

On Exploiting Adjacent Bits in NLS

Cameron McDonald and Philip Hawkes

Qualcomm Australia,
Level 3, 230 Victoria Rd,
Gladesville, NSW 2111, Australia.
{cameronm,phawkes}@qualcomm.com

Abstract. NLS is a stream cipher consisting of a non-linear feedback shift register (NFSR) and a non-linear filter (NLF). This paper presents a linear distinguishing attack on NLS using an improved version of the Crossword Puzzle (CP) attack [2] by exploiting the internal dependencies between the NFSR and NLF. We derive significantly higher bias linear approximations of the NFSR and NLF using linear combinations of adjacent bits.

1 Introduction

The eSTREAM project [6] is designed to identify new stream ciphers that may be suitable for widespread adoption as secure and efficient cryptographic primitives. NLS is one of the ciphers submitted to the project and has successfully passed phase 1 of the review process. The security supplied by NLS is provided by the combination of a non-linear feedback shift register (NFSR) coupled with a non-linear filter (NLF). In [2], Cho and Pieprzyk present a new form of attack, coined the “Crossword Puzzle” (CP) attack, which is a linear distinguishing attack. They apply their attack to NLS to produce a distinguisher with average bias around $O(2^{-30})$.

We give a brief definition of the NLS cipher, followed by a description of the CP attack and how it is applied to NLS. We highlight the specific components of NLS that are exploited by the CP attack and how they interact within the cipher. Using linear approximations based on adjacent bits, we build a distinguisher for the case $Konst = 0$ (key-dependent constant word) with overall bias $2^{-19.7}$. This is an improvement on the distinguisher found in [2], which has bias $2^{-26.3}$. Note that there is a new version of NLS (Version 2), the only modification being that $Konst$ is changed after every 65537 clocks. This modification is designed to resist this attack.

2 Brief Description of NLS

We give a brief description of the NLS cipher which is sufficient for the following analysis. For a thorough description, see [4]. The stream generator of NLS is constructed from a non-linear feedback shift register (NFSR) and a non-linear

filter (NLF). These two functions operate on an internal state, a vector of 32-bit words $\sigma_t = (r_t[0], \dots, r_t[16])$, where t is the clock time and σ_0 is the initial state. The NFSR, which transforms state σ_t into state σ_{t+1} , operates as follows:

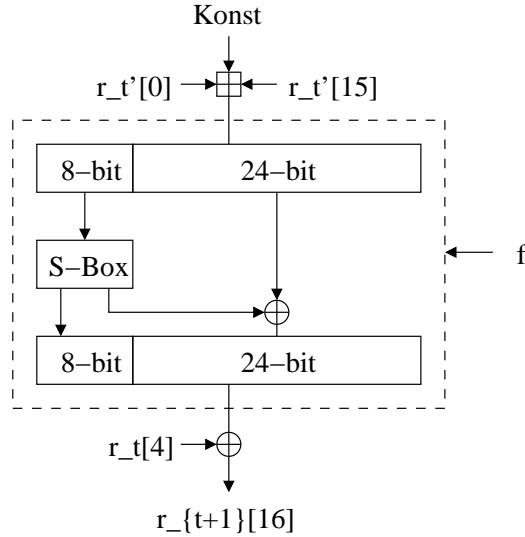
1. $r_{t+1}[i] = r_t[i + 1]$, for $i = 0, \dots, 15$;
2. $r_{t+1}[16] = f((r_t[0] \lll 19) + (r_t[15] \lll 9) + Konst) \oplus r_t[4]$;
3. $r_t[0]$ is abandoned;
4. if $t \equiv 0(\text{modulo } f16)$, $r_{t+1}[2]$ is modified by adding $t(\text{modulo } 2^{32})$,

where $Konst$ is a 32-bit key dependent constant. The output stream is generated 32-bits at a time by the NLF using the following combination of registers of the internal state:

$$v_t = (r_t[0] + r_t[16]) \oplus (r_t[1] + r_t[13]) \oplus (r_t[6] + Konst).$$

2.1 f function

The nonlinear function f combines XOR with a SBox, graphically it has the following representation:



The SBox is an 8-bit to 32-bit mapping, which is a combination of the Skipjack SBox and the ISRC SBox designed by the Queensland University of Technology. Let x_t be the input of the f function, that is $x_t = (r_t[0] \lll 19) + (r_t[15] \lll 9)$. The input to the SBox is the 8 most significant bits of x_t , which we denote x_t^H . Observe that the output to the f function, denoted y_t , can be defined by 2 cases:

$$y_{t,(i)} = \begin{cases} SBox(x_t^H)_{(i)} \oplus x_{(i)}, & \text{if } 0 \leq i < 24, \\ SBox(x_t^H)_{(i)}, & \text{if } 24 \leq i < 32, \end{cases}$$

where the i -th bit of variable x is denoted by $x_{(i)}$. Thus, we can write

$$r_{t+1}[16]_{(i)} = \begin{cases} r_t[4]_{(i)} \oplus y_{t,(i)} \oplus x_{t,(i)}, & \text{if } 0 \leq i < 24, \\ r_t[4]_{(i)} \oplus y_{t,(i)}, & \text{if } 24 \leq i < 32. \end{cases}$$

3 Crossword Puzzle (CP) Attack

The CP attack is a linear distinguishing attack which is an extension of the linear masking method introduced by Coppersmith, Halevi and Jutla [3]. The linear masking method was originally designed to examine traditional linear feedback shift register (LFSR) stream ciphers, whereas the CP attack can be applied to stream ciphers that consist of a non-linear feedback shift register (NFSR) and a non-linear filter (NLF). The foundation of the attack comprises of deriving linear approximations of the NFSR and NLF at different clock times, then combining these approximations in such a way that the internal state bits of the NFSR are cancelled out. We are then left with an approximation involving NLF output bits only, which can be used to launch a distinguishing attack provided the bias of the approximation is significantly high.

In [2], the authors derive approximations for the NFSR and NLF separately and then apply the CP attack to produce the distinguisher. They treat the approximations to the NFSR and NLF as independent relations and calculate the total bias using the Piling Up lemma [5]. In this article we modify the way the CP attack is applied to NLS to exploit some of the dependencies that exists between the internal state registers used in the NFSR and NLF. To initiate the motivation for the following study we introduce some of the relationships that are used in the CP attack. Observe the following two NFSR equations ($Konst = 0$), at clock times t and $(t + 16)$ respectively:

$$\begin{aligned} 0 &= f((r_t[0] \lll 19) + (r_t[15] \lll 9)) \oplus r_t[4] \oplus r_{t+1}[16], \\ 0 &= f((r_{t+16}[0] \lll 19) + (r_{t+16}[15] \lll 9)) \oplus r_{t+16}[4] \oplus r_{t+17}[16]. \end{aligned}$$

Using the fact that $r_{t+p}[0] = r_t[p]$, we can rewrite the above equations as:

$$\begin{aligned} 0 &= f((r_t[0] \lll 19) + (r_{t+15}[0] \lll 9)) \oplus r_{t+4}[0] \oplus r_{t+17}[0], \\ 0 &= f((r_t[16] \lll 19) + (r_{t+15}[16] \lll 9)) \oplus r_{t+4}[16] \oplus r_{t+1}[16]. \end{aligned}$$

We see that the inputs to the SBox function are:

$$(r_t[0] \lll 19) + (r_{t+15}[0] \lll 9) \text{ and } (r_t[16] \lll 19) + (r_{t+15}[16] \lll 9).$$

Now observe the relationship between the input and output of the NLF at clock times t and $(t + 15)$:

$$\begin{aligned} 0 &= (r_t[0] + r_t[16]) \oplus (r_t[1] + r_t[13]) \oplus r_t[6] \oplus v_t, \\ 0 &= (r_{t+15}[0] + r_{t+15}[16]) \oplus (r_{t+15}[1] + r_{t+15}[13]) \oplus r_{t+15}[6] \oplus v_{t+15}. \end{aligned}$$

Part of the CP attack involves finding a high bias approximation for f and joining the above two sets of equations to cancel out the state registers, leaving

v_t 's only. However, note that the two sets of approximations contain the same state registers $(r_t[0], r_t[16], r_{t+15}[0], r_{t+15}[16])$, and hence are not independent. This implies that the Piling Up lemma used in [2] does not give an accurate result in calculating the bias. In this paper we modify the application of the CP attack to exploit this interaction by developing the required equations as much as possible before assigning linear approximations. This is in contrast to the method used in [2], where the authors approximate first and then derive the equations.

4 Modular Addition and Adjacent Bits

The observation made in section §3 suggests there is a method of approximating the specific NLS additions mentioned that result in a higher bias. Upon closer examination, the additions that are exploited by the attack involve the addition of 4 state registers in the following manner:

$$(r'[a] + r''[b]) \oplus (r'[c] + r''[d]) \oplus (r[a] + r[c]) \oplus (r[b] + r[d]) = 0,$$

where $r'[i] = (r[i] \lll 19)$ and $r''[i] = (r[i] \lll 9)$ (this notation will be used throughout the paper). For example, in our analysis of the cipher, we examine the approximation:

$$(r'_t[0] + r''_{t+15}[0]) \oplus (r'_t[16] + r''_{t+15}[16]) \oplus (r_t[0] + r_t[16]) \oplus (r_{t+15}[0] + r_{t+15}[16]) = 0.$$

Similarly to [2], we investigate approximations that involve linear combinations of adjacent bits of variables. By using adjacent bit combinations we can reduce the affect that carry bits have on the overall approximation, as this leads to higher bias approximations. Throughout the following analysis we focus on linear approximations based on adjacent bits of input and output variables. To simplify our notation in the following analysis, we denote the i -th bit of variable x by $x_{(i)}$. Also, we use the following notation to denote the XOR of adjacent bits of x :

$$x_{(i+k,i)} = x_{(i+k)} \oplus \dots \oplus x_{(i+1)} \oplus x_{(i)}.$$

5 Linear Analysis

In this section we examine the different components of NLS and state linear approximations describing each part. Firstly, we analyse the SBox and produce linear approximations that have significantly high bias. Throughout the following analysis we examine the case where $Konst = 0$, this restriction can be dropped as is done in [2].

5.1 SBox

We observe different adjacent bit linear approximations involving the input and output of the f function and list ones that have significantly high bias in Table

1. Note that the linear properties of the Skipjack SBox (called "F-Table" in the original definition) have been previously studied by Biham et al. [1]. Their analysis accounts for linear approximations involving the 8 most significant bits of output only, as this is precisely the Skipjack SBox.

Max Adjacent Bit Length	Linear Approximation of SBox	Bias
2	$x_{t,(30,29)} \oplus y_{t,(30,29)}$	$2^{-2.29956}$
2	$x_{t,(25,24)} \oplus y_{t,(1,0)}$	$2^{-2.67807}$
2	$x_{t,(25,24)} \oplus y_{t,(16,15)}$	$2^{-2.67807}$
2	$x_{t,(30,29)} \oplus x_{t,(26,25)} \oplus y_{t,(19,18)} \oplus y_{t,(16,15)} \oplus y_{t,(7,6)} \oplus y_{t,(3,2)}$	$2^{-1.75207}$
4	$x_{t,(31,30)} \oplus x_{t,(27,26)} \oplus y_{t,(18,15)} \oplus y_{t,(12,9)} \oplus y_{t,(6,5)} \oplus y_{t,(3,2)}$	$2^{-1.60768}$
6	$x_{t,(30,29)} \oplus y_{t,(22,17)} \oplus y_{t,(14,9)} \oplus y_{t,(4,3)}$	$2^{-1.67807}$

Table 1. Linear Approximations of SBox

5.2 NFSR

From the definition of the f function, we can build the following description of the NFSR which holds with probability 1 for all i :

$$y_{t,(i)} = \begin{cases} r_t[4]_{(i)} \oplus r_{t+1}[16]_{(i)} \oplus x_{t,(i)}, & \text{if } 0 \leq i < 24, \\ r_t[4]_{(i)} \oplus r_{t+1}[16]_{(i)}, & \text{if } 24 \leq i < 32. \end{cases}$$

Following from this, we obtain with probability 1:

$$y_{t,(30,29)} \oplus r_t[4]_{(30,29)} \oplus r_{t+1}[16]_{(30,29)} = 0. \quad (1)$$

Using (1) and row 1 from Table 1 as a linear approximation to the SBox,

$$x_{t,(30,29)} \oplus y_{t,(30,29)},$$

we can build a linear approximation for the complete NFSR:

$$x_{t,(30,29)} \oplus r_t[4]_{(30,29)} \oplus r_{t+1}[16]_{(30,29)} = 0, \quad (2)$$

where $x_t = (r_t[0] \lll 19) + (r_t[15] \lll 9)$. This equation holds with bias $2^{-2.30} \stackrel{\text{def}}{=} bias_1$. We have chosen this particular linear approximation to the SBox as it yields the highest overall bias for the distinguisher.

Note that if we used a linear approximation that involves outputs from the 24 least significant bits, for example row 2 from Table 1, we would obtain the following approximation for the NFSR:

$$x_{t,(25,24)} \oplus x_{t,(1,0)} \oplus r_t[4]_{(1,0)} \oplus r_{t+1}[16]_{(1,0)} = 0.$$

This approximation holds true with bias $2^{-2.67}$.

5.3 Modular Addition

Modular addition is used throughout the structure of NLS, providing non-linearity in both the NFSR and the NLF. We analyse the modular addition of variables and derive relationships that allow us to build high bias linear approximations between particular state bits.

Firstly, we approximate the modular addition of two state registers $r[a]$ and $r[b]$ by a corresponding XOR only equation, drawing a comparison between the XOR of adjacent bits:

$$(r[a] + r[b])_{(i+k,i)} \oplus r[a]_{(i+k,i)} \oplus r[b]_{(i+k,i)} = 0. \quad (3)$$

It can be shown, that for even k , this equation has bias $2^{(-k/2)}$ for $0 \leq i < n - k$, see [2].

Consider the special addition of four variables $r[a]$, $r[b]$, $r[c]$ and $r[d]$ introduced in section §4:

$$\begin{aligned} & (r'[a] + r''[b])_{(i+k,i)} \oplus (r'[c] + r''[d])_{(i+k,i)} \oplus \\ & (r[a] + r[c])_{(i+k-19,i-19)} \oplus (r[b] + r[d])_{(i+k-9,i-9)} = 0, \end{aligned} \quad (4)$$

where all bit positions are taken modulo 32.

First, we examine the case for 2 adjacent bits ($k = 1$), followed by the case for 4 adjacent bits ($k = 3$). Tables 2 and 3 state the bias for these approximations for different starting bit i . We have also included the corresponding graphs of our analysis to highlight that the bias for 2 adjacent bits approaches $1/3 = 2^{1.58}$ and the bias for 4 adjacent bits approaches $1/8 = 2^{-3}$. There are three points of inconsistency ($i = 0, 9, 19$) on both graphs which occur when the analysis of (4) involves the the two least significant bits of a term. The method for finding these biases is discussed in the appendix.

i	0	1	2	3	4	5	6	7	8
bias	$2^{-15.02}$	$2^{-2.19}$	$2^{-1.73}$	$2^{-1.62}$	$2^{-1.59}$	$2^{-1.59}$	$2^{-1.59}$	$2^{-1.59}$	$2^{-1.58}$
i	9	10	11	12	13	14	15	16	17-18
bias	$2^{-2.26}$	$2^{-1.75}$	$2^{-1.63}$	$2^{-1.60}$	$2^{-1.59}$	$2^{-1.59}$	$2^{-1.59}$	$2^{-1.59}$	$2^{-1.58}$
i	19	20	21	22	23	24	25	26	27-30
bias	$2^{-2.26}$	$2^{-1.75}$	$2^{-1.63}$	$2^{-1.60}$	$2^{-1.59}$	$2^{-1.59}$	$2^{-1.59}$	$2^{-1.59}$	$2^{-1.58}$

Table 2. Bias for 2 adjacent bits of approximation (4)

For example, in our analysis we use the following approximation, which has bias $2^{-1.58}$:

$$\begin{aligned} & (r'_t[0] + r''_t[15])_{(30,29)} \oplus (r'_{t+16}[0] + r''_{t+16}[15])_{(30,29)} \\ & \oplus (r_t[0] + r_{t+16}[0])_{(11,10)} \oplus (r_t[15] + r_{t+16}[15])_{(21,20)} = 0. \end{aligned}$$

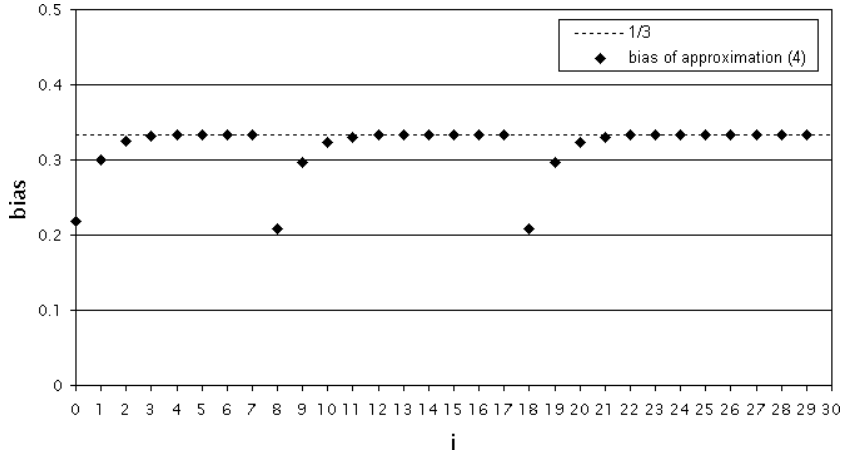


Fig. 1. Bias for 2 adjacent bits of approximation (4).

i	0	1	2	3	4	5	6	7	8
bias	$2^{-6.42}$	$2^{-3.90}$	$2^{-3.22}$	$2^{-3.06}$	$2^{-3.01}$	$2^{-3.00}$	$2^{-3.00}$	$2^{-3.00}$	$2^{-3.00}$
i	9	10	11	12	13	14	15	16	17-18
bias	$2^{-3.98}$	$2^{-3.24}$	$2^{-3.06}$	$2^{-3.02}$	$2^{-3.00}$	$2^{-3.00}$	$2^{-3.00}$	$2^{-3.00}$	$2^{-3.00}$
i	19	20	21	22	23	24	25	26	27-28
bias	$2^{-3.99}$	$2^{-3.25}$	$2^{-3.06}$	$2^{-3.02}$	$2^{-3.00}$	$2^{-3.00}$	$2^{-3.00}$	$2^{-3.00}$	$2^{-3.00}$

Table 3. Bias for 4 adjacent bits of approximation (4)

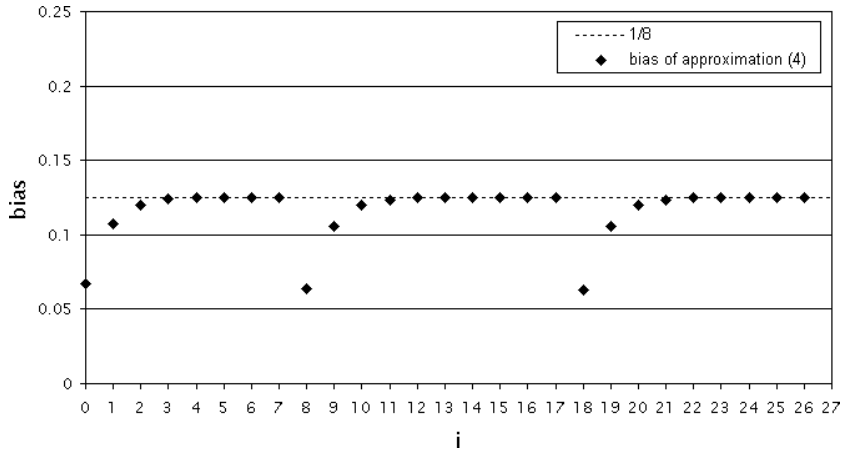


Fig. 2. Bias for 4 adjacent bits of approximation (4).

5.4 NLF

We examine the output function at particular clock times and rewrite the relationships in such a way that helps simplify our analysis later on. The following

relationships all hold with probability 1. We first observe the output functions at time $(t + 4)$ and $(t + 17)$:

$$v_{t+4} = (r_{t+4}[0] + r_{t+4}[16]) \oplus (r_{t+4}[1] + r_{t+4}[13]) \oplus r_{t+4}[6],$$

$$v_{t+17} = (r_{t+17}[0] + r_{t+17}[16]) \oplus (r_{t+17}[1] + r_{t+17}[13]) \oplus r_{t+17}[6].$$

Consider the XOR of k adjacent bits from i to $i + k$ of these equations:

$$\begin{aligned} & v_{t+4,(i+k,i)} \oplus v_{t+17,(i+k,i)} \\ &= (r_{t+4}[0] + r_{t+4}[16])_{(i+k,i)} \oplus (r_{t+4}[1] + r_{t+4}[13])_{(i+k,i)} \oplus r_{t+4}[6]_{(i+k,i)} \oplus \\ & \quad (r_{t+17}[0] + r_{t+17}[16])_{(i+k,i)} \oplus (r_{t+17}[1] + r_{t+17}[13])_{(i+k,i)} \oplus r_{t+17}[6]_{(i+k,i)}. \end{aligned}$$

Using the fact that $r_{t+p}[0] = r_t[p]$ for all p , we get

$$\begin{aligned} & v_{t+4,(i+k,i)} \oplus v_{t+17,(i+k,i)} \\ &= (r_t[4] + r_{t+16}[4])_{(i+k,i)} \oplus (r_{t+1}[4] + r_{t+13}[4])_{(i+k,i)} \oplus r_{t+6}[4]_{(i+k,i)} \oplus \quad (5) \\ & \quad (r_{t+1}[16] + r_{t+17}[16])_{(i+k,i)} \oplus (r_{t+2}[16] + r_{t+14}[16])_{(i+k,i)} \oplus r_{t+7}[16]_{(i+k,i)}. \end{aligned}$$

The output function at clock times t and $t + 15$ can be expressed similarly:

$$\begin{aligned} & v_{t,(i+k,i)} \oplus v_{t+15,(i+k,i)} \\ &= (r_t[0] + r_{t+16}[0])_{(i+k,i)} \oplus (r_{t+1}[0] + r_{t+13}[0])_{(i+k,i)} \oplus r_{t+6}[0]_{(i+k,i)} \oplus \quad (6) \\ & \quad (r_t[15] + r_{t+16}[15])_{(i+k,i)} \oplus (r_{t+1}[15] + r_{t+13}[15])_{(i+k,i)} \oplus r_{t+6}[15]_{(i+k,i)}. \end{aligned}$$

6 Distinguishing Attack on NLS

We use the same method as [2] to build our distinguisher for NLS. By observing clock times $t, t + 1, t + 6, t + 13$ and $t + 16$ we can use our approximation of the NFSR (2) to form the following system of equations, each having bias $2^{-2.30}$.

$$\begin{aligned} x_{t,(30,29)} \oplus r_t[4]_{(30,29)} \oplus r_{t+1}[16]_{(30,29)} &= 0, \\ x_{t+1,(30,29)} \oplus r_{t+1}[4]_{(30,29)} \oplus r_{t+2}[16]_{(30,29)} &= 0, \\ x_{t+6,(30,29)} \oplus r_{t+6}[4]_{(30,29)} \oplus r_{t+7}[16]_{(30,29)} &= 0, \quad (7) \\ x_{t+13,(30,29)} \oplus r_{t+13}[4]_{(30,29)} \oplus r_{t+14}[16]_{(30,29)} &= 0, \\ x_{t+16,(30,29)} \oplus r_{t+16}[4]_{(30,29)} \oplus r_{t+17}[16]_{(30,29)} &= 0. \end{aligned}$$

As in [2] the distinguisher is formed by adding these 5 equations together. We consider this as two separate problems, the first involves the XOR of the first term from each equation in (7), the second involves the XOR of the second and third terms from equation (7).

6.1 Combining 1st Terms

Observe from the definition of the f function:

$$x_{t,(30,29)} \oplus x_{t+16,(30,29)} \oplus (r'_t[0] + r''_t[15])_{(30,29)} \oplus (r'_{t+16}[0] + r''_{t+16}[15])_{(30,29)} = 0. \quad (8)$$

Taking into account rotations, we can use (4) to approximate (8),

$$x_{t,(30,29)} \oplus x_{t+16,(30,29)} \oplus (r_t[0] + r_{t+16}[0])_{(11,10)} \oplus (r_t[15] + r_{t+16}[15])_{(21,20)} = 0, \quad (9)$$

which has bias $2^{-1.58}$ (as shown in Table 2). Similarly, the approximation:

$$x_{t+1,(30,29)} \oplus x_{t+13,(30,29)} \oplus (r_{t+1}[0] + r_{t+13}[0])_{(11,10)} \oplus (r_{t+1}[15] + r_{t+13}[15])_{(21,20)} = 0, \quad (10)$$

has bias $2^{-1.58}$.

Also, observe that with probability 1:

$$x_{t+6,(30,29)} \oplus (r'_{t+6}[0] + r''_{t+6}[15])_{(30,29)} = 0. \quad (11)$$

Using (3), we can approximate (11) to get,

$$x_{t+6,(30,29)} \oplus r'_{t+6}[0]_{(30,29)} \oplus r''_{t+6}[15]_{(30,29)} = 0,$$

which has bias 2^{-1} . Removing the rotations, we achieve:

$$x_{t+6,(30,29)} \oplus r_{t+6}[0]_{(11,10)} \oplus r_{t+6}[15]_{(21,20)} = 0. \quad (12)$$

Combining approximations (6), (9), (10) and (12) we have the following approximation:

$$\begin{aligned} & x_{t,(30,29)} \oplus x_{t+1,(30,29)} \oplus x_{t+6,(30,29)} \oplus x_{t+13,(30,29)} \oplus x_{t+16,(30,29)} \oplus \\ & v_{t,(11,10)} \oplus v_{t+15,(21,20)} = 0, \end{aligned} \quad (13)$$

which has total bias $(2^{-1.58})^2 \cdot 2^{-1} = 2^{-4.16} \stackrel{\text{def}}{=} \text{bias}_2$. This approximation accounts for the addition of the first term of each approximation in equation (7).

6.2 Combining 2nd and 3rd Terms

The addition of the last two terms can be approximated similarly. Using approximation (3) twice, we get the approximation to the XOR of the 2nd terms of equation (7):

$$\begin{aligned} & r_t[4]_{(30,29)} \oplus r_{t+1}[4]_{(30,29)} \oplus r_{t+6}[4]_{(30,29)} \oplus r_{t+13}[4]_{(30,29)} \oplus r_{t+16}[4]_{(30,29)} \oplus \\ & (r_t[4] + r_{t+16}[4])_{(30,29)} \oplus (r_{t+1}[4] + r_{t+13}[4])_{(30,29)} \oplus r_{t+6}[4]_{(30,29)} = 0, \end{aligned}$$

which has bias $(2^{-1})^2$. Similarly, we get the approximation to the XOR of the 3rd terms of equation (7):

$$\begin{aligned} & r_{t+1}[16]_{(30,29)} \oplus r_{t+2}[16]_{(30,29)} \oplus r_{t+7}[16]_{(30,29)} \oplus r_{t+14}[16]_{(30,29)} \oplus r_{t+17}[16]_{(30,29)} \\ & (r_{t+1}[16] + r_{t+17}[16])_{(30,29)} \oplus (r_{t+2}[16] + r_{t+14}[16])_{(30,29)} \oplus r_{t+7}[16]_{(30,29)} = 0, \end{aligned}$$

which has bias $(2^{-1})^2$. Using (5) and the above two approximations we get an approximation for the XOR of the 2nd and 3rd terms in (7):

$$\begin{aligned} & r_t[4]_{(30,29)} \oplus r_{t+1}[4]_{(30,29)} \oplus r_{t+6}[4]_{(30,29)} \oplus r_{t+13}[4]_{(30,29)} \oplus r_{t+16}[4]_{(30,29)} \oplus \\ & r_{t+1}[16]_{(30,29)} \oplus r_{t+2}[16]_{(30,29)} \oplus r_{t+7}[16]_{(30,29)} \oplus r_{t+14}[16]_{(30,29)} \oplus r_{t+17}[16]_{(30,29)} \oplus \\ & v_{t+4,(30,29)} \oplus v_{t+17,(30,29)} = 0, \end{aligned} \quad (14)$$

which has bias $(2^{-1})^4 \stackrel{\text{def}}{=} bias_3$.

Combining all terms.

Substituting (13) and (14) into (7) gives the final distinguisher:

$$v_{t,(11,10)} \oplus v_{t+15,(21,20)} \oplus v_{t+4,(30,29)} \oplus v_{t+17,(30,29)} = 0.$$

The total bias for the distinguisher is $(bias_1)^5 \cdot bias_2 \cdot bias_3 = (2^{-2.30})^5 \cdot 2^{-4.16} \cdot (2^{-1})^4 = 2^{-19.7}$. This is the highest bias found in our analysis. Table 4 lists biases of other distinguishers that were found to be significant, and the contribution given by the sbox and modular additions:

x mask	y mask	SBox	2-bit adjacent	4-bit adjacent	Total Bias
0x60	0x60000000	$(2^{-2.30})^5$	$2^{-8.2}$	-	$2^{-19.7}$
0x03	0x00018000	$(2^{-2.68})^5$	$2^{-12.33}$	-	$2^{-25.73}$
0x03	0x00003000	$(2^{-2.68})^5$	$2^{-12.36}$	-	$2^{-25.75}$
0xC0	0x00006000	$(2^{-2.83})^5$	$2^{-12.35}$	-	$2^{-26.50}$
0x0C	0x00000018	$(2^{-2.83})^5$	$2^{-12.42}$	-	$2^{-26.57}$
0x0F	0x0000000C	$(2^{-2.54})^5$	$2^{-8.47}$	$2^{-8.00}$	$2^{-29.17}$
0x78	0x00000006	$(2^{-2.68})^5$	$2^{-9.39}$	$2^{-8.00}$	$2^{-30.76}$

Table 4. Different Distinguisher Biases

7 Conclusion

We have modified the application of the Crossword Puzzle attack on the NLS stream cipher to further exploit the interaction between the NFSR and NLF. Our analysis found that linear approximations of the components had significantly higher bias when the linear combinations involved adjacent bits. Using this we are able to build a linear distinguisher for NLS ($Konst = 0$) with overall bias $2^{-19.7}$. Therefore it is possible to distinguish the NLS stream from a random stream after approximately 2^{40} keystream words. This is an improvement on the results claimed in [2]. Note that this can not be applied to NLSv2 because $Konst$ changes.

We have also observed that additional dependencies exist between observations of the keystream at different time intervals. These dependencies occur because the same state registers will be contained within the distinguisher at different observation times. This observation is currently being studied and results will be produced shortly.

References

1. Eli Biham, Alex Biryukov, Orr Dunkelman, Eran Richardson and Adi Shamir, *Initial Observations on Skipjack: Cryptanalysis of Skipjack-3XOR*, Proceedings of Selected Areas in Cryptography, SAC'98, Lecture Notes in Computer Science, 1998.
2. Joo Yeon Cho and Josef Pieprzyk, *Crossword Puzzle Attack on NLS* Cryptology ePrint Archive, Report 2006/049, 2006.
3. Don Coppersmith, Shai Halevi and Charanjit Jutla, *Cryptanalysis of stream ciphers with linear masking* Cryptology ePrint Archive, Report 2002/02, 2002.
4. Philip Hawkes, Michael Paddon, Gregory G. Rose, Miriam Wiggers de Vries *Primitive Specification for NLS* <http://www.ecrypt.eu.org/stream/nls.html>
5. Mitsuru Matsui, *Linear cryptanalysis method for DES cipher* Advances in Cryptology-Eurocrypt'93, Lecture Notes in Computer Science, vol.765, R. Cramer (Ed.), pp.386-397, Springer-Verlag, 1993.
6. <http://www.ecrypt.eu.org/stream/>

A Bias of equation (4)

Recall approximation (4):

$$(r'[a] + r''[b])_{(i+k,i)} \oplus (r'[c] + r''[d])_{(i+k,i)} \oplus (r[a] + r[c])_{(i+k-19,i-19)} \oplus (r[b] + r[d])_{(i+k-9,i-9)} = 0.$$

There are 4 additions involved in approximation (4), namely $(r''[a] + r'[b])$, $(r''[c] + r'[d])$, $(r[a] + r[c])$ and $(r[b] + r[d])$. Each of these 4 additions may or may not contribute a carry bit to the particular bits XORed in the approximation. We denote $carryin_i = (c_3, c_2, c_1, c_0)$ to be the carry vector at bit position i that symbolises the carry bits contributed in to the approximation by each of the 4 additions. There are 16 different $carryin_i$ vectors that can affect this approximation, $\{(0, 0, 0, 0), (0, 0, 0, 1), \dots, (1, 1, 1, 1)\}$, and each case must be studied.

To calculate the total bias of approximation (4), we need to calculate the bias of approximation (4) given each $carryin_i$ vector, and also the probability of each carryin vector occurring. Note that $carryin_{i+1}$ (vector of carry bits coming out of the 4 additions) given from approximation (4) is only dependent on the 4 variables $(r[a], r[b], r[c], r[d])$ and $carryin_i$. It is not dependent on $carryin_{i-1} \dots carryin_0$. Assuming that the values at bit position i of variables $r[a], r[b], r[c]$ and $r[d]$ are either 0 or 1 with equal probability, we can state that $carryin_{i+1}$ is dependent on the $carryin_i$ only, and hence the sequence of

$carryin_i$ vectors forms a markov process. To calculate the probability of each $carryin_i$ occurring, we examine approximation (4) over all possible $carryin_i$ and note $carryin_{i+1}$ given by each case, this determines the transition matrix for this markov process. This transition matrix gives the probability of $carryin_{i+1}$, given $carryin_i$, where the rows and columns represent $carryin_{i+1}$ and $carryin_i$ respectively.

$$M_{carryin_i} = \frac{1}{16} \cdot \begin{bmatrix} 7 & 3 & 3 & 1 & 3 & 2 & 2 & 1 & 3 & 2 & 2 & 1 & 1 & 1 & 1 & 1 \\ 1 & 5 & 1 & 3 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 5 & 3 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 5 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 3 & 1 & 1 & 0 \\ 1 & 2 & 0 & 0 & 2 & 6 & 1 & 2 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 1 \\ 1 & 0 & 2 & 0 & 2 & 1 & 6 & 2 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 1 \\ 0 & 1 & 1 & 3 & 0 & 1 & 1 & 5 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 5 & 1 & 1 & 0 & 3 & 1 & 1 & 0 \\ 1 & 2 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 6 & 1 & 2 & 0 & 2 & 0 & 1 \\ 1 & 0 & 2 & 0 & 0 & 0 & 1 & 0 & 2 & 1 & 6 & 2 & 0 & 0 & 2 & 1 \\ 0 & 1 & 1 & 3 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 5 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 3 & 5 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 3 & 1 & 5 & 1 \\ 1 & 1 & 1 & 1 & 1 & 2 & 2 & 3 & 1 & 2 & 2 & 3 & 1 & 3 & 3 & 7 \end{bmatrix}$$

Using this matrix we can calculate the probability that each carryout vector occurs, given the probability of each carryin vector occurring. To calculate the carryin vector distribution for bit position i ($carryin_{(i)}$) of approximation (4) we can simply multiply the transition matrix by the carryin vector from the previous bit position $i - 1$ ($carryin_{(i-1)}$), since $carryin_i = carryout_{(i-1)}$. To find these distributions we need to calculate the carryin vector at bit position 0 ($carryin_{(0)}$). That is, we need the carry vector in to

$$(r''[a] + r'[b])_{(0)} \oplus (r''[c] + r'[d])_{(0)} \oplus (r[a] + r[c])_{(13)} \oplus (r[b] + r[d])_{(23)} = 0.$$

There cannot be any carry bit coming in to the additions involving the least significant bits, however there can be carry bits coming into the additions that involve bit positions 13 and 23. There are 4 possible carryin vectors: $(0, 0, 0, 0)$, $(0, 0, 0, 1)$, $(0, 0, 1, 0)$ and $(0, 0, 1, 1)$. To calculate the probability of each of these carryin vectors, first observe the following equation that gives the probability of a carryin bit equal to zero for bit position i .

$$P(carry[i] = 0) = \frac{1}{2} \cdot \left(1 + \frac{1}{2^i}\right)$$

Therefore the probability of each of these carry vectors occurring can be calculated and is shown in the following table:

Carry vector	$P(c_3 = 0)$	$P(c_2 = 0)$	$P(c_1 = 0)$	$P(c_0 = 0)$	$P(\text{carryvector})$
0000	1	1	$\frac{1}{2}(1 + 2^{-13})$	$\frac{1}{2}(1 + 2^{-23})$	$\frac{1}{4}(1 + 2^{-13} + 2^{-23} + 2^{-36})$
0001	1	1	$\frac{1}{2}(1 + 2^{-13})$	$\frac{1}{2}(1 - 2^{-23})$	$\frac{1}{4}(1 + 2^{-13} - 2^{-23} - 2^{-36})$
0010	1	1	$\frac{1}{2}(1 - 2^{-13})$	$\frac{1}{2}(1 + 2^{-23})$	$\frac{1}{4}(1 - 2^{-13} + 2^{-23} - 2^{-36})$
0011	1	1	$\frac{1}{2}(1 - 2^{-13})$	$\frac{1}{2}(1 - 2^{-23})$	$\frac{1}{4}(1 - 2^{-13} - 2^{-23} + 2^{-36})$

Table 5. Probability distribution for carry vectors in the LSB case.

Using the probabilities for the carry bits going into the least significant bit and the transition matrix, we can calculate the probabilities of the carry bits going into the next significant bit (ie. $i = 1$). We can continue the same process to calculate the carryin probability distribution for all bit positions i . Note there are two inconsistencies produced at $i = 9$ and $i = 19$, these occur since we are examining approximations:

$$(r''[a] + r'[b])_{(10,9)} \oplus (r''[c] + r'[d])_{(10,9)} \oplus (r[a] + r[c])_{(23,22)} \oplus (r[b] + r[d])_{(1,0)} = 0$$

$$(r''[a] + r'[b])_{(20,19)} \oplus (r''[c] + r'[d])_{(20,19)} \oplus (r[a] + r[c])_{(1,0)} \oplus (r[b] + r[d])_{(10,9)} = 0.$$

and we have a similar problem as before where we are considering least significant bits.

We now need to calculate the bias of the XOR of adjacent bits of approximation (4). We give results for the cases involving 2 and 4 adjacent bits as these are the most important results needed for our study.

A.1 Two Adjacent Bits

We observe all cases for 2 adjacent bits of approximation (4) and calculate the bias of the approximation given each carryin vector. The results are given in the following table, where we use the hexadecimal value of the carryin vector as notation. We can then multiply these bias' by the corresponding probability

carryin	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
bias	0.5	0.0	0.0	-0.5	0.0	0.5	0.5	0.0	0.0	0.5	0.5	0.0	-0.5	0.0	0.0	0.5

Table 6. Bias for 2 adjacent bits of approximation (4) given carryin

of each carryin vector occurring to give us the total bias for 2 adjacent bits of approximation (4) at bitposition i . For example, the bias at bit position 0 is

determined by

$$\begin{aligned}
 P(A = 0) &= \sum_{i=0}^{15} P(\text{carryin}_i = 0) \cdot \text{Bias}(A = 0 | \text{carryin}_i) \\
 &= \frac{1}{8}(1 + 2^{-13} + 2^{-23} + 2^{-36}) + \frac{1}{8}(1 - 2^{-13} - 2^{-23} + 2^{-36}) \\
 &= -0.00003
 \end{aligned}$$

We can repeat this method to achieve the bias for all bit positions, the results are given in table (2).

A.2 Four Adjacent Bits

Using a similar process, we can calculate the bias for 4 adjacent bits of approximation (4). The bias distribution, given the *carryin* vectors is stated in the following table:

carryin	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
bias	0.22	-0.06	-0.06	-0.14	-0.06	0.22	0.22	-0.06	-0.06	0.22	0.22	-0.06	-0.14	-0.06	-0.06	0.22

Table 7. Bias for 4 adjacent bits of approximation (4) given carryin

Using the probabilities of carryin vectors, we can calculate the overall bias for 4 adjacent bits of approximation (4) for all bit positions. The results are given in table (3).