Distinguishing Attacks on the Stream Cipher Py[⋆]

Gautham Sekar[†], Souradyuti Paul[‡], and Bart Preneel[‡]

[†]Birla Institute of Technology and Science, Pilani, India,
Dept. of Electronics and Instrumentation, Dept. of Physics.
gautham.sekar@gmail.com,
[‡]Katholieke Universiteit Leuven, Dept. ESAT/COSIC,
Kasteelpark Arenberg 10,
B–3001, Leuven-Heverlee, Belgium
{Souradyuti.Paul, Bart.Preneel}@esat.kuleuven.ac.be

Abstract. The stream cipher Py, which was designed by Biham and Seberry, is a submission for ECRYPT stream cipher competition. The cipher which is based on two large arrays (one is 256 bytes and the other is 1040 bytes) is specifically designed for high speed software applications (Py is more than 2.5 times faster than the RC4 on Pentium III). The paper, for the first time, detects a weakness in the mechanism of the stream cipher Py. We find a statistical bias in the distribution of the output-words at the 1st and the 3rd rounds of the cipher (more generally at the rounds t and $t+2$ where $t > 0$). Using this bias, a distinguisher is constructed that works effectively with $2^{83.82}$ randomly chosen key/IV's. Essentially, for each of $2^{83.82}$ randomly chosen key/IV's, the attacker collects only a pair of bits from the outputs at the 1st and the 3rd rounds to establish the distinguisher. The first implication of the results is that it nullifies the claim of the designers of the cipher, that no distinguishing attacks on the cipher are possible with running time less than the exhaustive search (note that the recommended key-length of Py is 256 bits). Secondly, the fact that the bias is found within the outputs generated in the first three rounds (i.e., a segment of 24 bytes) shows that, for Py, the recommended stream length of 2^{64} bytes is also not secure. These results constitute an academic break of the cipher. We have also detected several biases among many pairs of bits and designed distinguishers from them (with the numbers of key/IV's $2^{85.82}$, $2^{87.82}$ etc.) however, they are less effective than the one described above.

[⋆] This work was supported in part by the Concerted Research Action (GOA) Mefisto 2000/06 and Ambiorix 2005/11 of the Flemish Government and in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.

1 Introduction

The cipher Py, designed by Biham and Seberry [1], was submitted to ECRYPT project [2] as a candidate for Profile 1 which covers software based stream ciphers suitable for high-speed applications. In the last couple of years a growing enthusiasm has been noticed among the cryptographers to design fast and secure stream ciphers because of weaknesses being found in many standards such as RC4 and also due to the failure of the NESSIE project [4] to select any stream cipher for its profile as all of them were found weak. The current stream cipher, namely Py, is one of the attempts in this direction.

The working principle of Py is, to some extent, similar to that of RC4 as it uses the technique of random shuffle to update the internal state. In addition to that Py uses a new technique of rotating all array elements in every round with a minimal of running time. The high performance (2.5 times faster than the RC4 on Pentium III) and its apparent security make this cipher very attractive for selection to the Profile 1 of the ECRYPT project.

In this paper we detect several biased pairs of output bits of Py at rounds t and $t+2$ (where $t > 0$). The weaknesses originate from the non-uniformity of the distributions of carry bits in integer addition that are used in Py. Using those biases we have constructed several distinguishers. We show that the best of them works successfully with $2^{83.82}$ randomly chosen key/IV's. This result essentially violates the designers' claim that distinguishing attack on Py is impossible with time less than the exhaustive search, thereby, breaks the cipher academically. As the bias can be found in the outputs of 1st and 3rd rounds of Py (which requires a segment of 24 bytes of output), the designers recommendation for the allowable length of output stream of 2^{64} bytes is also not safe. However, the weaknesses of Py which are described in this paper, still cannot be implemented in practice as it would require high running time.

2 Description of Py

Py is a synchronous stream cipher which normally uses a 32 byte key (however, the key can be of any size from 1 byte to 256 bytes) and a 16 byte initial value or IV (IV can also be of any size from 1 byte to 64 bytes). Py works in three phases – a key setup algorithm, an IV setup algorithm and a round function in which two output-words (each output-word is 4 bytes long) are generated in every round. The internal state of Py contains two S-boxes Y, P and a variable s. Y contains 260 elements each of which is 32 bits long. The elements of Y are indexed by [-3, -2,..., 256]. P is a permutation of the elements of $\{0, ..., 255\}$. The main feature of the stream cipher Py is that the S-boxes are updated like 'rolling arrays' [1]. The technique of 'rolling arrays' means that, in each round of Py, *(i)* one or two elements of the S-boxes are updated (line 1 and 7 of Algorithm 1) and *(ii)* all the elements of them are cyclically rotated by one position toward left (line 2 and 8 of Algorithm 1). In our analysis, we have assumed that, after the key/IV setup, Y, P and the variable s are uniformly distributed and independent. Under this assumption we analyzed the round function of Py (or Pseudorandom Bit Generation Algorithm (PRBG)) which is described in Algorithm 1. See [1] for a detailed description of the Key/IV setup algorithms.

The inputs to Algorithm 1 are $Y[-3, ..., 256]$, $P[0, ..., 255]$ and s, which are obtained after the key/IV setup. Lines 1 and 2 describe how P is updated and rotated. In the update stage, the 0th element of P is swapped with another element in P, which is accessed indirectly, using $Y[185]$. The next step involves a cyclic rotation by one position, of the elements in P. This implies that the entry in $P[0]$ becomes the entry in $P[255]$ in the next round and the entry in $P[i]$ becomes the entry in $P[i-1]$ ($\forall i \in \{1, 2, ..., 255\}$). Lines 3 and 4 of Algorithm 1 indicate how s is updated and its elements rotated. Here, the '$ROTL32(s, x)$' function implies a cyclic left rotation of s by x bit-positions. The output-words (each 32-bit) are generated in lines 5 and 6. The last two lines of the algorithm explain the update and rotation of the elements of Y. The rotation of Y is carried in the same manner as the rotation of P.

Algorithm 1 Single Round of Py

```
Require: $Y[-3, ..., 256]$, $P[0, ..., 255]$, a 32-bit variable $s$
Ensure: 64-bit random output
   /*Update and rotate $P$*/
1: swap ($P[0]$, $P[Y[185]\&255]$);
2: rotate ($P$);
   /* Update s*/
3: $s += Y[P[72]] - Y[P[239]]$;
4: $s = ROTL32(s, ((P[116] + 18)\&31))$;
   /* Output 8 bytes (least significant byte first)*/
5: output $((ROTL32(s, 25) \oplus Y[256]) + Y[P[26]])$;
6: output $((\quad s \quad \oplus Y[-1]) + Y[P[208]])$;
   /* Update and rotate $Y$*/
7: $Y[-3] = (ROTL32(s, 14) \oplus Y[-3]) + Y[P[153]]$;
8: rotate($Y$);
```

3 Notation and Convention

As Py uses different types of internal and external states (e.g. S-box, 32-bit integer) and they are updated every round, it is important to denote all the states and rounds in a simple but consistent way for easy understanding of our analysis. In every round of Py, we have the S-box P and the variable s getting updated before the output-words are generated in that round (see Algorithm 1). The other S-box, namely Y, gets updated after the output generation stage. Hence, in the beginning of any round i, we consider to have P_{i-1}, s_{i-1} but Y_i. At the end of the round, we have P_i, s_i and Y_{i+1}. If this convention is followed, we have P_i, s_i and Y_i in the formulas for the generation of the output-words in round i.

The nth element of arrays Y_i and P_i, are denoted by $Y_i[n]$ and $P_i[n]$ respectively. The jth bit of $Y_i[n]$, $P_i[n]$ and s_i are denoted by $Y_i[n]_{(j)}$, $P_i[n]_{(j)}$ and $s_{i(j)}$, respectively (following the convention that the least significant bit is the 0th bit). O_{lm} denotes the lth ($l \in \{1,2\}$) output-word generated in the mth round of Py. $O_{lm(j)}$ denotes the jth bit of O_{lm}. For example, $O_{13(5)}$ denotes the 5th bit of the 1st output-word of round 3. The output-word generated in line 5 of Algorithm 1 will be called the '1st output-word'. The output-word generated in line 6 of Algorithm 1 is described as the '2nd output-word'.

The '+' operator denotes *addition modulo* 2^{32} except when it is used to increment elements of P (particularly in expressions of the form $P_i[n] = P_j[m] + 1$, where '+' denotes *addition over* $\mathbb{Z}$). Similarly, the '-' and '$\oplus$' denote *subtraction modulo* 2^{32} and bitwise *exclusive-or.* $P[A]$ denotes the probability of occurrence of the event A.

4 Motivational Observation

Our main observation is that, if certain conditions on the elements of the S-box P are satisfied then the least significant bit (lsb) of the 1st output-word at the 1st round is equal to the lsb of the 2nd output-word at the 3rd round.

Theorem 1. $O_{11(0)} = O_{23(0)}$ *if the following six conditions on the elements of the S-box P are simultaneously satisfied.*

1. $P_2[116] \equiv -18 (\text{mod}\, 32)$ *(event A),*
2. $P_3[116] \equiv 7 (\text{mod}\, 32)$ *(event B),*
3. $P_2[72] = P_3[239] + 1$ *(event C),*
4. $P_2[239] = P_3[72] + 1$ *(event D),*
5. $P_1[26] = 1$ *(event E),*
6. $P_3[208] = 254$ *(event F).*

Proof. The formulas for the O_{11}, O_{23}, s_2 are given below (see Sect. 2).

$$O_{11} = (ROTL32(s_1, 25) \oplus Y_1[256]) + Y_1[P_1[26]], \quad (1)$$

$$O_{23} = (s_3 \oplus Y_3[-1]) + Y_3[P_3[208]], \quad (2)$$

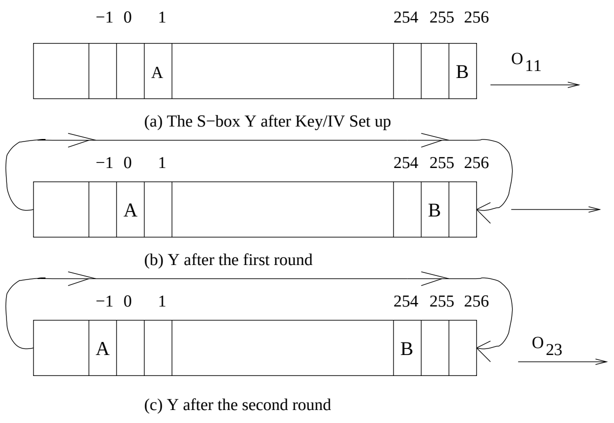
$$s_2 = ROTL32(s_1 + Y_2[P_2[72]] - Y_2[P_2[239]], ((P_2[116] + 18) \bmod 32)). \quad (3)$$

– Condition 1 (i.e., $P_2[116] \equiv -18 (\text{mod}\, 32)$) reduces (3) to

$$s_2 = s_1 + Y_2[P_2[72]] - Y_2[P_2[239]].$$

– Condition 2 (i.e., $P_3[116] \equiv 7 (\text{mod}\, 32)$) together with Condition 1 implies

$$s_3 = ROTL32((s_1 + Y_2[P_2[72]] - Y_2[P_2[239]] + Y_3[P_3[72]] - Y_3[P_3[239]]), 25).$$



−1 0 1 254 255 256
A
B
O11
(a) The S−box Y after Key/IV Set up
−1 0 1 254 255 256
A
B
(b) Y after the first round
−1 0 1 254 255 256
A
B
O23
(c) Y after the second round

Fig. 1. (a) $P_1[26] = 1$ (condition 5): A and B are used in O_{11}, (b) Y_2 (i.e., Y after 1^{st} round), (c) $P_3[208] = 254$ (condition 6): A and B are used in O_{23}

– Condition 3 and 4 (that is, $P_2[72] = P_3[239] + 1$ and $P_2[239] = P_3[72] + 1$) reduces the previous equation to

$$s_3 = ROTL32(s_1, 25). \tag{4}$$

From (1), (2), (4) we get,

$$O_{11} = (ROTL32(s_1, 25) \oplus Y_1[256]) + Y_1[P_1[26]], \tag{5}$$
$$O_{23} = (ROTL32(s_1, 25) \oplus Y_3[-1]) + Y_3[P_3[208]]. \tag{6}$$

In Fig. 1, conditions 5 and 6 are described. According to the figure,

$$A = Y_1[P_1[26]] = Y_3[-1] \tag{7}$$
$$B = Y_1[256] = Y_3[P_3[208]]. \tag{8}$$

Applying (7) and (8) in (5) and (6) we get,

$$O_{11(0)} \oplus O_{23(0)} = Y_1[256]_{(0)} \oplus Y_1[P_1[26]]_{(0)} \oplus Y_3[-1]_{(0)} \oplus Y_3[P_3[208]]_{(0)} = 0.$$

This completes the proof. □

5 Bias in the Distribution of the 1st and the 3rd Outputs

In this section, we shall compute $P[O_{11(0)} \oplus O_{23(0)} = 0]$ using the results of Sect. 4. We now recall the six events (or conditions) A, B, C, D, E, F as described in Theorem 1. First, we shall compute $P[A \cap B \cap C \cap D \cap E \cap F]$. We assume that the key set-up and the IV set-up algorithms of Py are perfect, and after the execution of them, the distributions of the permutation P, the elements of Y and the s are uniform and random.

Therefore, on a randomly chosen key/IV pair,

$$P[E] = \frac{1}{256}. \tag{9}$$

Next, we compute $P[A \cap E]$. Observe that event E, (i.e., $P_1[26] = 1$) implies $P_2[25] = 1$ or $P_2[255] = 1$. In each case, $P_2[116] \neq 1$. Therefore,

$$P[A|E] = \frac{8}{255}.$$

Hence, using Bayes' rule,

$$P[A \cap E] = P[A|E] \cdot P[E] = \frac{8}{255} \cdot \frac{1}{256}. \tag{10}$$

Next, we compute $P[A \cap B \cap E]$ using the following formula,

$$P[A \cap B \cap E] = P[B|A \cap E] \cdot P[A \cap E].$$

If $P_1[26] = 1$ then there are two cases: either $P_2[25] = 1$ or $P_2[255] = 1$. The first case implies $P_3[24] = 1$ or $P_3[255] = 1$ and the second case implies $P_3[254] = 1$ or $P_3[255] = 1$. Therefore, $P_3[116] \neq 1$. Also, $P_3[116] \neq P_2[116]$. Hence,

$$P[B|A \cap E] = \frac{8}{254}.$$

Therefore,

$$P[A \cap B \cap E] = \frac{8}{254} \cdot \frac{8}{255} \cdot \frac{1}{256}. \tag{11}$$

Next, we compute $P[A \cap B \cap C \cap E]$. We again note that if $P_1[26] = 1$ then either $P_2[25] = 1$ or $P_2[255] = 1$. The first case implies $P_3[24] = 1$ or $P_3[255] = 1$ and the second case implies $P_3[254] = 1$ or $P_3[255] = 1$. Proceeding in the same way, we observe that $P_2[72] \neq 1$ and $P_3[239] \neq 1$. Also by similar arguments, $P_3[239] \neq P_2[72]$, $P_3[239] \neq P_2[116]$ and $P_3[239] \neq P_3[116]$. Hence,

$$P[C|A \cap B \cap E] = \frac{251}{252 \cdot 253}.$$

Therefore, using (11),

$$\begin{aligned} P[A \cap B \cap C \cap E] &= P[C|A \cap B \cap E] \cdot P[A \cap B \cap E] \\ &= \frac{251}{252 \cdot 253} \cdot \frac{8}{254} \cdot \frac{8}{255} \cdot \frac{1}{256}. \end{aligned} \tag{12}$$

Similarly, we compute

$$P[A \cap B \cap C \cap D \cap E] = \frac{249}{250 \cdot 251} \cdot \frac{251}{252 \cdot 253} \cdot \frac{8}{254} \cdot \frac{8}{255} \cdot \frac{1}{256}. \tag{13}$$

Finally, it can be shown that,

$$\begin{aligned}P[A \cap B \cap C \cap D \cap E \cap F] &= \frac{1}{249} \cdot \frac{249}{250 \cdot 251} \cdot \frac{251}{252 \cdot 253} \cdot \frac{8}{254} \cdot \frac{8}{255} \cdot \frac{1}{256} \\ &\approx 2^{-41.91}. \end{aligned} \tag{14}$$

Under the assumption of randomness and uniformity of the distributions of the S-box elements and the s after the Key/IV set up, if any of the six events – described in Theorem 1 – does not occur then $P[O_{11(0)} \oplus O_{23(0)} = 0] = \frac{1}{2}$. That is,

$$P[O_{11(0)} \oplus O_{23(0)} = 0 | (A \cap B \cap C \cap D \cap E \cap F)^c] = \frac{1}{2}.$$

We denote the event $A \cap B \cap C \cap D \cap E \cap F$ by L and its compliment by L^c. Therefore,

$$\begin{aligned}P[O_{11(0)} \oplus O_{23(0)} = 0] &= P[O_{11(0)} \oplus O_{23(0)} = 0 | L] \cdot P[L] \\ &+ P[O_{11(0)} \oplus O_{23(0)} = 0 | L^c] \cdot P[L^c] \\ &= 1 \cdot 2^{-41.91} + \frac{1}{2} \cdot (1 - 2^{-41.91}) \\ &= \frac{1}{2} \cdot (1 + 2^{-41.91}). \end{aligned} \tag{15}$$

Note that, if Py had been a secure pseudorandom bit generator then the above probability would have been exactly $\frac{1}{2}$.

6 The Distinguisher

A *distinguisher* is an algorithm which distinguishes a stream of bits from a perfectly random stream of bits, that is, a stream of bits that has been chosen according to the uniform distribution. There are two ways a cryptanalyst may try to distinguish between a string, generated by an insecure pseudorandom bit generator, and one from a perfectly random source. In one case, she selects *only* one key randomly and produce keystream, seeded by the chosen key, long enough to detect a bias. In this scenario the attacker is “weak” as she has a keystream produced by a single key and therefore, the *distinguisher* is called a *weak distinguisher*. In the other case the adversary may use *any number of* randomly chosen keys and the respective keystreams generated by those keys. In this case the adversary is “strong” because she may collect outputs to her advantage from many keystreams to detect a bias. Therefore, the *distinguisher* so constructed is called a *strong distinguisher*. A bias present in the output at time t in a single stream may hardly be detected by a *weak distinguisher* but a *strong distinguisher* can easily discover the anomaly with a few bytes. This fact was exploited by Mantin and Shamir[3] to detect a strong bias toward zero in the second output byte of RC4.

The distinguishers that we construct in this section and Sect. 7, using the bias described in Sect. 5, are *strong distinguishers*. In Sect. 8 we have constructed a *weak distinguisher*. Here, essentially what we do is collect many pairs of outputs (i.e., outputs in the first and the third rounds), each generated by a randomly chosen key/IV and finally show that distribution of the pair is not uniform on a randomly chosen key/IV (this fact is already proved in Theorem 1). In the rest of the section we address the question as to how many randomly key/IV's are required to establish the distinguisher. To this end we use a corollary of a theorem used in [3] (see the paper for the proof).

Corollary 1. *If an event e occurs in a distribution X with probability p and in Y with probability $p(1+q)$ then, if $p = \frac{1}{2}$, $O(\frac{1}{q^2})$ samples are required to distinguish X from Y with non-negligible probability of success.*

Proof. Let X_e and Y_e denote the random variables specifying the number of occurrences of e in t samples. Then X_e and Y_e have binomial distributions with parameters (t, p) and $(t, p(1+q))$ respectively. The expectations of X_e and Y_e are denoted by $E[X_e]$ and $E[Y_e]$, their variances by $V(X_e)$ and $V(Y_e)$ and their standard deviations by $\sigma(X_e)$ and $\sigma(Y_e)$ respectively.

$$E[X_e] = tp, E[Y_e] = tp(1+q)$$
$$V(X_e) = tp(1-p), V(Y_e) = tp(1+q)(1-p(1+q))$$
$$\sigma(X_e) = \sqrt{V(X_e)} = \sqrt{tp(1-p)}, \sigma(Y_e) = \sqrt{V(Y_e)} = \sqrt{tp(1+q)(1-p(1+q))}$$

Here we determine the size of t that ensures a difference of at least one standard deviation between the expectations of the two distributions: $E[Y_e] - E[X_e] \geq \sigma(X_e) \implies tp(1+q) - tp \geq \sqrt{tp(1-p)} \implies \frac{tq}{2} \geq \frac{\sqrt{t}}{2}$ (as $p = \frac{1}{2}$) $\implies t \geq \frac{1}{q^2}$. Hence, $O(\frac{1}{q^2})$ samples (the constant increases in tune with the desired success probability) are enough to establish the distinguishing attack. This proves the corollary. □

In our case, X, Y and e are the distribution of (O_{11}, O_{23}) collected from a perfectly random source, the distribution of these variables from Py and the event that $O_{11(0)} \oplus O_{23(0)} = 0$. Thus, $p = \frac{1}{2}$ and $q = \frac{1}{2^{41.91}}$. Therefore, to establish the distinguisher with non-negligible success probability, the minimum number of required samples (i.e., the number of key/IV's) is $t = \frac{1}{(2^{41.91})^2} = 2^{83.82}$. This results show that the designers' claim that distinguishing attacks on Py is not possible with running time less than the exhaustive search is not true.

7 Biases among other Pairs of Bits and Distinguishers

In Sect. 5, we showed a bias in $(O_{11(0)}, O_{23(0)})$. In this section, we show that the bias is present in $(O_{11(i)}, O_{23(i)})$ for all i, where $0 \leq i \leq 31$, however, the amount of bias is reduced as i increases. From (1) and (2), we get,

$$O_{11(i)} = ROTL32(s_1, 25)_{(i)} \oplus Y_1[256]_{(i)} \oplus Y_1[P_1[26]]_{(i)} \oplus c_{1(i)},$$
$$O_{23(i)} = s_{3(i)} \oplus Y_3[-1]_{(i)} \oplus Y_3[P_3[208]]_{(i)} \oplus c_{3(i)}$$

where $0 \le i \le 31$ and c_1, c_3 are the carry terms in (1) and (2) respectively. If all the 6 conditions of Theorem 1 are satisfied, then O_{11} and O_{23} can be written in the following form (see Theorem 1)

$$O_{11} = (S \oplus B) + A, \tag{16}$$
$$O_{23} = (S \oplus A) + B \tag{17}$$

which implies that

$$O_{11(i)} \oplus O_{23(i)} = c_{1(i)} \oplus c_{3(i)}.$$

In this section we give a general expression to compute

$$P[O_{11(i)} \oplus O_{23(i)} = 0], \qquad 0 \le i \le 31.$$

Note that

$$\begin{aligned} P[O_{11(i)} \oplus O_{23(i)} = 0] &= P[O_{11(i)} \oplus O_{23(i)} = 0|L] \cdot P[L] \\ &+ P[O_{11(i)} \oplus O_{23(i)} = 0|L^c] \cdot P[L^c] \\ &= \underbrace{P[c_{1(i)} \oplus c_{3(i)} = 0|L]}_{p_i} \cdot P[L] \\ &+ \underbrace{P[O_{11(i)} \oplus O_{23(i)} = 0|L^c]}_{X_i} \cdot P[L^c] \end{aligned} \tag{18}$$

where the event L is $A \cap B \cap C \cap D \cap E \cap F$ and $0 \le i \le 31$.

From (14) we can calculate $P[L]$ and $P[L^c]$. Now we compute $P[c_{1(i)} \oplus c_{3(i)} = 0|L]$ (denoted by p_i in (18), similarly p_{i-1} should be understood) recursively. We consider the bits of the variables A, B, S in (16) and (17) in Table 1 (note that these variables are uniformly distributed and independent). From Table 1, using Bayes' rule, we obtain the following recursion to compute p_i,

$$p_i = \frac{p_{i-1}}{2} + \frac{1}{4}.$$

We already know $p_0 = 1$ (i.e., $P[O_{11(0)} \oplus O_{23(0)} = 0|L] = 1$). Therefore, using the above recurrence relation, finally we get,

$$p_i = \frac{1}{2} + \frac{1}{2^{i+1}}, \quad 0 \le i \le 31. \tag{19}$$

As shown in (18), $X_i = \frac{1}{2}$ for all i's except when $i = 25$ (we omit the details of the proof).[1] It can be shown that X_{25} is marginally greater than $\frac{1}{2}$. Therefore, (18) and (19) imply,

$$P[O_{11(i)} \oplus O_{23(i)} = 0] > \frac{1}{2}.$$

[1] The case when (i)$P_2[72] = P_3[72] + 1$, (ii)$P_2[239] = P_3[239] + 1$ and the conditions 1,2,5,6 of Theorem 1 are satisfied, is responsible for the anomalous behavior at $i = 25$.

Table 1. Truth table for computing p_i using p_{i-1}. The last column in each row indicates the probability of the occurrence of that row

$c_{1(i-1)} \oplus c_{3(i-1)}$	$S_{(i-1)}$	$B_{(i-1)}$	$A_{(i-1)}$	$c_{1(i)} \oplus c_{3(i)}$	Probability
0	0	0	0	0	$\frac{p_{i-1}}{8}$
0	0	0	1	0	$\frac{p_{i-1}}{8}$
0	0	1	0	0	$\frac{p_{i-1}}{8}$
0	0	1	1	0	$\frac{p_{i-1}}{8}$
0	1	0	0	0	$\frac{p_{i-1}}{8}$
0	1	0	1	1	
0	1	1	0	1	
0	1	1	1	0	$\frac{p_{i-1}}{8}$
1	0	0	0	0	$\frac{1-p_{i-1}}{8}$
1	0	0	1	1	
1	0	1	0	1	
1	0	1	1	0	$\frac{1-p_{i-1}}{8}$
1	1	0	0	1	
1	1	0	1	1	
1	1	1	0	1	
1	1	1	1	1	

One can note that $p_{i=0}$ attains the maximum value 1 (see (19)). In Sect. 5 and 6, we used $p_{i=0} = 1$ to construct our distinguisher. The above equations suggest that many distinguishers can be generated using different $(O_{11(i)}, O_{23(i)})$'s rather than only $(O_{11(0)}, O_{23(0)})$, however, the amount of bias decreases as i increases (i.e., we get the most effective distinguisher when $i = 0$ as described in Sect. 6). For example, when $i = 1$, using $p_1 = \frac{3}{4}$ from (19),

$$\begin{aligned}
P[O_{11(1)} \oplus O_{23(1)} = 0] &= P[O_{11(1)} \oplus O_{23(1)} = 0|L] \cdot P[L] \\
&\quad + P[O_{11(1)} \oplus O_{23(1)} = 0|L^c] \cdot P[L^c] \\
&= p_1 \cdot P[L] + P[O_{11(1)} \oplus O_{23(1)} = 0|L^c] \cdot P[L^c] \\
&= \frac{3}{4} \cdot 2^{-41.91} + \frac{1}{2} \cdot (1 - 2^{-41.91}) \\
&= \frac{1}{2} \cdot (1 + 2^{-42.91}).
\end{aligned}$$

Now, for the above case, i.e., taking the 1st and the 3rd bits of O_{11} and O_{23}, using Corollary 1, the minimum number of samples (i.e., the number of key/IV's) required to establish a distinguisher is $2^{85.82}$ (note that in Sect. 6, we considered 0th bits of O_{11} and O_{23} and the number of samples was $2^{83.82}$). Similarly, if we consider $i = 2$ then the number of required samples is $2^{87.82}$.

8 Existence of Bias at t and $t+2$ rounds and a Distinguisher

We assert that all the above results are valid if we consider any rounds t and $t+2$ instead of just rounds 1 and 3. In other words, instead of $(O_{11(0)}, O_{23(0)})$, one can show that the bias exists even in the distribution of $(O_{1,t(i)}, O_{2,(t+2)(i)})$. Now, we state a theorem which is the generalized version of Theorem 1.

Theorem 2. $O_{1,t(0)} = O_{2,(t+2)(0)}$ *if the following six conditions on the elements of the S-box P are simultaneously satisfied.*

1. $P_{t+1}[116] \equiv -18 (\mathrm{mod}\, 32)$,
2. $P_{t+2}[116] \equiv 7 (\mathrm{mod}\, 32)$,
3. $P_{t+1}[72] = P_{t+2}[239] + 1$,
4. $P_{t+1}[239] = P_{t+2}[72] + 1$,
5. $P_t[26] = 1$,
6. $P_{t+2}[208] = 254$.

Using the above theorem and the techniques used before, it is easy to show (see (15)),

$$P[O_{1,t(0)} \oplus O_{2,(t+2)(0)} = 0] = \frac{1}{2}(1 + 2^{-41.91}).$$

The fact that the above probability is valid $\forall t > 0$, allows us to generate a *weak distinguisher* with number of rounds $2^{83.82}$ (see Sect. 6 for a definition of a weak distinguisher). This means that $2^{83.82} \times 2^3 = 2^{86.82}$ bytes of a single stream generated by a randomly chosen key is sufficient to distinguish Py from random. However, the amount of bytes falls outside the allowable stream length of Py (which is 2^{64} bytes).

9 Conclusion and Remarks

In this paper, for the first time, several weaknesses on the stream cipher Py have been presented. We presented a class of distinguishers on the stream cipher Py, the best of which works with $2^{83.82}$ random key/IV's which is better than the exhaustive search (for the cipher the recommended key length is 256 bits). We also showed that the output stream of Py of recommended length of 2^{64} bytes, contains biases at different points on it. These results break the cipher Py academically. However, the time complexity of the best distinguishing attack mentioned in this paper is far beyond the reach of the fastest machine available. Therefore, these weaknesses pose no practical threat to the security of the cipher at this moment. However, the shortened version of Py, known as Py6, may contain more serious weaknesses than the ones described here, but the complete description of Py6 is not provided by the designers.

10 Acknowledgments

The work has been done when the first author was visiting the research group COSIC at the Electrical Engineering Department of Katholieke Universiteit Leuven, Belgium.

References

1. E. Biham, J. Seberry, "Py (Roo): A Fast and Secure Stream Cipher using Rolling Arrays," *ecrypt submission*, 2005.
2. Ecrypt, `http://www.ecrypt.eu.org`.
3. I. Mantin, A. Shamir, "A Practical Attack on Broadcast RC4," *Fast Software Encryption 2001* (M. Matsui, ed.), vol. 2355 of *LNCS*, pp. 152-164, Springer-Verlag, 2001.
4. NESSIE: New European Schemes for Signature, Integrity and Encryption `http://www.cryptonessie.org`.