A Linear Distinguisher for Dragon

Joo Yeon Cho and Josef Pieprzyk

Centre for Advanced Computing – Algorithms and Cryptography,
Department of Computing, Macquarie University,
NSW, Australia, 2109
{jcho,josef}@ics.mq.edu.au

Abstract. Dragon stream cipher is one of the focus ciphers which have reached Phase 2 of the eSTREAM project. In this paper, we present a new method of building a linear distinguisher for Dragon. The distinguisher is constructed by exploiting the biases of two S-boxes and the modular addition which are basic components of the nonlinear function F. The bias of the distinguisher is estimated to be around $2^{-75.8}$ which is better than the bias of the distinguisher presented by Englund and Maximov. We claim that Dragon is distinguishable from a random cipher by using around $2^{151.6}$ keystream words and 2^{59} memory.
Keywords : Stream Ciphers, eSTREAM, Dragon, Modular Addition.

1 Introduction

Dragon [1] is a word-oriented stream cipher submitted to the eSTREAM project [4]. Dragon is one of the focus ciphers (Software) which are included in Phase 2 of the eSTREAM. During Phase 1, Englund and Maximov presented a distinguishing attack against Dragon. Their distinguisher is constructed using around 2^{155} keystream words and 2^{96} memory [2].

In this paper, Unlike Englund and Maximow, we use a different approach to find more efficient distinguisher. In a nut shell, we first derive linear approximations for the basic nonlinear blocks used in the cipher, namely, for two S-boxes and for modular additions. Next we combine those approximations and build a linear approximation for the whole state update function F. While combining these elementary approximations we use two basic operations that we call cutting and bypassing. The bypassing operation replaces the original element by its approximation but the cutting operation replaces some combination of internal variables by zero. Finally, we design the distinguisher by linking the approximation of the update function F with the observable variables for two consecutive clocks.

The bias of our distinguisher is estimated to be around $2^{-75.8}$ when 2^{59} bits of internal memory are guessed. Hence, we claim that Dragon is distinguishable from the random process with around $2^{151.6}$ words data complexity and with 2^{59} memory complexity. This complexity is better than the one presented in [2] which needs 2^{155} data and 2^{96} memory. Our distinguisher is also described explicitly by showing the best approximations of the nonlinear components of the cipher. In contrast, the previous best attack by Englund and Maximov used a statistical argument to evaluate a bias of the function F.

This paper is organized as follows. Section 2 presents a brief description of Dragon. In Section 3, a series of linear approximations of nonlinear components of Dragon is presented. And then, a distinguisher is built by combining the approximations properly. Section 4 concludes the work.

2 A brief description of Dragon

Dragon consists of a 1024-bit nonlinear feedback register, a nonlinear state update function, and a 64-bit internal memory. Dragon produces a 64-bit (two words) output every clock. The nonlinear state update function, which is called the function F, takes six state words (192-bit) as input and produces six words (192 bits) as output. Among the output words of F function, two words are used as new state words and two words are produced as a keystream. The detail structure of the F function is displayed in Figure 1.

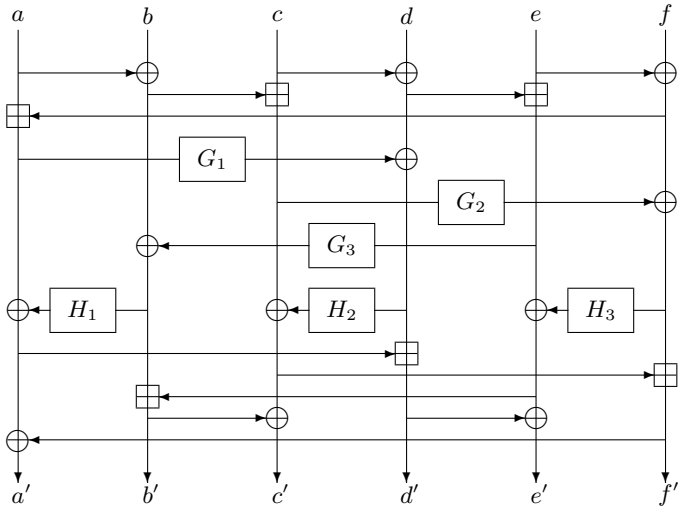

Fig. 1. F function

Suppose that the 32-bit input x is split into four bytes such as $x = x_0||x_1||x_2||x_3$ where x_i denotes a single byte. Then, the functions G and H that are used inside the F function are constructed by using two 8×32 s-boxes, which are called as S_1 and S_2, in the following way.

$$
\begin{aligned}
G_1(x) &= S_1(x_0) \oplus S_1(x_1) \oplus S_1(x_2) \oplus S_2(x_3) \\
G_2(x) &= S_1(x_0) \oplus S_1(x_1) \oplus S_2(x_2) \oplus S_1(x_3) \\
G_3(x) &= S_1(x_0) \oplus S_2(x_1) \oplus S_1(x_2) \oplus S_1(x_3) \\
H_1(x) &= S_2(x_0) \oplus S_2(x_1) \oplus S_2(x_2) \oplus S_1(x_3) \\
H_2(x) &= S_2(x_0) \oplus S_2(x_1) \oplus S_1(x_2) \oplus S_2(x_3) \\
H_3(x) &= S_2(x_0) \oplus S_1(x_1) \oplus S_2(x_2) \oplus S_2(x_3)
\end{aligned}
$$

Using the F function, the keystream is generated as follows.

1. Input : $\{B_0, B_1, \ldots, B_{31}\}$ and $M = (M_L||M_R)$
2. $a = B_0, b = B_9, c = B_{16}, d = B_{19}, e = B_{30} \oplus M_L, f = B_{31} \oplus M_R$ where $M = M_R||M_L$.
3. $(a', b', c', d', e', f') = F(a, b, c, d, e, f)$
4. $B_0 = b', B_1 = c'$ and $B_i = B_{i-2}, 2 \leq i \leq 31, M = M + 1$
5. Output : $k = (a'||e')$

For a complete description of Dragon, we refer the reader to the paper [1].

3 A linear distinguisher for Dragon

Let n be a non-negative integer. Given two vectors $x = (x_{n-1}, \ldots, x_0)$ and $y = (y_{n-1}, \ldots, y_0)$ where $x, y \in GF(2^n)$, let $x \cdot y$ denote a standard inner product defined as $x \cdot y = x_{n-1}y_{n-1} \oplus \ldots \oplus x_0 y_0$. A linear mask is a constant vector that is used to compute an inner product of a n-bit string.

Let us assume that we have a function $f : \{0,1\}^m \rightarrow \{0,1\}^n$ for some positive integers m and n. Given a linear output mask $\Gamma \in GF(2^n)$ and a linear input mask $\Lambda \in GF(2^m)$, the bias of an approximation $\Gamma \cdot f(x) = \Lambda \cdot x$ is measured as follows.

$$\epsilon_f(\Gamma, \Lambda) = 2^{-n}(\#(\Gamma \cdot f(x) \oplus \Lambda \cdot x = 0) - \#(\Gamma \cdot f(x) \oplus \Lambda \cdot x = 1))$$

where $x \in GF(2^m)$ and runs through all possible values. Then, $Pr[\Gamma \cdot f(x) = \Lambda \cdot x] = \frac{1}{2}(1 + \epsilon_f(\Gamma, \Lambda))$. Note that given q independent approximations holding each bias of ϵ, the combination of q approximations has the bias of ϵ^q according to the well-known Piling-up Lemma [3].

This section is organized as follows. First, we derive effective approximations for the two S-boxes that are used to construct them. Next we do similar work for the modular addition and find best linear approximations for it. Finally, we show how to combine the approximations found to construct the distinguisher.

3.1 Approximations of functions G and H

According to the structure of the functions G and H, the essential components of the functions G and H are the two S-boxes: S_1 and S_2. Hence, the linear approximations of the functions G and H can be constructed by combining approximations of S_1 and S_2 appropriately. For our distinguisher which will be described next section, we need to compute the biases of the approximations displayed in Table 1. Note that the approximations of the

approximation	bias
$\Gamma \cdot G_1(x) = \Gamma \cdot x$	$\epsilon_{G_1}(\Gamma, \Gamma)$
$\Gamma \cdot G_2(x) = \Gamma \cdot x$	$\epsilon_{G_2}(\Gamma, \Gamma)$
$\Gamma \cdot H_1(x) = 0$	$\epsilon_{H_1}(\Gamma, 0)$
$\Gamma \cdot H_2(x) = 0$	$\epsilon_{H_2}(\Gamma, 0)$
$\Gamma \cdot H_3(x) = 0$	$\epsilon_{H_3}(\Gamma, 0)$

Table 1. Required approximations of G, H for the distinguisher

function G use an input and an output masks which are identical, while those of function H use only an output mask. The reason for this will be explained in Subection 3.2.

We call the approximations of the form $\Gamma \cdot G(x) = \Gamma \cdot x$ **by-passing** approximations, whereas the approximations of the form $\Gamma \cdot H(x) = 0$ **cutting** approximations.

Approximations of the function H Assume that a 32-bit x is a uniformly distributed random variable and is divided into four bytes. In other words, we have $x = x_0||x_1||x_2||x_3$, where x_i is a single byte. Then, the approximation $\Gamma \cdot H_1(x) = 0$ can be represented as

$$\Gamma \cdot H_1(x) = \Gamma \cdot S_2(x_0) \oplus \Gamma \cdot S_2(x_1) \oplus \Gamma \cdot S_2(x_2) \oplus \Gamma \cdot S_1(x_3) = 0 \quad (1)$$

Hence, the bias $\epsilon_{H_1}(\Gamma, 0)$ can be computed as

$$\epsilon_{H_1}(\Gamma, 0) = \epsilon_{S_2}(\Gamma, 0)^3 \times \epsilon_{S_1}(\Gamma, 0),$$

where $\epsilon_{S_i}(\Gamma, 0)$ is the bias of an approximation $\Gamma \cdot S_i(x_j) = 0$. Since $\Gamma \cdot H_2$ and $\Gamma \cdot H_3$ are equivalent to $\Gamma \cdot H_3$, their biases are identical to that of H_1, i.e. $\epsilon_{H_1}(\Gamma, 0) = \epsilon_{H_2}(\Gamma, 0) = \epsilon_{H_3}(\Gamma, 0)$.

Approximations of the function G Assume that a 32-bit variable x is a uniformly distributed random variable and is divided into four bytes so that $x = x_0||x_1||x_2||x_3$, where x_i is a byte. Also, assume that a mask Γ is divided into four submasks so $\Gamma = \Gamma_0||\Gamma_1||\Gamma_2||\Gamma_3$, where $\Gamma_i \in \{0,1\}^8$. Then, the approximation $\Gamma \cdot x = \Gamma \cdot G(x)$ can be decomposed into

$$\begin{aligned}\Gamma \cdot (G_1(x) \oplus x) &= (\Gamma \cdot S_1(x_0) \oplus \Gamma_0 \cdot x_0) \oplus (\Gamma \cdot S_1(x_1) \oplus \Gamma_1 \cdot x_1) \oplus \\ &\quad (\Gamma \cdot S_1(x_2) \oplus \Gamma_2 \cdot x_2) \oplus (\Gamma \cdot S_2(x_3) \oplus \Gamma_3 \cdot x_3) \\ &= 0\end{aligned}$$

Hence, the bias $\epsilon_G(\Gamma, \Gamma)$ can be computed as follows

$$\epsilon_G(\Gamma, \Gamma) = \epsilon_{S_1(x_0)}(\Gamma, \Gamma_0) \times \epsilon_{S_1(x_1)}(\Gamma, \Gamma_1) \times \epsilon_{S_1(x_2)}(\Gamma, \Gamma_2) \times \epsilon_{S_2(x_3)}(\Gamma, \Gamma_3),$$

where $\epsilon_{S_i(x_j)}(\Gamma, \Gamma_j)$ denotes the bias of the approximation $\Gamma \cdot S_i(x_j) \oplus \Gamma_j \cdot x_j = 0$.

Linear approximations of modular addition Let x and y be uniformly distributed random vectors, where $x, y \in GF(2^n)$ for a positive n. Given a mask $\Gamma \in GF(2^n)$, the bias of modular addition, which is denoted by $\epsilon_+(\Gamma, \Gamma)$, is defined as follows.

$$Pr[\Gamma \cdot (x \boxplus y) = \Gamma \cdot (x \oplus y)] = \frac{1}{2}(1 + \epsilon_+(\Gamma, \Gamma)). \tag{2}$$

We also define that, given a vector x, the Hamming weight of x is defined as the number of nonzero coordinates of x.

Theorem 1. *Let n and m are positive integers. Assume that $\Gamma = (\gamma_{n-1}, \cdots, \gamma_0)$ where $\gamma_i \in \{0,1\}$ and the Hamming weight of Γ is m. Let W_Γ denote a vector of bit positions where $\gamma_i = 1$. So, $W_\Gamma = (w_{m-1}, \cdots, w_0)$ where $0 \le w_j \le n$.*
If m is even, then,

$$\epsilon_+(\Gamma, \Gamma) = 2^{-d_1} \text{ where } d_1 = \sum_{i=0}^{m/2-1} (w_{2i+1} - w_{2i}) \tag{3}$$

If m is odd, then

$$\epsilon_+(\Gamma, \Gamma) = 2^{-d_2} \text{ where } d_2 = \sum_{i=1}^{(m-1)/2} (w_{2i} - w_{2i-1}) + w_0 \tag{4}$$

Proof. See Appendix A.

For example, if Γ = 0x00000C49, the Hamming weight of Γ is 5 and $W_\Gamma = (11, 10, 6, 3, 0)$. Hence, $\epsilon_+(\Gamma, \Gamma) = 2^{-[(11-10)+(6-3)]} = 2^{-4}$.

Corollary 1. *Let m be a positive integer. Given a mask vector Γ whose Hamming weight is m, the approximation $\Gamma \cdot (x \boxplus y) = \Gamma \cdot (x \oplus y)$ has at most a bias of $2^{-(m-1)/2}$.*

Proof. See Appendix B.

3.2 Linear approximation of F function

According to the state update method of Dragon, there is the following relation between two state words chosen at clock t and $t+15$. [1]

$$B_0[t] = B_{30}[t+15], \quad t = 0, 1, \ldots \tag{5}$$

We know that $a = B_0$ and $e = B_{30} \oplus M_L$ where a and e are two words out of six input words of the F function. Then, we try to find the linear approximations $\Gamma \cdot a' = \Gamma \cdot a$ and $\Gamma \cdot e' = \Gamma \cdot e$ where a' and e' are two output words of the F function that are produced as a keystream.

We regard the outputs of the functions G and H as independent and uniformly distributed random variables. This assumption is reasonable since each G or H function has unique input parameters at given clock t so that the output of the functions G and H are mutually independent. Hence, the functions G and H are described without input parameters in the following expressions.

The approximation of a' As illustrated in Figure 1, an output word a' is expressed as follows.

$$a' = [(a \boxplus (e \oplus f)) \oplus H_1] \oplus [(e \oplus f \oplus G_2) \boxplus (H_2 \oplus ((a \oplus b) \boxplus c))]$$

Due to the linear property of Γ, we know that

$$\Gamma \cdot a' = \Gamma \cdot [(a \boxplus (e \oplus f)) \oplus H_1] \oplus \Gamma \cdot [(e \oplus f \oplus G_2) \boxplus (H_2 \oplus ((a \oplus b) \boxplus c))].$$

By applying Approximation (2), we get

$$\Gamma \cdot [(e \oplus f \oplus G_2) \boxplus (H_2 \oplus ((a \oplus b) \boxplus c))] = \Gamma \cdot (e \oplus f \oplus G_2) \oplus \Gamma \cdot [(H_2 \oplus ((a \oplus b) \boxplus c))]$$

which holds with the bias of $\epsilon_+(\Gamma, \Gamma)$. Hence, we have

$$\Gamma \cdot a' = \Gamma \cdot [(a \boxplus (e \oplus f)) \oplus H_1] \oplus \Gamma \cdot (e \oplus f \oplus G_2) \oplus \Gamma \cdot [H_2 \oplus ((a \oplus b) \boxplus c)].$$

Next, the two types of approximations are used in our analysis: First, cutting approximations are used for the functions H_1 and H_2. That is, we use $\Gamma \cdot H_1 = 0$ and $\Gamma \cdot H_2 = 0$ which hold the biases of $\epsilon_{H_1}(\Gamma, 0)$ and $\epsilon_{H_2}(\Gamma, 0)$, respectively. Intuitively, these approximations allow to simplify the form of the final approximation of the function F by using replacing the output variables by zeros. Second, by-passing approximations are used for the function G_2. That is, we use $\Gamma \cdot G_2 = \Gamma \cdot [(a \oplus b) \boxplus c]$ that has a bias of $\epsilon_{G_2}(\Gamma, \Gamma)$. In this category of approximations we are able to replace input variables by a combination of output variables (and vice versa). The replacement is specified by the by-passing approximation. Then, we have

$$\begin{aligned} \Gamma \cdot a' &= \Gamma \cdot [(a \boxplus (e \oplus f))] \oplus \Gamma \cdot (e \oplus f \oplus [(a \oplus b) \boxplus c]) \oplus \Gamma \cdot [(a \oplus b) \boxplus c] \\ &= \Gamma \cdot [(a \boxplus (e \oplus f))] \oplus \Gamma \cdot (e \oplus f) \end{aligned}$$

Finally, by applying Approximation (2) for the modular addition, we obtain

$$\Gamma \cdot a' = \Gamma \cdot a \tag{6}$$

[1] This relation was also observed in [2].

We know that $\Gamma \cdot [(a \boxplus (e \oplus f))] = \Gamma \cdot a \oplus \Gamma \cdot (e \oplus f)$ holds the bias of $\epsilon_+(\Gamma, \Gamma)$. Therefore, the bias of Approximation (6) can be computed from the biases of the component approximations as follows:

$$\epsilon_{a'}(\Gamma, \Gamma) = \epsilon_+(\Gamma, \Gamma)^2 \times \epsilon_{H_1}(\Gamma, 0) \times \epsilon_{H_2}(\Gamma, 0) \times \epsilon_{G_2}(\Gamma, \Gamma)$$

Since the 32-bit word a' is an upper part of a 64-bit keystream output at each clock, Approximation (6) is equivalent to the following expression.

$$\Gamma \cdot k_0[t] = \Gamma \cdot B_0[t] \tag{7}$$

where $k_0[t]$ denotes the upper part of a 64-bit k at clock t.

The approximation of e' As depicted in Figure 1, an output word e' is described as

$$e' = [((a \boxplus (e \oplus f)) \oplus H_1) \boxplus (c \oplus d \oplus G_1)] \oplus [H_3 \oplus ((c \oplus d) \boxplus e)]$$

Similarly to the case of a', we would like to obtain an approximation $\Gamma \cdot e' = \Gamma \cdot e$. To do this, we first apply Approximation (2) for modular addition and as the result we get

$$\Gamma \cdot e' = \Gamma \cdot [(a \boxplus (e \oplus f)) \oplus H_1] \oplus \Gamma \cdot (c \oplus d \oplus G_1) \oplus \Gamma \cdot [H_3 \oplus ((c \oplus d) \boxplus e)]$$

Next, we apply the cutting approximations for functions H_1, H_3 and the by-passing approximation for the function G_1. That is, we use the following approximations

$$\Gamma \cdot H_1 = 0, \quad \Gamma \cdot H_3 = 0, \quad \Gamma \cdot G_1 = \Gamma \cdot [a \boxplus (e \oplus f)]$$

that hold with the biases of $\epsilon_{H_1}(\Gamma, 0), \epsilon_{H_3}(\Gamma, 0)$ and $\epsilon_{G_1}(\Gamma, \Gamma)$, respectively. These approximations are plugged into the above relation and we obtain the following result

$$\begin{aligned} \Gamma \cdot e' &= \Gamma \cdot [(a \boxplus (e \oplus f))] \oplus \Gamma \cdot (c \oplus d \oplus [a \boxplus (e \oplus f)]) \oplus \Gamma \cdot [(c \oplus d) \boxplus e] \\ &= \Gamma \cdot (c \oplus d) \oplus \Gamma \cdot [(c \oplus d) \boxplus e]. \end{aligned}$$

Finally, by applying Approximation (2) for modular addition, we can conclude that output e' and input e satisfy the following approximation

$$\Gamma \cdot e' = \Gamma \cdot e \tag{8}$$

with the bias of

$$\epsilon_{e'}(\Gamma, \Gamma) = \epsilon_+(\Gamma, \Gamma)^2 \times \epsilon_{H_1}(\Gamma, 0) \times \epsilon_{H_3}(\Gamma, 0) \times \epsilon_{G_1}(\Gamma, \Gamma).$$

Since the 32-bit word e' is a lower part of a 64-bit keystream output k at each clock, Approximation (8) is equivalent to the following expression.

$$\Gamma \cdot k_1[t] = \Gamma \cdot (B_{30}[t] \oplus M_L[t]) \tag{9}$$

where $k_1[t]$ and $M_L[t]$ denote the lower part of a 64-bit k and the upper part of a 64-bit memory word M at clock t, respectively.

Building the distinguisher According to Equation (5), Approximations (7) and (9) can be combined in such a way that

$$\Gamma \cdot k_0[t] = \Gamma \cdot B_0[t] = \Gamma \cdot B_{30}[t+15] = \Gamma \cdot (k_1[t+15] \oplus M_L[t+15])$$

By guessing (partially) the initial value of M, we can build the following distinguisher.

$$\Gamma \cdot k_0[t] = \Gamma \cdot (k_1[t+15]) \tag{10}$$

For the correctly guessed initial value of M, the distinguisher (10) shows the bias of

$$\begin{aligned}\epsilon_D(\Gamma,\Gamma) &= \epsilon_{a'}(\Gamma,\Gamma) \times \epsilon_{e'}(\Gamma,\Gamma) \\ &= \epsilon_+(\Gamma,\Gamma)^4 \times \epsilon_{H_1}(\Gamma,0)^2 \times \epsilon_{H_2}(\Gamma,0) \times \epsilon_{H_3}(\Gamma,0) \times \epsilon_{G_1}(\Gamma,\Gamma) \times \epsilon_{G_2}(\Gamma,\Gamma)\end{aligned} \tag{11}$$

3.3 Searching for best linear mask

We implemented our mask search for the function F to achieve the biggest bias of the distinguisher (10). The space of a linear mask Γ contains $2^{32}-1$ elements. For each mask Γ, the following procedure is performed to compute the bias given by Expression (11).

Step 1. For an input x that varies from 0 to 255, measure the biases of $\Gamma \cdot S_1(x) = 0$ and $\Gamma \cdot S_2(x) = 0$, respectively. Then, compute $\epsilon_{H_1}(\Gamma,0)$, $\epsilon_{H_2}(\Gamma,0)$ and $\epsilon_{H_3}(\Gamma,0)$.
Step 2. The mask Γ is divided into four submasks $\Gamma = \Gamma_0||\Gamma_1||\Gamma_2||\Gamma_3$. For an input x that varies from 0 to 255, measure the bias of $\Gamma \cdot S_1(x) = \Gamma_i \cdot x$ and $\Gamma \cdot S_2(x) = \Gamma_i \cdot x$ for some $0 \le i \le 3$. Then, compute the biases $\epsilon_{G_1}(\Gamma,\Gamma)$ and $\epsilon_{G_2}(\Gamma,\Gamma)$.
Step 3. Compute $\epsilon_+(\Gamma,\Gamma)$ using Theorem 1.
Step 4. Finally, compute $\epsilon_D(\Gamma,\Gamma)$.

Time complexity for searching We assume that given x, finding $S_1(x)$ or $S_2(x)$ takes a single time unit. For each mask Γ, we can compute $\epsilon_{H_1}(\Gamma,0)$ by measuring the biases of two S-boxes, which requires $2^8 \times 2$ time units. The biases $\epsilon_{H_2}(\Gamma,0)$ and $\epsilon_{H_3}(\Gamma,0)$ are the same as the $\epsilon_{H_1}(\Gamma,0)$. For the bias $\epsilon_{G_1}(\Gamma,\Gamma)$, we need to decompose the function G_1 into four blocks, which are measured individually. This requires $2^8 \times 4$ time units. Equivalently, we need the same time complexity for measuring the bias $\epsilon_{G_2}(\Gamma,\Gamma)$. Hence, the total complexity (measured by the time units needed to compute the biases) is

$$2^{32} \times (2^8 \times 2 + \times 2^8 \times 4 + 2^8 \times 4) = 5 \times 2^{41}$$

3.4 Our results

We searched for a linear mask which maximizes the bias (11). Due to Corollary 1, the bias $\epsilon_+(\Gamma,\Gamma)$ decreases exponentially as the Hamming weight of a linear mask increases. Hence, there is a better chance to achieve higher bias when the Hamming weight is smaller.

In result, we found that the best linear approximation of the function F is using the distinguisher (10) with the mask Γ = 0x0600018d. The bias of the distinguisher in this case is $2^{-75.8}$ as listed in Table 2. In order to remove the impact of the unknown state of the internal memory on the bias, we need to guess the first 27 bits of initial value of M_L and 32 bits of M_R. Hence, we need to store all possible values of the internal state which takes $2^{27+32} = 2^{59}$ bits.

Table 3 shows that the biases of the functions G and H are much higher than those expected by the designers of Dragon (see Section 5.5 in [1]). Hence, the complexity of linear cryptanalysis expected by designers is incorrect.

Γ	$\epsilon_H(\Gamma,0)$	$\epsilon_{G_1}(\Gamma,\Gamma)$	$\epsilon_{G_2}(\Gamma,\Gamma)$	$\epsilon_+(\Gamma,\Gamma)$	$\epsilon_{a'}(\Gamma,\Gamma)$	$\epsilon_{e'}(\Gamma,\Gamma)$	$\epsilon_D(\Gamma,\Gamma)$
0x0600018d	2^{-3}	$-2^{-8.58}$	$2^{-13.59}$	$2^{-15.91}$	$-2^{-39.1}$	$-2^{-36.7}$	$2^{-75.8}$

Table 2. The biases of the distinguisher

Γ	$\epsilon_{S_1(x_0)}(\Gamma,\Gamma_0)$	$\epsilon_{S_1(x_1)}(\Gamma,\Gamma_1)$	$\epsilon_{S_1(x_2)}(\Gamma,\Gamma_2)$	$\epsilon_{S_2(x_3)}(\Gamma,\Gamma_3)$	$\epsilon_{G_1}(\Gamma,\Gamma)$
0x0600018d	$-2^{-4.42}$	$2^{-2.30}$	$2^{-3.19}$	$-2^{-3.68}$	$2^{-13.59}$
	$\epsilon_{S_1(x_0)}(\Gamma,\Gamma_0)$	$\epsilon_{S_1(x_1)}(\Gamma,\Gamma_1)$	$\epsilon_{S_2(x_2)}(\Gamma,\Gamma_2)$	$\epsilon_{S_1(x_3)}(\Gamma,\Gamma_3)$	$\epsilon_{G_2}(\Gamma,\Gamma)$
	$-2^{-4.42}$	$2^{-2.30}$	2^{-6}	$-2^{-3.19}$	$2^{-15.91}$
	$\epsilon_{S_1}(\Gamma,0)$	$\epsilon_{S_2}(\Gamma,0)$	$\epsilon_{H_1}(\Gamma,0)$	$\epsilon_{H_2}(\Gamma,0)$	$\epsilon_{H_3}(\Gamma,0)$
	$2^{-2.30}$	$-2^{-2.09}$	$-2^{-8.58}$	$-2^{-8.58}$	$-2^{-8.58}$

Table 3. The biases of the functions G and H used in the distinguisher

4 Conclusion

In this paper, we presented a new distinguisher for Dragon. Since the amount of observations for the distinguishing attack is by far larger than the limit of keystream available from a single key, our distinguisher leads only to a theoretical attack on Dragon. However, our analysis shows that some approximations of the functions G and H have larger biases than the ones expected by the designers. As far as we know, our distinguisher is the best one for Dragon published so far in open literature. In addition, we present an efficient algorithm to compute the biases of approximations of modular additions, which may be useful for other distinguishing attacks that use modular addition.

Acknowledgment We wish to thank Matt Henricksen for invaluable comments. The authors were supported by ARC grants DP0451484 and DP0663452. This work was also supported by Macquarie University ARC Safety Net Grant.

References

1. E. Dawson, K. Chen, M. Henricksen, W. Millan, L. Simpson, H. Lee, and S. Moon, *Dragon: A fast word based stream cipher*, eSTREAM, ECRYPT Stream Cipher Project, Report 2005/006, 2005, http://www.ecrypt.eu.org/stream.
2. H. Englund and A. Maximov, *Attack the Dragon.*, Progress in Cryptology - INDOCRYPT 2005, Lecture Notes in Computer Science, vol. 3797, Springer, 2005, pp. 130–142.
3. M. Matsui, *Linear cryptoanalysis method for DES cipher.*, Advances in Cryptology - EUROCRYPT '93, Lecture Notes in Computer Science, vol. 765, Springer, 1993, pp. 386–397.
4. ECRYPT NoE, *eSTREAM - the ECRYPT stream cipher project*, Available at http://www.ecrypt.eu.org/stream/, 2005.

A Proof of Theorem 1

Suppose that $z = x \boxplus y$ where $x = (x_{n-1}, \cdots, x_0), y = (y_{n-1}, \cdots, y_0)$ and $z = (z_{n-1}, \cdots, z_0)$. Then, each z_i bit is expressed a function of $x_i, \cdots, x_0$ and $y_i, \cdots, y_0$ bits as follows.

$$z_0 = x_0 \oplus y_0, \quad z_i = x_i \oplus y_i \oplus x_{i-1}y_{i-1} \oplus \sum_{j=0}^{i-2} x_j y_j \prod_{k=j+1}^{i-1} (x_k \oplus y_k), \quad i = 1, \cdots, n$$

If we define the carry $R(x, y)$ as

$$R(x,y)_0 = x_0 y_0, \quad R(x,y)_i = x_i y_i \oplus \sum_{j=0}^{(i-1)} x_j y_j \prod_{k=j+1}^{i} (x_k \oplus y_k), \quad i = 1, 2, \ldots$$

Then, it is clear that $z_i = x_i \oplus y_i \oplus R(x,y)_{i-1}$ for $i > 0$. By the definition, $R(x,y)_i$ has the following recursive relation.

$$R(x,y)_i = x_i y_i \oplus (x_i \oplus y_i) R(x,y)_{i-1} \tag{12}$$

First, we examine the bias of the Γ of which the Hamming weight is 2, i.e. $m = 2$. Without loss of generality, we assume that $\gamma_i = 1$ and $\gamma_j = 1$ where $0 \le j < i < n$. Then, by Relation (12), Approximation (2) is expressed as

$$\begin{aligned} \Gamma \cdot (x \boxplus y) \oplus \Gamma \cdot (x \oplus y) &= z_i \oplus z_j \oplus (x_i \oplus y_i) \oplus (x_j \oplus y_j) \\ &= R(x,y)_{i-1} \oplus R(x,y)_{j-1} \\ &= x_{i-1} y_{i-1} \oplus (x_{i-1} \oplus y_{i-1}) R(x,y)_{i-2} \oplus R(x,y)_{j-1} \end{aligned}$$

Let us denote $p_{i-1} = Pr[R(x,y)_{i-1} \oplus R(x,y)_{j-1} = 0]$. Since x_i and y_i are assumed as uniformly distributed random variables, the probability p_{i-1} is split into the three cases as follows.

$$p_{i-1} = \begin{cases} Pr[R(x,y)_{j-1} = 0], & \text{if } (x_{i-1}, y_{i-1}) = (0,0) \\ Pr[1 \oplus R(x,y)_{j-1} = 0], & \text{if } (x_{i-1}, y_{i-1}) = (1,1) \\ Pr[R(x,y)_{i-2} \oplus R(x,y)_{j-1} = 0], & \text{if } (x_{i-1}, y_{i-1}) = (0,1), (1,0) \end{cases}$$

Clearly, $Pr[R(x,y)_{j-1} = 0] = 1 - Pr[1 \oplus R(x,y)_{j-1} = 0]$. Hence, we get

$$p_{i-1} = \frac{1}{4} + \frac{1}{2} Pr[R(x,y)_{i-2} \oplus R(x,y)_{j-1} = 0] = \frac{1}{4} + \frac{1}{2} p_{i-2}$$

If $j = i - 1$, then $Pr[R(x,y)_{i-2} \oplus R(x,y)_{j-1} = 0] = 1$. Hence, $p_{i-1} = \frac{1}{4} + \frac{1}{2} = \frac{3}{4}$. Otherwise, p_{i-2} is determined recursively by the same technique used as above until p_{j-1} is reached. Hence, we obtain the following result.

$$p_{i-1} = \frac{1}{4}(1 + \cdots + 2^{-(i-j-1)}) + 2^{-(i-j)} = \frac{1}{2}(1 + 2^{-(i-j)}) \tag{13}$$

Therefore, the $\epsilon_+(\Gamma, \Gamma)$ is determined by only the difference between two position i and j of Γ.

Next, we consider the case that Γ has an arbitrary Hamming weight, which is denoted m. Assume that we convert m into an even number m' by using the following technique.

- If m is even, then set $m' = m$.
- If m is odd and $\gamma_0 = 0$, then set $\gamma_0 = 1$ and $m' = m + 1$.
- If m is odd and $\gamma_0 = 1$, then set $\gamma_0 = 0$ and $m' = m - 1$.

In result, the Γ is transformed to Γ' which has the Hamming weight of m'. Since the modular addition is linear for the least significant bit, $\epsilon_+(\Gamma, \Gamma) = \epsilon_+(\Gamma', \Gamma')$. Hence, a new position vector for Γ' is defined as $W_{\Gamma'} = (w_{m'-1}, \cdots, w_0)$ where $0 \le w_j \le n$.

Now, we decompose Γ' into a combination of sub-masks which have the Hamming weight of 2. That is, Γ is expressed as

$$\Gamma = \Omega_{m'/2-1} \oplus \cdots \oplus \Omega_0$$

where Ω_k is a sub-mask which has the nonzero coordinates only at position w_{2k} and w_{2k+1} for $k = 0, 1, \cdots, \frac{m'}{2} - 1$. Clearly, the number of such sub-masks is $\frac{m'}{2}$. For example, if $\Gamma = (0, 0, 1, 1, 0, 1, 1)$, then $\Gamma = \Omega_1 \oplus \Omega_0 = (0, 0, 1, 1, 0, 0, 0) \oplus (0, 0, 0, 0, 0, 1, 1)$.
From (13), we know that the bias of $\Omega_k \cdot (x \boxplus y) \oplus \Omega_k \cdot (x \oplus y)$ is only determined by the difference $w_{2k+1} - w_{2k}$. Hence, according to the Piling-up Lemma [3], the bias of $\Gamma \cdot (x \boxplus y) \oplus \Gamma \cdot (x \oplus y)$ is obtained by combining the $\frac{m'}{2}$ approximations independently. Note that the there are no inter-dependencies among sub-masks. Therefore, the claimed bias is computed as

$$\epsilon_+(\Gamma, \Gamma) = 2^{-[w_{m'-1} - w_{m'-2}] + \cdots + [w_1 - w_0]}$$

If m' is replaced by m, we obtain the claimed bias. □

B Proof of Corollary 1

Recall Theorem 1. If m is even, then,

$$d_1 = \sum_{i=0}^{m/2-1} (w_{2i+1} - w_{2i}) \leq \sum_{i=0}^{m/2-1} 1 = m/2$$

If m is odd, then,

$$d_2 = \sum_{i=1}^{(m-1)/2} (w_{2i} - w_{2i-1}) + w_0 \leq \sum_{i=1}^{(m-1)/2} 1 = (m-1)/2$$

Since $2^{-m/2} < 2^{-(m-1)/2}$, the bias $\epsilon_+(\Gamma, \Gamma) \leq 2^{-(m-1)/2}$. □