0}$	k_{32}	$Q_{27,0}$	$Q_{31,0}$
$Q_{32,0}$	g	$Q_{31,0}$	k_{31}	$Q_{28,0}$	$Q_{30,0}$
$Q_{31,0}$	g	$Q_{30,0}$	k_{30}	$Q_{26,0}$	y_t
$Q_{30,0}$	g	$Q_{29,0}$	k_{29}	$Q_{24,0}$	$Q_{28,0}$
$Q_{29,0}$	g	$Q_{28,0}$	k_{28}	$Q_{25,0}$	$Q_{27,0}$
$Q_{28,0}$	g	$Q_{27,0}$	k_{27}	y_t	$Q_{26,0}$
$Q_{27,0}$	g	$Q_{26,0}$	k_{26}	$Q_{21,0}$	$Q_{25,0}$
$Q_{26,0}$	g	$Q_{25,0}$	k_{25}	$Q_{22,0}$	$Q_{24,0}$
$Q_{25,0}$	g	$Q_{24,0}$	k_{24}	$Q_{20,0}$	y_t
$Q_{24,0}$	g	$Q_{23,0}$	k_{23}	$Q_{18,0}$	$Q_{22,0}$
$Q_{23,0}$	g	$Q_{22,0}$	k_{22}	$Q_{19,0}$	$Q_{21,0}$
$Q_{22,0}$	g	$Q_{21,0}$	k_{21}	y_t	$Q_{20,0}$
$Q_{21,0}$	g	$Q_{20,0}$	k_{20}	$Q_{15,0}$	$Q_{19,0}$
$Q_{20,0}$	g	$Q_{19,0}$	k_{19}	$Q_{16,0}$	$Q_{18,0}$
$Q_{19,0}$	g	$Q_{18,0}$	k_{18}	$Q_{14,0}$	y_t
$Q_{18,0}$	g	$Q_{17,0}$	k_{17}	$Q_{12,0}$	$Q_{16,0}$
$Q_{17,0}$	g	$Q_{16,0}$	k_{16}	$Q_{13,0}$	$Q_{15,0}$
$Q_{16,0}$	g	$Q_{15,0}$	k_{15}	y_t	$Q_{14,0}$
$Q_{15,0}$	g	$Q_{14,0}$	k_{14}	$Q_{9,0}$	$Q_{13,0}$
$Q_{14,0}$	g	$Q_{13,0}$	k_{13}	$Q_{10,0}$	$Q_{12,0}$
$Q_{13,0}$	g	$Q_{12,0}$	k_{12}	$Q_{8,0}$	y_t
$Q_{12,0}$	g	$Q_{11,0}$	k_{11}	$Q_{6,0}$	$Q_{10,0}$
$Q_{11,0}$	g	$Q_{10,0}$	k_{10}	$Q_{7,0}$	$Q_{9,0}$
$Q_{10,0}$	g	$Q_{9,0}$	k_9	y_t	$Q_{8,0}$
$Q_{9,0}$	g	$Q_{8,0}$	k_8	$Q_{3,0}$	$Q_{7,0}$
$Q_{8,0}$	g	$Q_{7,0}$	k_7	$Q_{4,0}$	$Q_{6,0}$
$Q_{7,0}$	g	$Q_{6,0}$	k_6	$Q_{2,0}$	y_t
$Q_{6,0}$	g	$Q_{5,0}$	k_5	0	0
$Q_{5,0}$	g	$Q_{4,0}$	k_4	$Q_{1,0}$	$Q_{3,0}$
$Q_{4,0}$	g	$Q_{3,0}$	k_3	y_t	$Q_{2,0}$
$Q_{3,0}$	g	$Q_{2,0}$	k_2	0	0
$Q_{2,0}$	g	$Q_{1,0}$	k_1	0	0
$Q_{1,0}$	g	y_t	k_0	0	0

3.4 Known Weakness of KNOT

In his PhD thesis [3], Joan Daemen reported that "the output of KNOT has a detectable imbalance". We checked this bias experimentally and observed that the keystream produced for a uniformly distributed input is biased by $\varepsilon \simeq 2^{-9}$. Indeed, the probability of observing 1 as output is $\frac{1}{2}(1+\varepsilon)$.

This bias results from an imbalance in the last stages of the keystream function. More precisely, when expressing the output as a function of intermediate register R_5 as

$$w_t = \theta(R_5^t(0), \ldots, R_5^t(15))$$

where $R_5^t(i)$ is the i-th bit of register R_5 at time t, we observe that θ can be written as the exclusive or of 10 terms. The first term is the constant 1, the other 9 terms are polynomials of degree 2 or higher in the $R_5^t(i)$.

Therefore each term has probability $\frac{3}{4}$ to have value 1 and a bias of about 2^{-9} in the output is expected. By exhaustive search, we observed that the output is

Table 2. State-transition functions

Stage	*Length*	*Name*	*State-transition*
1	64	R_1	$\psi(Q)$
2	64	R_2	$\tau(R_1)$
3	32	R_3	$\psi(R_2)$
4	32	R_4	$\tau(R_3)$
5	16	R_5	$\psi(R_4)$
6	16	R_6	$\tau(R_5)$
7	8	R_7	$\psi(R_6)$

1 for 32832 out of the 65536 possible inputs of the function θ. Then assuming that the content of register R_5 is balanced, we expect a bias

$$\varepsilon = 2 \times \frac{32832}{65536} - 1 = \frac{1}{2^9}$$

This observation gives a distinguisher on the cipher requiring about $(2^9)^2 = 2^{18}$ pairs of plaintext-ciphertext bits.

4 A Key Recovery Attack

In this section, we propose a chosen ciphertext attack. The basic idea behind our attack is that introducing two different ciphertext sequences in the finite state machine Q may induce a collision with non-negligible probability on its whole internal state when the difference between these ciphertexts has a particular form. Therefore, equality of the keystream bits produced by these ciphertexts occurs with probability slightly bigger than $\frac{1}{2}$.

Thus, by observing the keystream produced for chosen ciphertexts, we are able to recover some secret information concerning the internal values of Q. Then, this information allows us to recover the secret key with less computation than an exhaustive search.

4.1 Obtaining Collisions on Q

In order to obtain collisions on the internal state of the finite state machine Q, we compute the value of the keystream function for two different inputs Z and $\hat{Z}$ of the form $Z = (z_0, \ldots, z_{94}, 0)$ and $\hat{Z} = (z_0, \ldots, z_{94}, 1)$. Concretely, this is done by introducing two sequences of 96 ciphertext bits in the finite state machine that only differ by the first bit introduced. After the 96-th bit is introduced, we observe the internal state of the system. These inputs yield two internal states we denote by Q and $\hat{Q}$. The two corresponding keystream bits produced are denoted by w and $\hat{w}$.

All sets Q_i can be computed by a function G_i of the form

$$Q_i = G_i(z_0, \ldots, z_{i-1} | k_0, \ldots, k_{i-1})$$

It appears that no set Q_i for $i \leq 95$ depends on z_{95}. Since all other input bits are the same in both executions, the only memory cells that might be different between Q and $\hat{Q}$ are the ones in Q_{96}.

Thus, to obtain a collision between Q and $\hat{Q}$, it suffices to find a collision between Q_{96} and $\hat{Q}_{96}$. The authors of [4] claimed that each set Q_i can be generally computed at time t as

$$Q_i^t = z_{i-1}^t \oplus G'_i(z_0^t, \ldots, z_{i-2}^t | k_0, \ldots, k_{i-1}) \tag{1}$$

Actually, we observed that this property does not hold for $i = 96$. More precisely, unlike all other update functions, the updating of Q_{96} does not depend linearly on Q_{95}. At time t, all cells of Q_{96} can be computed as:

$$Q_{96,i}^t = Q_{95,j}^{t-1}.X \oplus Y \text{ , } 0 \leq i \leq 15$$

for some j. X and Y are two binary values that depend only on the secret key and the content of $Q_0^{t-1}, \ldots, Q_{94}^{t-1}$. From (1), it follows that Q_{95}^{t-1} linearly depends on the flipped input bit z_{95}, while X and Y only depend on the other input bits $z_0, \ldots, z_{94}$.

Hence, in spite of the flipped input z_{95}, memory cells of Q_{96} may still have the same value, depending on the value of X and therefore on the value of the 95 other input bits. We assumed each event is balanced and occurs with probability $\frac{1}{2}$. Thus, each cell in Q_{96} is equal in both executions with probability $\frac{1}{2}$.

Since Q_{96} contains 16 cells, Q and $\hat{Q}$ are equal with probability

$$(\frac{1}{2})^{16} = \frac{1}{2^{16}}$$

The existence of such collisions implies that w and $\hat{w}$ will be equal with a bias $\varepsilon = 2^{-16}$. This observation gives a distinguisher on the cipher. However, collisions can be used more efficiently to recover some secret information about the internal state of Q.

4.2 Recovering Internal State Values

In the previous section, we showed how to obtain internal state collisions on Q for two different ciphertexts introduced. Such collisions occur at time t when the internal state of the finite state machine at time $t-1$ has some particular properties. Thus, using these collisions, we are able to recover some secret information about the internal state of Q. Observing the properties of memory cells in Q_{96} regarding to this, we divide them in two parts

- *First part:* $Q_{96,0}, \ldots, Q_{96,7}$
  The updating of these cells depends only on sets Q_i with $90 \leq i \leq 95$. These sets are located in the rightmost part of Q. Thus, the equality of these 8 cells in both executions at time t depends on all other input bits $z_0, \ldots, z_{94}$. We will assume that each of these events occur with probability $\frac{1}{2}$.

- *Second part:* $Q_{96,8}, \ldots, Q_{96,15}$
  The updating of these cells depends on Q_{95} and Q_{94}, but also on several Q_i located further on the left in Q. It can be shown that the equality for each of these 8 cells at time t depends respectively on the value of $Q_{69}, \ldots, Q_{76}$ at time $t-1$. More precisely, equalities occur when $Q_{69,0}^{t-1} = 1, \ldots, Q_{76,0}^{t-1} = 1$. These events only depend on input bits $z_1, \ldots, z_{76}$ and we will assume each equality occurs with probability $\frac{1}{2}$.

According to these observations, we fix the values of $z_1, \ldots, z_{76}$. With probability 2^{-8}, this will induce a collision on the Second Part of Q_{96}. Then, we try all possible values for the 19 remaining input bits (z_0 and $z_{77}, \ldots, z_{94}$). This does not change anything concerning the Second Part, but, in addition, we also get a collision on the First Part of Q_{96} with probability 2^{-8}.

Observing the number of equalities in the keystream bits produced, we should be able to observe the expected bias and thus to detect when a collision on Q occurs. Obtaining such a collision means that the values chosen for $z_1, \ldots, z_{76}$ yield value 1 for the cells of $Q_{69}, \ldots, Q_{76}$. Besides, relation (1) shows that these internal values only depend on the first 76 secret key bits.

Hence, by observing some internal collisions on Q, we recover the internal values of 8 cells in the finite state machine for known inputs. These values only depend on the first 76 key bits, which allows us to mount a key recovery attack.

4.3 Practical Realization of this Attack

In order to recover some secret information concerning the internal state of Q, we want to observe a bias $\varepsilon = 2^{-8}$ in the number of equal keystream bits produced. Besides, 19 ciphertext bits are randomized so we get a number of experiences $\Omega = 2^{19}$. Since

$$\Omega > \frac{1}{\varepsilon^2}$$

we generally consider that this bias can be efficiently detected.

Unfortunately, these Ω experiences are not independent. It appears that among the 19 ciphertext bits we randomize, 2 bits have no influence on the occurrence of internal state collisions:

- the latest input to the keystream function, z_0.
  This ciphertext bit is not immediately introduced in Q_{96}. Thus, collisions between Q and $\hat{Q}$ will occur or not, independently of the value of z_0.
- the second oldest input bit, z_{94}.
  Collisions on the First Part of Q_{96} at time t depend on the value of several internal values at time $t-1$ that do not actually depend on z_{94}. Thus this input bit has no effect on the occurrence of collisions.

Therefore, only 2^{17} independent experiences can be obtained. Randomizing z_0 and z_{94} will only cause some repeated experiences as regards to collisions. With only 2^{17} experiences, the statistical bound is very tight and it is likely that false alarms will be launched when trying to detect collisions.

However, the initial 2^{19} experiences can be grouped in 2^{17} sets of 4 repeated experiences, depending on the remaining 17 randomizable bits. If the fixed 76 ciphertext bits yield a collision in the Second Part of Q_{96}, a collision on the First Part of Q_{96} and hence on Q should occur with probability 2^{-8}. In case of full collision, 4 keystream equalities should be observed in the corresponding group of experiences. Otherwise, no internal state collisions are possible and we expect balanced results concerning keystream equalities. Denoting by 0 the case of keystream equality and 1 the other case, the following distribution is expected in each group:

Table 3. Expected Distributions

Events	$(0,0,0,0)$	$(0,0,0,1)$	...	$(1,1,1,1)$
Collisions	$\frac{1}{2^8}+(1-\frac{1}{2^8})\frac{1}{16}$	$(1-\frac{1}{2^8})\frac{1}{16}$	...	$(1-\frac{1}{2^8})\frac{1}{16}$
No Collision	$\frac{1}{16}$	$\frac{1}{16}$	...	$\frac{1}{16}$

The event (0,0,0,0) should be largely over-represented in case of a collision on the Second Part of Q_{96}. The bias is still about $\varepsilon = 2^{-8}$ and the number of possible tests is $\Omega = 2^{17}$. However, such a bias is easier to detect on a 16-valued distribution than on a bi-valued one.

Using this technique, internal state collisions on Q can be efficiently detected. Therefore, some secret information about the internal values of Q is recovered as described in the previous section.

4.4 Recovering the Key

The recovered values in the internal state yield some verifiable equations on key bits $k_0, \ldots, k_{75}$. Therefore, when enough equations of this type are obtained, an exhaustive search on these 76 key bits is possible, using these equations as a stopping condition on the search. Generally, it is necessary to obtain at least 76 equations for such an exhaustive search to work. However, each collision found as described in the previous section gives only 8 conditions on key bits. Thus, 10 collisions at least are needed.

Moreover, finding such a collision requires to run the keystream function with M chosen ciphertext bits, where

$$M = 2^{19} \times 2 \times 96 \times \frac{1}{p_{succ}}$$

and the probability of collision on the Second Part p_{succ} is roughly equal to 2^{-8}. Thus

$$M = 2^{34.6}$$

Hence the data complexity of our attack is $10 \times M = 2^{39}$. Basically, the exhaustive search on the 76 secret key bits has a complexity of 2^{76}. In fact, it is obviously better to verify the conditions on Q_{69} first in order to guess only 69 key bits. After the size of the key space is reduced using these conditions on Q_{69}, the other key bits can be successively guessed. Using this basic optimization, our attack has a time complexity of 2^{69}.

5 An Improved Key Recovery Attack

In this section, we present a way to make our attack more efficient. We describe how to obtain more useful information about the internal state of Q, in order to decrease the complexity of the exhaustive search on the secret key bits.

The basic idea is to recover information about cells located further on the left in Q, using the internal state collisions we found in the previous section. This information will depend on less key bits, which will make the exhaustive search faster. Basically, we are able to recover some internal values of $Q_{62,0}$ similarly to what we previously obtained on $Q_{69,0}, \ldots, Q_{76,0}$.

In the previous section, we showed how to detect when these 8 cells have simultaneously value 1. In fact, it is very easy to use this attack to observe when $Q_{69,0}$ has value 1 by randomizing 7 additional input bits $z_{70}, \ldots, z_{76}$ as well. Flipping these input bits does not change the value of $Q_{69,0}$.

If $Q_{69,0} = 1$, the octet $(Q_{69,0}, \ldots Q_{76,0})$ is $(1, \ldots, 1)$ once among the 2^7 experiences. Otherwise, this event never happens. Since it can be detected as previously, we finally learn the value of

$$Q_{69} = G_{69,0}(z_0, \ldots, z_{68} | k_0, \ldots, k_{68})$$

for any ciphertext. Each query to this function requires the same amount of data as the research of a collision. Therefore it requires $2^{34.6}$ chosen input bits to be introduced in the keystream function.

To obtain some information about the internal state, we observe

$$\begin{aligned} a_0 &= G_{69}(z_0, 0, z_2, \ldots, z_{68} | k_0, \ldots, k_{68}) \\ a_1 &= G_{69}(z_0, 1, z_2, \ldots, z_{68} | k_0, \ldots, k_{68}) \end{aligned}$$

Using various properties of the propagation of ciphertext differences in the finite state machine, similarly to what we did in the previous section, it can be shown that $a_0 \neq a_1$ at time t if and only if $Q_{63,0}^{t-1} = Q_{62,0}^{t-2} = 1$. This information can be used as a stopping condition in the exhaustive search of the first 62 key bits.

Only a few conditions of this type are needed in order to reduce the key space. Then additional conditions on $Q_{69,0}, \ldots, Q_{76,0}$ can be used to recover the correct key bits. The resulting attack has a data complexity of 2^{40} and a time complexity of 2^{62}.

6 Conclusion

In this paper, we present differential attacks on the stream cipher KNOT. The best attack has time complexity 2^{62} and requires the production of 2^{40} keystream bits. An open question is to analyze the asynchronous stream cipher $\mathit{\Gamma\Upsilon}$ proposed by Daemen as an improved version of KNOT in [3].

References

1. E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems. In A.J. Menezes and S.A. Vanstone, editors, *Advances in Cryptology – Crypto'90*, volume 537 of *Lectures Notes in Computer Science*, pages 2–21. Springer-Verlag, 1990. Extended Abstract.
2. E. Biham and A. Shamir. Differential cryptanalysis of the full 16-round DES. In E.F. Brickell, editor, *Advances in Cryptology – Crypto'92*, volume 740 of *Lectures Notes in Computer Science*, pages 487–496. Springer-Verlag, 1992.
3. J. Daemen. *Cipher and hash function design. Strategies based on linear and differential cryptanalysis*. PhD thesis, march 1995. Chapter 9.
4. J. Daemen, R. Govaerts, and J. Vandewalle. A practical approach to the design of high speed self-synchronizing stream ciphers. In *Singapore ICCS/ISITA '92*, pages 279–283. IEEE, 1992.
5. S. Fluhrer, I. Mantin, and A. Shamir. Weaknesses in the key scheduling algorithm of rc4. In S. Vaudenay and A.M. Youssef, editors, *Selected Areas in Cryptography – 2001*, volume 2259 of *Lectures Notes in Computer Science*, pages 1–24. Springer-Verlag, 2001.
6. B. Kaliski and Y.L. Yin. On differential and linear cryptanalysis of the rc5 encryption algorithm. In D. Coppersmith, editor, *Advances in Cryptology – Crypto'95*, volume 963 of *Lectures Notes in Computer Science*, pages 171–184. Springer, 1995.
7. L. Knudsen and W. Meier. Improved differential attacks on rc5. In N. Koblitz, editor, *Advances in Cryptology – Crypto'96*, volume 1109 of *Lectures Notes in Computer Science*. Springer, 1996.
8. U.M. Maurer. New approaches to the design of self-synchronizing stream ciphers. In D.W. Davies, editor, *Advances in Cryptology – Eurocrypt'91*, volume 547 of *Lectures Notes in Computer Science*, pages 458–471. Springer-Verlag, 1991.
9. National Bureau of Standard U.S. *DES modes of operation*, 1980.
10. B. Preneel, M. Nuttin, R. Rijmen, and J. Buelens. Cryptanalysis of the cfb mode of the des with a reduced number of rounds. In D.R. Stinson, editor, *Advances in Cryptology – Crypto'93*, volume 773 of *Lectures Notes in Computer Science*. Springer-Verlag, 1993.