# New Attacks on the Stream Cipher TPy6 and Design of New Ciphers the TPy6-A and the TPy6-B[*]

Gautham Sekar, Souradyuti Paul, and Bart Preneel

Katholieke Universiteit Leuven, Dept. ESAT/COSIC,
Kasteelpark Arenberg 10,
B–3001, Leuven-Heverlee, Belgium
{gautham.sekar, souradyuti.paul, bart.preneel}@esat.kuleuven.be

**Abstract.** The stream ciphers Py, Pypy and Py6 were designed by Biham and Seberry for the ECRYPT-eSTREAM project in 2005. The ciphers were promoted to the 'Focus' ciphers of the Phase II of the eSTREAM project. However, due to some cryptanalytic results on the ciphers, strengthened versions of the ciphers, namely TPy, TPypy and TPy6 were built. So far there exists no attacks on TPy6. In this paper, we find hitherto unknown weaknesses in the keystream generation algorithms of the Py6 and of its stronger variant TPy6. Exploiting these weaknesses, a large number of distinguishing attacks are mounted on the ciphers, the best of which works with $2^{224.6}$ data and comparable time. In the second part, we present two new ciphers derived from the TPy6, namely TPy6-A and TPy6-B, whose performances are 2.65 cycles/byte and 4.4 cycles/byte on Pentium III. As a result, to the best of our knowledge, on Pentium platforms TPy6-A becomes the fastest stream cipher in the literature. Based on our security analysis, we conjecture that no attacks better than brute force are possible on the ciphers TPy6-A and TPy6-B.

## 1 Introduction

At first, we recall a brief history of the Py-family of ciphers.

**Timeline: the Py-family of Ciphers**

- **April 2005.** The ciphers Py and Py6, designed by Biham and Seberry, were submitted to the ECRYPT project for analysis and evaluation in the category of software based stream ciphers [3]. The impressive speed of the cipher Py in software (about 2.5 times faster than the RC4) made it one of the fastest and most attractive contestants.
- **March 2006 (at FSE 2006).** Paul, Preneel and Sekar reported distinguishing attacks with $2^{89.2}$ data and comparable time against the cipher Py [9]. Crowley [5] later reduced the complexity to $2^{72}$ by employing a Hidden Markov Model.
- **March 2006 (at the Rump session of FSE 2006).** A new cipher, namely Pypy, was proposed by the designers to rule out the aforementioned distinguishing attacks on Py [4].
- **May 2006 (presented at Asiacrypt 2006).** Distinguishing attacks were reported against Py6 with $2^{68.6}$ data and comparable time by Paul and Preneel [10].
- **October 2006 (presented at Eurocrypt 2007).** Wu and Preneel showed key recovery attacks against the ciphers Py, Pypy, Py6 with chosen IVs [15]. This attack was subsequently improved by Isobe *et al.* [8].
- **January 2007.** Three new ciphers TPypy, TPy, TPy6 were proposed by the designers [2]; the ciphers can very well be viewed as the strengthened versions of the previous ciphers Py, Pypy and Py6 where the above attacks do not apply.

[*] The first author is supported by an IWT SoBeNeT project. The second author is funded by the IBBT (Interdisciplinary Institute for BroadBand Technology), a research institute founded by the Flemish Government in 2004. This is the full version of the paper published in the proceedings of WEWoRC 2007. A minor typographical error has been rectified here.

– **February 2007.** Sekar *et al.* published attacks on TPy and TPypy, each of which requires $2^{281}$ data and comparable time [11].
– **August 2007 (presented at SAC 2007).** Tsunoo *et al.* showed a distinguishing attack on TPypy with a data complexity of $2^{199}$ [14].
– **October 2007 (to be presented at ISC 2007).** Sekar *et al.* showed attacks on TPy, the best of which requires $2^{268.6}$ data and comparable time [12]. So far there exist no attacks on the cipher TPy6.
– **December 2007 (to be presented at Indocrypt 2007).** Sekar *et al.* showed related-key attacks on Py, Pypy, TPy and TPypy, each requiring $2^{193.7}$ data and comparable time [13].

**Algorithm 1** The keystream generation algorithm of TPy6 (and Py6)

```
Require: $Y[-3,...,64]$, $P[0,...,63]$, a 32-bit variable $s$
Ensure: 64-bit random output
    /*Update and rotate $P$*/
 1: swap ($P[0]$, $P[Y[43]\&63]$);
 2: rotate ($P$);
    /* Update s*/
 3: $s+=Y[P[18]]-Y[P[57]]$;
 4: $s=ROTL32(s,((P[26]+18)\&31))$;
    /* Output 4 or 8 bytes (least significant byte first)*/
 5: output $((ROTL32(s,25)\oplus Y[64])+Y[P[8]])$;
 6: output $((\quad s \quad \oplus Y[-1])+Y[P[21]])$;
    /* Update and rotate $Y$*/
 7: $Y[-3]=(ROTL32(s,14)\oplus Y[-3])+Y[P[48]]$;
 8: rotate($Y$);
```

## 2 Notation and Convention

– The $m$th bit ($m=0$ denotes the least significant bit or lsb) of the first output-word generated at round $n$ is denoted by $O_{n(m)}$. The second output-word is not used anywhere in our analysis.
– $P_n$, $Y_{n+1}$ and $s_n$ are the inputs to the algorithm at round $n$. When this convention is followed, we see that $O_n=(ROTL32(s_n,25)\oplus Y_n[64])+Y_n[P_n[26]]$- the index '$n$' is maintained throughout the expression.
– The $ROTL32(x,k)$ denotes that the variable $x$ is cyclically rotated to the left by $k$ bits.
– $Y_n[m]$, $P_n[m]$ denote the $m$th elements of array $Y_n$ and $P_n$ respectively.
– $Y_n[m]_i$, $P_n[m]_i$ denote the $i$th bit ($i=0$ denotes the lsb) of $Y_n[m]$, $P_n[m]$ respectively.
– The symbol '&' denotes the *and* operator.
– The operators '+' and '−' denote *addition modulo* $2^{32}$ and *subtraction modulo* $2^{32}$ respectively, except when used with expressions which relate two elements of array $P$. In this case they denote *addition and subtraction over* $\mathbb{Z}$.
– The symbol '$\oplus$' denotes bitwise *exclusive-or* and $\bigcap$ denotes set intersection.
– In $O_{n(i)}$, $s_{n(i)}$ and $Y_n[P_m[X]]_i$, the index representing bit position, i.e., $i$ denotes $i$ mod 32.
– $Y^c_n[P_m[X]]_i$ denotes the complement of $Y_n[P_m[X]]_i$.
– The pseudorandom bit generation algorithm of a stream cipher is denoted by PRBG.

## 3 Distinguishing attacks on the Py6 and the TPy6

We detect a large number of input-output correlations of TPy6 and Py6 that allow us to build distinguishers. The weak states which lead to the best distinguishing attack are outlined in the following theorem.

**Theorem 1.** $O_{1(i)} \oplus O_{3(i+7)} \oplus O_{7(i+7)} \oplus O_{8(i+7)} = 0$ *if the following* 17 *conditions are simultaneously satisfied.*
*1.* $P_1[26] \equiv -18 \bmod 32$ *(event* $E_1$*), 2.* $P_2[26] \equiv 7 \bmod 32$ *(event* $E_2$*), 3.* $P_3[26] \equiv -4 \bmod 32$ *(event* $E_3$*), 4.* $P_7[26] \equiv 3 \bmod 32$ *(event* $E_4$*),*
*5.* $P_8[26] \equiv 3 \bmod 32$ *(event* $E_5$*), 6.* $P_1[18] = P_2[57] + 1$ *(event* $E_6$*), 7.* $P_1[57] = P_2[18] + 1$ *(event* $E_7$*), 8.* $P_7[18] = P_8[18] + 1$ *(event* $E_8$*),*
*9.* $P_7[57] = P_8[57] + 1$ *(event* $E_9$*), 10.* $P_3[18] = 62$ *(event* $E_{10}$*), 11.* $P_1[8] = P_3[57] + 2$ *(event* $E_{11}$*), 12.* $P_1[18] = 3$ *(event* $E_{12}$*),*
*13.* $P_3[8] = 0$ *(event* $E_{13}$*), 14.* $P_1[57] = P_7[8] + 6$ *(event* $E_{14}$*), 15.* $P_7[48] = 60$ *(event* $E_{15}$*), 16.* $P_6[48] = P_8[8] + 2$ *(event* $E_{16}$*),*
*17.* $d_{7(i-7)} \oplus d_{8(i-7)} \oplus c_{1(i)} \oplus d_{3(i)} \oplus d_{1(i+7)} \oplus c_{3(i+7)} \oplus c_{7(i+7)} \oplus e_{7(i+7)} \oplus c_{8(i+7)} \oplus e_{8(i+7)} = 0.$[1]

**Proof.** First, we state and prove two lemmata which will be used to establish the theorem.

**Lemma 1.** *If*
*1.* $P_1[26] \equiv -18 \bmod 32$*, 2.* $P_3[26] \equiv -4 \bmod 32$*, 3.* $P_7[26] \equiv 3 \bmod 32$*, 4.* $P_8[26] \equiv 3 \bmod 32$
*then the following equations are satisfied:*

$$\begin{aligned}
1.\ O_{1(i)} &= s_{0(i+7)} \oplus Y_1[P_1[18]]_{i+7} \oplus Y_1^c[P_1[57]]_{i+7} \oplus Y_1[256]_i \oplus Y_1[P_1[8]]_i \\
&\quad \oplus c_{1(i)} \oplus d_{1(i+7)}, \\
2.\ O_{3(i+7)} &= s_{2(i)} \oplus Y_3[P_3[18]]_i \oplus Y_3^c[P_3[57]]_i \oplus Y_3[256]_{i+7} \oplus Y_3[P_3[8]]_{i+7} \\
&\quad \oplus c_{3(i+7)} \oplus d_{3(i)}, \\
3.\ O_{7(i+7)} &= Y_7[P_7[18]]_{i-7} \oplus Y_7^c[P_7[57]]_{i-7} \oplus Y_6[-3]_{i+7} \oplus Y_7[P_7[8]]_{i+7} \\
&\quad \oplus Y_6[P_6[48]]_{i+7} \oplus c_{7(i+7)} \oplus d_{7(i-7)} \oplus e_{7(i+7)}, \\
4.\ O_{8(i+7)} &= Y_8[P_8[18]]_{i-7} \oplus Y_8^c[P_8[57]]_{i-7} \oplus Y_7[-3]_{i+7} \oplus Y_8[P_8[8]]_{i+7} \\
&\quad \oplus Y_7[P_7[48]]_{i+7} \oplus c_{8(i+7)} \oplus d_{8(i-7)} \oplus e_{8(i+7)}.
\end{aligned}$$

**Proof.** From Figure 1, we get

$$Y_n[i] = Y_{n+1}[i-1] \tag{1}$$

when $-2 \le i \le 64$. When $i = -3$,

$$Y_{n+1}[64] = (ROTL32(s_i, 14) \oplus Y_n[-3]) + Y_n[P_n[48]].$$

Generalizing (1), we have

$$Y_n[i] = Y_{n+k}[i-k] \tag{2}$$

when $-3 \le i - k \le 63$. Line 5 of Algorithm 1 gives

$$O_7 = (ROTL32(s_7, 25) \oplus Y_7[64]) + Y_7[P_7[8]]. \tag{3}$$

Let the $c_7$ denote the carry in the above equation. Since $ROTL32(s_7, 25)_i = s_{7(i-25 \bmod 32)}$,

$$O_{7(i)} = s_{7(i-25 \bmod 32)} \oplus Y_7[64]_i \oplus Y_7[P_7[8]]_i \oplus c_{7(i)}. \tag{4}$$

Lines 3 and 4 of Algorithm 1 give us

$$s_7 = ROTL32(s_6 + Y_7[P_7[18]] - Y_7[P_7[57]], P_7[26] + 18 \bmod 32) \tag{5}$$

$$\begin{aligned}
\Rightarrow s_{7(j)} &= s_{6(j-k \bmod 32)} \oplus Y_7[P_7[18]]_{j-k \bmod 32} \oplus Y_7^c[P_7[57]]_{j-k \bmod 32} \\
&\quad \oplus d_{7(j-k \bmod 32)}
\end{aligned} \tag{6}$$

where $k = P_7[26] + 18 \bmod 32$, $d_{7(i)} = f_{7(i)} \oplus g_{7(i)}$ and $d_{7(0)} = 1$ ($f_7$ and $g_7$ are the carry terms in (5) which are explained in Sect. 4.2). For simplicity, henceforth we denote $X_{(i \bmod 32)}$ by $X_{(i)}$. Thus (6) becomes,

$$s_{7(j)} = s_{6(j-k)} \oplus Y_7[P_7[18]]_{j-k} \oplus Y_7^c[P_7[57]]_{j-k} \oplus d_{7(j-k)}. \tag{7}$$

[1] The terms $c$, $d$, $e$ are the carries generated in certain expressions, the descriptions of which can be found in the proof of Theorem 1.

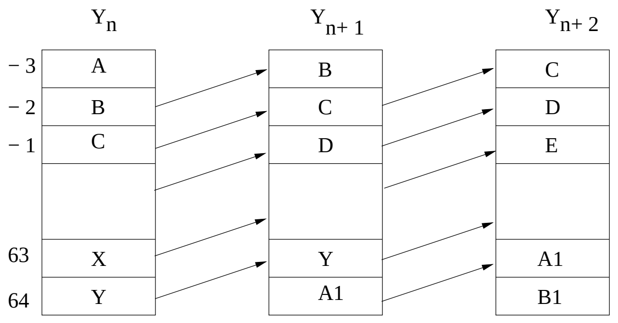

**Fig. 1.** The figure shows the update of the S-box $Y$. $Y_n[i] = Y_{n+1}[i-1]$ when $-2 \le i \le 64$. $Y_{n+1}[64] = A1$ when $i = -3$ and $A1 = (ROTL32(s_n, 14) \oplus A) + Y_n[P_n[48]]$. Generalizing the above, we can write $Y_n[i] = Y_{n+k}[i-k]$ when $-3 \le i-k \le 63$.

If $j = i - 25 \mod 32$, then (7) becomes

$$s_{7(i-25)} = s_{6(i-k-25)} \oplus Y_7[P_7[18]]_{i-k-25} \oplus Y_7^c[P_7[57]]_{i-k-25} \oplus d_{7(i-k-25)}. \tag{8}$$

Substituting (8) in (4), we get,

$$O_{7(i)} = s_{6(i-k-25)} \oplus Y_7[P_7[18]]_{i-k-25} \oplus Y_7^c[P_7[57]]_{i-k-25} \oplus Y_7[64]_i \\ \oplus Y_7[P_7[8]]_i \oplus c_{7(i)} \oplus d_{7(i-k-25)}. \tag{9}$$

Next, we have

$$Y_7[64] = (ROTL32(s_6, 14) \oplus Y_6[-3]) + Y_6[P_6[48]], \tag{10}$$

$$Y_7[64]_i = s_{6(i-14)} \oplus Y_6[-3]_i \oplus Y_6[P_6[48]]_i \oplus e_{7(i)} \tag{11}$$

where $e_7$ is the carry term in (10). Substituting (11) in (9), we get,

$$O_{7(i)} = s_{6(i-k-25)} \oplus s_{6(i-14)} \oplus Y_7[P_7[18]]_{i-k-25} \oplus Y_7^c[P_7[57]]_{i-k-25} \oplus Y_6[-3]_i \\ \oplus Y_7[P_7[8]]_i \oplus Y_6[P_6[48]]_i \oplus c_{7(i)} \oplus d_{7(i-k-25)} \oplus e_{7(i)}. \tag{12}$$

Now, if $k = -11$ (i.e., $k \equiv -11 \mod 32 \Rightarrow P_7[26] + 18 \equiv -11 \mod 32 \Rightarrow P_7[26] \equiv 3 \mod 32$) then $s_{6(i-k-25)} \oplus s_{6(i-14)} = 0$. Hence, when $P_7[26] \equiv 3 \mod 32$, (12) becomes

$$O_{7(i)} = Y_7[P_7[18]]_{i-14} \oplus Y_7^c[P_7[57]]_{i-14} \oplus Y_6[-3]_i \oplus Y_7[P_7[8]]_i \\ \oplus Y_6[P_6[48]]_i \oplus c_{7(i)} \oplus d_{7(i-14)} \oplus e_{7(i)}. \tag{13}$$

By similar arguments, when $P_8[26] \equiv 3 \mod 32$,

$$O_{8(i)} = Y_8[P_8[18]]_{i-14} \oplus Y_8^c[P_8[57]]_{i-14} \oplus Y_7[-3]_i \oplus Y_8[P_8[8]]_i \\ \oplus Y_7[P_7[48]]_i \oplus c_{8(i)} \oplus d_{8(i-14)} \oplus e_{8(i)}. \tag{14}$$

From (9), we get

$$O_{1(i)} = s_{0(i-k-25)} \oplus Y_1[P_1[18]]_{i-k-25} \oplus Y_1^c[P_1[57]]_{i-k-25} \oplus Y_1[64]_i \\ \oplus Y_1[P_1[8]]_i \oplus c_{1(i)} \oplus d_{1(i-k-25)}. \tag{15}$$

When $k = 0$ (i.e., $P_1[26] \equiv -18 \mod 32$), the above equation reduces to

$$O_{1(i)} = s_{0(i+7)} \oplus Y_1[P_1[18]]_{i+7} \oplus Y_1^c[P_1[57]]_{i+7} \oplus Y_1[64]_i \oplus Y_1[P_1[8]]_i \\ \oplus c_{1(i)} \oplus d_{1(i+7)}. \tag{16}$$

Similarly, when $P_3[26] \equiv -4 \bmod 32$, we have

$$\begin{aligned} O_{3(i+7)} = {} & s_{2(i)} \oplus Y_3[P_3[18]]_i \oplus Y_3^c[P_3[57]]_i \oplus Y_3[64]_{i+7} \oplus Y_3[P_3[8]]_{i+7} \\ & \oplus c_{3(i+7)} \oplus d_{3(i)}. \end{aligned} \tag{17}$$

From (13) and (14), we derive the following results:

$$\begin{aligned} O_{7(i+7)} = {} & Y_7[P_7[18]]_{i-7} \oplus Y_7^c[P_7[57]]_{i-7} \oplus Y_6[-3]_{i+7} \oplus Y_7[P_7[8]]_{i+7} \\ & \oplus Y_6[P_6[48]]_{i+7} \oplus c_{7(i+7)} \oplus d_{7(i-7)} \oplus e_{7(i+7)}, \end{aligned} \tag{18}$$

$$\begin{aligned} O_{8(i+7)} = {} & Y_8[P_8[18]]_{i-7} \oplus Y_8^c[P_8[57]]_{i-7} \oplus Y_7[-3]_{i+7} \oplus Y_8[P_8[8]]_{i+7} \\ & \oplus Y_7[P_7[48]]_{i+7} \oplus c_{8(i+7)} \oplus d_{8(i-7)} \oplus e_{8(i+7)}. \end{aligned} \tag{19}$$

This completes the proof. □

Now we state the second lemma.

**Lemma 2.** $s_{0(i+7)} = s_{2(i)}$ *if the following conditions are simultaneously satisfied,*

1. $P_1[26] \equiv -18 \bmod 32$,
2. $P_2[26] \equiv 7 \bmod 32$,
3. $P_1[18] = P_2[57] + 1$,
4. $P_1[57] = P_2[18] + 1$.

**Table 1.** Terms generated in $O_{1(i)} \oplus O_{3(i+7)} \oplus O_{7(i+7)} \oplus O_{8(i+7)}$, given the events $E_1$ to $E_7$ simultaneously occur (the terms are grouped by their bit positions)

| **Bit position:** $i-7$ | **Bit position:** $i$ | **Bit position:** $i+7$ |
|---|---|---|
| $Y_7[P_7[18]]$ | $Y_1[64]$ | $Y_1[P_1[18]]$ |
| $Y_7[P_7[57]]$ | $Y_1[P_1[8]]$ | $Y_1[P_1[57]]$ |
| $Y_8[P_8[18]]$ | $Y_3[P_3[18]]$ | $Y_3[256]$ |
| $Y_8[P_8[57]]$ | $Y_3[P_3[57]]$ | $Y_3[P_3[8]]$ |
| Carries | Carries | $Y_6[P_6[48]]$ |
| | | $Y_6[-3]$ |
| | | $Y_7[P_7[8]]$ |
| | | $Y_7[P_7[48]]$ |
| | | $Y_7[-3]$ |
| | | $Y_8[P_8[8]]$ |
| | | Carries |

**Proof.** Equation (5) gives us:

$$s_1 = ROTL32(s_0 + Y_1[P_1[18]] - Y_1[P_1[57]], P_1[26] + 18 \bmod 32).$$

The first condition ($P_1[26] \equiv -18 \bmod 32$) reduces this to

$$s_1 = s_0 + Y_1[P_1[18]] - Y_1[P_1[57]].$$

Therefore,

$$\begin{aligned} s_2 = ROTL32(& s_0 + Y_2[P_2[18]] - Y_2[P_2[57]] + Y_1[P_1[18]] - Y_1[P_1[57]], \\ & P_2[26] + 18 \bmod 32). \end{aligned}$$

Conditions 3 and 4, when used with (1), reduce the above equation to

$$s_2 = ROTL32(s_0, P_2[26] + 18 \bmod 32).$$

Finally, with condition 2 (i.e., $P_2[26] \equiv 7 \bmod 32$), the previous equation becomes

$$\begin{aligned} s_2 &= ROTL32(s_0, 25) \\ \Rightarrow s_{2(i)} &= ROTL32(s_0, 25)_i = s_{0(i-25)} \\ &= s_{0(i+7)}. \end{aligned} \tag{20}$$

This completes the proof. □

Now we observe that, when the conditions listed under (i) Lemma 1 (i.e., events $E_1$, $E_3$, $E_4$ and $E_5$) and (ii) Lemma 2 (i.e., events $E_1$, $E_2$, $E_6$ and $E_7$) are simultaneously satisfied, then the expression $O_{1(i)} \oplus O_{3(i+7)} \oplus O_{7(i+7)} \oplus O_{8(i+7)}$ is the XOR of the terms which are listed in Table 1 (grouped according to the bit positions).[2] Similarly, the 'carries' in Table 1 are elaborated in Table 2.

**Table 2.** Carry terms generated in $O_{1(i)} \oplus O_{3(i+7)} \oplus O_{7(i+7)} \oplus O_{8(i+7)}$ grouped by their bit positions

| **Bit position:** $i-7$ | **Bit position:** $i$ | **Bit position:** $i+7$ |
|---|---|---|
| $d_7$ | $c_1$ | $d_1$ |
| $d_8$ | $d_3$ | $c_3$ |
| | | $c_7$ |
| | | $e_7$ |
| | | $c_8$ |
| | | $e_8$ |

If the $Y$-terms in Table 1 are pairwise equated (this is achieved when the events $E_8$ through to $E_{16}$ occur) then we get

$$\begin{aligned} O_{1(i)} \oplus O_{3(i+7)} \oplus O_{7(i+7)} \oplus O_{8(i+7)} = &\, d_{7(i-7)} \oplus d_{8(i-7)} \oplus c_{1(i)} \oplus d_{3(i)} \oplus d_{1(i+7)} \\ &\oplus c_{3(i+7)} \oplus c_{7(i+7)} \oplus e_{7(i+7)} \oplus c_{8(i+7)} \\ &\oplus e_{8(i+7)}. \end{aligned} \tag{21}$$

Now, when the RHS of (21) equals zero (i.e., $E_{17}$ occurs) we get

$$O_{1(i)} \oplus O_{3(i+7)} \oplus O_{7(i+7)} \oplus O_{8(i+7)} = 0.$$

This completes the proof. □

## 4 Computation of the Bias

In this section, we quantify the bias in the outputs of TPy6 (and hence Py6) induced by the fortuitous events similar to the one described in Sect. 3. Now it is important to note that there may be *more than one set of 17 conditions* possible, where each of them results in $O_{1(i)} \oplus O_{3(i+7)} \oplus O_{7(i+7)} \oplus O_{8(i+7)} = 0$ (let us assume that there are $n$ such sets). In Theorem 1, we listed one such set. Our experiments suggest that these $n$ sets are mutually independent, however, a formal proof of that is nontrivial.

Each of the events $E_1$ to $E_5$ occurs with approximate probability $\frac{1}{32}$ and each of the events $E_6$ to $E_{16}$ occurs with probability which is approximately $\frac{1}{64}$. Let $p$ denote the probability that condition 17 is satisfied. Let $F$ denote the event $\bigcap_{j=1}^{16} E_j$. Therefore,

$$P[F] = (\frac{1}{32})^5 \cdot (\frac{1}{64})^{11}.$$

[2] Note that none of the terms listed in Table 1 is of the form $A^c$ because we used the fact that $A^c \oplus B^c = A \oplus B$ in (16), (17), (18) and (19).

We see that there are $n$ $F$-like events (i.e., the intersection of 16 conditions). Let $F_n$ denote the union of these $n$ events. Since, each event occurs with approximately the same probability,

$$\begin{aligned} P[F_n] &\approx n \cdot P[F] \\ &\approx n \cdot (\frac{1}{32})^5 \cdot (\frac{1}{64})^{11} \\ &= n \cdot \frac{1}{2^{91}}. \end{aligned}$$

From Table 1, we get the maximum number of ways that terms of a particular column can be pairwise equated and hence the upper bound on $n$ can be calculated to be $2 \cdot 2 \cdot 945 = 3780$, that is, $n < 3780$.

## 4.1 Formulating the Bias

Now, we establish a formula to compute $P[O_{1(i)} \oplus O_{3(i+7)} \oplus O_{7(i+7)} \oplus O_{8(i+7)} = 0]$, under the assumption of a perfectly random key/IV setup and the uniformity of bits when $F_n$ does not occur.Our experiments suggest that it is infeasible to find a set of conditions such that the overall bias (computed on the basis of the aforementioned assumption of randomness in the event that $F_n$ does not occur) is canceled or reduced in magnitude. Therefore, this assumption is reasonable. Let $T$ denote $O_{1(i)} \oplus O_{3(i+7)} \oplus O_{7(i+7)} \oplus O_{8(i+7)}$. Then using Bayes' rule we get

$$\begin{aligned} P[T=0] &= P[T=0|F_n \cap E_{17}] \cdot P[F_n \cap E_{17}] + P[T=0|F_n^c \cup E_{17}^c] \cdot P[F_n^c \cup E_{17}^c] \\ &= P[T=0|F_n \cap E_{17}] \cdot P[F_n \cap E_{17}] + P[T=0|F_n^c \cap E_{17}] \cdot P[F_n^c \cap E_{17}] \\ &+ P[T=0|F_n \cap E_{17}^c] \cdot P[F_n \cap E_{17}^c] + P[T=0|F_n^c \cap E_{17}^c] \cdot P[F_n^c \cap E_{17}^c] \\ &= 1 \cdot (n \cdot p \cdot \frac{1}{2^{91}}) + \frac{1}{2} \cdot (1 - n \cdot \frac{1}{2^{91}}) \cdot p + 0 \cdot P[F_n \cap E_{17}^c] \\ &+ \frac{1}{2} \cdot (1 - n \cdot \frac{1}{2^{91}}) \cdot (1-p) \\ &= \frac{1}{2} + n \cdot (2p-1) \cdot \frac{1}{2^{92}}. \end{aligned} \tag{22}$$

Hence, we see that the distribution of the outputs $(O_{1(i)}, O_{3(i+7)}, O_{7(i+7)}, O_{8(i+7)})$ is biased. The bias is equal to $n \cdot (2p-1) \cdot \frac{1}{2^{92}}$. In the following section, we provide formulas to compute $p$, i.e., the probability that $E_{17}$ occurs; or more generally, the probability that the 17th condition of each of the $n$ $F$-like events occurs, i.e., $P[d_{7(i-7)} \oplus d_{8(i-7)} \oplus c_{1(i)} \oplus d_{3(i)} \oplus d_{1(i+7)} \oplus c_{3(i+7)} \oplus c_{7(i+7)} \oplus e_{7(i+7)} \oplus c_{8(i+7)} \oplus e_{8(i+7)}] = 0$.

## 4.2 Biases in the Carry Terms

In this section, we provide formulas to calculate the bias in the carry terms. The carry terms $c$ and $e$ are generated in expressions of the form $(S \oplus X) + Z$. We now proceed to calculate $P[c_{l(i)} = 0]$ assuming that $S$, $X$ and $Z$ are uniformly distributed and independent. Under this assumption, $P[S_i = 0] = P[X_i = 0] = P[Z_i = 0] = \frac{1}{2}$, that is, the probability that the carry bit at position $i$ equals zero depends only on $i$. Stated otherwise, $P[c_{(i)} = 0] = P[e_{(i)} = 0]$. Let $P[c_{(i)} = 0]$ be denoted by $p_i$. Since there is no carry on the lsb, $p_0 = 1$. We now have Table 3.

From Table 3, using Bayes' rule we get

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

Solving this recursion, given $p_0 = 1$, we get

$$p_i = \frac{1}{2} + \frac{1}{2^{i+1}}. \tag{23}$$

**Table 3.** Truth table for computing $p_i$ (NR=Not Required)

| $c_{(i-1)}$ | $S_{(i-1)}$ | $X_{(i-1)}$ | $Z_{(i-1)}$ | $c_{(i)}$ | 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 | NR |
| 0 | 1 | 1 | 0 | 1 | NR |
| 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 | NR |
| 1 | 0 | 1 | 0 | 1 | NR |
| 1 | 0 | 1 | 1 | 0 | $\frac{1-p_{i-1}}{8}$ |
| 1 | 1 | 0 | 0 | 1 | NR |
| 1 | 1 | 0 | 1 | 1 | NR |
| 1 | 1 | 1 | 0 | 1 | NR |
| 1 | 1 | 1 | 1 | 1 | NR |

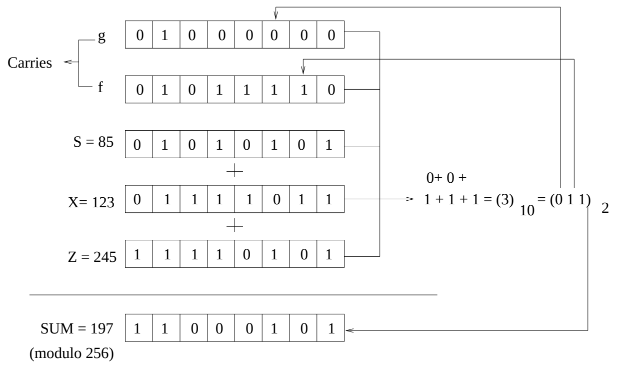

**Fig. 2.** An example showing how the carries are generated when three 8-bit variables $S = 85$, $X = 123$ and $Z = 245$ are added

Now, the carry terms $f$ and $g$ are generated in expressions of the form $S + X - Z$. This can be rewritten as $S + X + Z^c + 1$ since the additions in these two expressions are modulo $2^{32}$. The presence of two carries in $S + X + Z$ is demonstrated using the Figure 2. The carries generated in $S + X + Z^c + 1$ can be thought of as carries generated in $S + X + A$ where $A = Z^c$ and the carries on the lsb $f_{(0)} = 1$, $g_{(0)} = 0$. Let $q_i$ denote $P[f_{(i)} = 0]$ and $r_i$ denote $P[g_{(i)} = 0]$. Hence, $q_0 = 0$, $r_0 = 1$ and $r_1 = 1$. Now we have Table 4.

From Table 4, using Bayes' rule we get

$$q_i = \frac{1}{2} + \frac{5 \cdot q_{i-1} \cdot r_{i-1}}{8} - \frac{q_{i-1}}{4} - \frac{r_{i-1}}{4}, \tag{24}$$

$$r_{i+1} = \frac{1}{2} - \frac{q_{i-1} \cdot r_{i-1}}{4} + \frac{3 \cdot q_{i-1}}{8} + \frac{3 \cdot r_{i-1}}{8}. \tag{25}$$

Using the initial conditions, $q_0 = 0$, $r_0 = 1$ and $r_1 = 1$, $q_i$ and $r_i$ are computed recursively. Since $d_{m(i)}$ denotes $f_{m(i)} \oplus g_{m(i)}$ for any $m > 0$,

1. $P[d_{7(i-7)} = 0] = P[d_{8(i-7)} = 0] = q_{i-7 \bmod 32} \cdot r_{i-7 \bmod 32}$
   $+ (1 - q_{i-7 \bmod 32}) \cdot (1 - r_{i-7 \bmod 32})$,

**Table 4.** Truth table for computing $q_i$ and $r_{i+1}$ using $q_{i-1}$ and $r_{i-1}$ (NR=Not Required)

| $f_{(i-1)}$ | $g_{(i-1)}$ | $S_{(i-1)}$ | $X_{(i-1)}$ | $Z_{(i-1)}$ | $f_{(i)}$ | $g_{(i+1)}$ | Probability |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | $\frac{q_{i-1} \cdot r_{i-1}}{8}$ |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | $\frac{q_{i-1} \cdot r_{i-1}}{8}$ |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | $\frac{q_{i-1} \cdot r_{i-1}}{8}$ |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | NR |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | $\frac{q_{i-1} \cdot r_{i-1}}{8}$ |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | NR |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | NR |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | $\frac{q_{i-1} \cdot r_{i-1}}{8}$ |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | $\frac{q_{i-1} \cdot (1-r_{i-1})}{8}$ |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | NR |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | NR |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | NR |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | NR |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | NR |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | NR |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 | $\frac{q_{i-1} \cdot (1-r_{i-1})}{8}$ |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | $\frac{(1-q_{i-1}) \cdot r_{i-1}}{8}$ |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | NR |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | NR |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | NR |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | NR |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | NR |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | NR |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | $\frac{(1-q_{i-1}) \cdot r_{i-1}}{8}$ |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | NR |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | NR |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | NR |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | $\frac{(1-q_{i-1}) \cdot (1-r_{i-1})}{8}$ |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | NR |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | $\frac{(1-q_{i-1}) \cdot (1-r_{i-1})}{8}$ |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 | $\frac{(1-q_{i-1}) \cdot (1-r_{i-1})}{8}$ |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | $\frac{(1-q_{i-1}) \cdot (1-r_{i-1})}{8}$ |

2. $P[c_{1(i)} = 0] = \frac{1}{2} + \frac{1}{2^{i+1}}$,
3. $P[d_{3(i)} = 0] = q_i \cdot r_i + (1 - q_i) \cdot (1 - r_i)$,
4. $P[d_{1(i+7)} = 0] = q_{i+7 \bmod 32} \cdot r_{i+7 \bmod 32}$
   $+ (1 - q_{i+7 \bmod 32}) \cdot (1 - r_{i+7 \bmod 32})$,
5. $P[c_{3(i+7)} = 0] = P[c_{7(i+7)} = 0] = P[e_{7(i+7)} = 0] = P[c_{8(i+7)} = 0]$
   $= P[e_{8(i+7)} = 0] = \frac{1}{2} + \frac{1}{2^{(i+7 \bmod 32)+1}}$.

Using the above formulas, the value of $p$ can be computed for any given $i$. Running simulation, we find that the maximum bias in the chosen outputs occurs when $i = 25$ which corresponds to $p = 0.5 - 2^{-34.2}$. Hence, (22) gives us

$$P[T = 0] = \frac{1}{2} - \frac{n}{2^{125.2}}$$
$$\Rightarrow P[T = 1] = \frac{1}{2} + \frac{n}{2^{125.2}},$$

when $i = 25$. Substituting $n = 3780$ in the above equation, we get:

$$P[T = 1] = \frac{1}{2} + \frac{1}{2^{113.3}}. \tag{26}$$

This is an upper bound on the probability that the outputs $(O_{1(i)}, O_{3(i+7)}, O_{7(i+7)}, O_{8(i+7)})$ of TPy6 (and hence Py6) are biased. From Sect. 3, we found that $n \geq 1$. From the previous discussion, we see that $n < 3780$. Hence, $1 \leq n < 3780$. If $n = 1$, then $P[T = 1] = \frac{1}{2} + \frac{1}{2^{125.2}}$. Thus,

$$\frac{1}{2}(1 + \frac{1}{2^{124.2}}) \leq P[T = 1] < \frac{1}{2}(1 + \frac{1}{2^{112.3}}). \tag{27}$$

## 5 The Distinguisher

A distinguisher is an algorithm which distinguishes a given stream of bits from a stream of bits generated by a perfect PRBG. The distinguisher is constructed by collecting sufficiently many outputs $(O_{1(25)}, O_{3(0)}, O_{7(0)}, O_{8(0)})$ generated by as many key/IVs. To compute the minimum number of samples required to establish the distinguisher, we use the following corollary of a theorem from [7].

**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.*

In the present case, $e$ is the event $O_{1(25)} \oplus O_{3(0)} \oplus O_{7(0)} \oplus O_{8(0)} = 0$, $X$ is the distribution of the outputs $O_1$, $O_3$, $O_7$ and $O_8$ produced by a perfectly random keystream generator and $Y$ is the distribution of the outputs produced by TPy6. From (27), $p = \frac{1}{2}$ and the highest value of $q = \frac{1}{2^{112.3}}$. Hence $O(\frac{1}{(2^{-112.3})^2}) = O(2^{224.6})$ output samples are needed to construct the best distinguisher with a non-negligible probability of success.

## 6 A Family of Distinguishers

In Sect. 3 we found that the outputs at rounds 1, 3, 7 and 8 are biased allowing us to build a distinguisher. It is found that there exist plenty of 4-tuples of biased outputs. The generalization is presented in the following theorem.

**Theorem 2.** *The distribution of the outputs $(O_{r(i)}, O_{r+2(i+7)}, O_{t(i+7)}, O_{u(i+7)})$ of the TPy6 are biased for many suitably chosen $(r, t, u)$'s where $r > 0$; $t, u \geq 5$; $t \notin \{r, r+2, u\}$; $u \notin \{r, r+2, t\}$.*

We omit the proof as it is similar to the proof furnished for Theorem 1. This allows us to construct a family of distinguishers for the cipher TPy6. It seems possible to combine these huge number of distinguishers in order to construct one single efficient distinguisher; however, any concrete mathematical model to combine them is still an interesting open problem. Another major implication of the above generalization theorem is the fact that the TPy6 outputs will remain always biased no matter how many initial outputwords are discarded from the keystream.

## 7 Two New Ciphers: TPy6-A and TPy6-B

The Py-family of stream ciphers has been subject to extensive analysis ever since Py and Py6 were proposed in April 2005. The impressive speeds of the ciphers in software, particularly the Py6 and the TPy6, have motivated us to modify the TPy6 to rule out all the attacks described in the previous sections of the paper. Firstly, many attacks on the Py, in particular, [5], [8], [9] and [15] can be easily translated to attacks on the Py6. However, due to smaller internal state of the TPy6, the attack described in [11] does not apply to TPy6. Secondly, the speed of execution of Py6 on Pentium-III is about 2.82 cycles/byte which is very fast. These observations make TPy6 and Py6 more favorable to be used as fast stream ciphers than Py. The TPy6 is resistant to the attacks described in [8], [15]. In order to generate a fast and secure stream cipher, we redesign the TPy6 where the variable rotation of a 32-bit term $s$ in the round function is replaced by a constant, non-zero rotation term. The resultant cipher is named TPy6-A. It is shown that this tweak clearly reduces one addition operation in each round (thereby, the performance is improved) and makes the cipher secure against all the existing attacks on Py6 and TPy6. A relatively slower version, where one outputword is removed from each round of TPy6, is also proposed. The speeds of execution of the TPy6-A and TPy6-B on Pentium-III are 2.65 cycles/byte and 4.4 cycles/byte. Our security analysis conjectures that the TPy6-A and TPy6-B are immune to all attacks better than brute force. Algorithm 2 describes the PRBGs of the TPy6-A and TPy6-B. Note that the key/IV setup algorithms of the ciphers are identical to the key/IV setup of TPy6.

**Algorithm 2** Round functions of TPy6-A and TPy6-B

```
Require: $Y[-3, ..., 64]$, $P[0, ..., 63]$, a 32-bit variable $s$
Ensure: 64-bit random (TPy6-A) or 32-bit random (TPy6-B) output
    /*Update and rotate $P$*/
 1: swap ($P[0]$, $P[Y[43]\&63]$);
 2: rotate ($P$);
    /* Update s*/
 3: $s+ = Y[P[18]] - Y[P[57]]$;
 4: $s = ROTL32(s, 19)$; /*Tweak: variable rotation in TPy6 replaced by a constant non-zero rotation*/
    /* Output 8 bytes (least significant byte first)*/
 5: output $((ROTL32(s, 25) \oplus Y[64]) + Y[P[8]])$;/*this step is removed for TPy6-B*/
 6: output $((\quad s \quad \oplus Y[-1]) + Y[P[21]])$;
    /* Update and rotate $Y$*/
 7: $Y[-3] = (ROTL32(s, 14) \oplus Y[-3]) + Y[P[48]]$;
 8: rotate($Y$);
```

### 7.1 Security Analysis

In this section we justify how the new ciphers TPy6-A and TPy6-B should be able to resist several common attacks against array-based stream ciphers.

*(i) Resistance to Distinguishing Attacks:* The TPy6-A and the TPy6-B are modified versions of the TPy6. The following distinguishing attacks are applicable to the TPy6. We now show why those attacks do not apply to the new designs.

1. *Paul-Preneel attack[10]:* This attack applies to the TPy6. Condition 1 under Theorem 1 in [9], that is, $P_2[26] \equiv -18 \bmod 32$, is impossible when $c = 19$ (in which case we have $P_2[26] \equiv 1 \bmod 32$). Note that when $c = 0$, $P_2[26] \equiv -18 \bmod 32$ is satisfied. Therefore, $c = 0$ is not a safe choice.
2. *The attack described in this paper:* Again, condition 1 under Theorem 1, that is, $P_1[26] \equiv -18 \bmod 32$, is violated when $c = 19$ (in which case we have $P_1[26] \equiv 1 \bmod 32$). Note that

condition 1 is common to all the $n$ sets ($1 \leq n < 3780$) of conditions (see Sect. 4) and hence its violation nullifies the attack.

In Appendix A, we elaborate more on the usefulness of selection of *constant rotation* to eliminate any distinguishing attacks on the new ciphers.
*(ii) Resistance to Differential attacks*: Wu and Preneel found weaknesses in the IV setups of the Py6 [15]. Exploiting these weaknesses, it is possible to recover some key-dependent information. The cipher TPy6 was specifically designed to rule out these weaknesses. Since the IV setup algorithms of the TPy6-A and the TPy6-B are identical with that of TPy6, these attacks are no longer applicable to the new ciphers.

*(iii) Resistance to Algebraic attacks and Guess-and-Determine Attacks:* TPy6-A and TPy6-B are array-based stream ciphers. The sizes of the internal states of TPy6-A and TPy6-B are 2,720 bits each, which is quite large. Hence, it appears infeasible to mount algebraic attacks that are otherwise common against LFSR-based stream ciphers which have low footprints. From our experiments, we expect that the TPy6-A and TPy6-B are also secure against guess-and-determine attacks.

## 8 Conclusions and Open Problems

The first contribution of the paper is the development of a family of distinguishers from the outputs at rounds $r$, $r+2$, $t$ and $u$ of the cipher TPy6 (and Py6), where $r > 0$; $t, u \geq 5$; $t \notin \{r, r+2, u\}$; $u \notin \{r, r+2, t\}$. The best distinguisher works with data complexity $2^{224.6}$. It is reasonable to assume that these weak states can be combined to mount a more efficient attack on TPy; however, methods to combine many distinguishers into a single yet more efficient one is still an open problem. The second contribution is a proposal of two new, extremely fast stream ciphers TPy6-A and TPy6-B, which rule out all the existing attacks on the TPy6 and are conjectured to be immune to all attacks better than brute force.

## References

1. T. Baignères, P. Junod and S. Vaudenay, “How Far Can We Go Beyond Linear Cryptanalysis?,” *Asiacrypt 2004* (P. Lee, ed.), vol. 3329 of *LNCS*, pp. 432–450, Springer-Verlag, 2004.
2. E. Biham, J. Seberry, “Tweaking the IV Setup of the Py Family of Ciphers – The Ciphers Tpy, TPypy, and TPy6,” Published on the author’s webpage at `http://www.cs.technion.ac.il/ biham/`, January 25, 2007.
3. E. Biham, J. Seberry, “Py (Roo): A Fast and Secure Stream Cipher using Rolling Arrays,” *ecrypt submission*, 2005.
4. E. Biham, J. Seberry, “Pypy (Roopy): Another Version of Py,” *ecrypt submission*, 2006.
5. P. Crowley, “Improved Cryptanalysis of Py,” *Workshop Record of SASC 2006 - Stream Ciphers Revisited*, ECRYPT Network of Excellence in Cryptology, February 2006, Leuven (Belgium), pp. 52-60.
6. S. Fluhrer, I. Mantin, A. Shamir, “Weaknesses in the Key Scheduling Algorithm of RC4,” Selected Areas in Cryptography 2001 (S. Vaudenay, A. Youssef, eds.), vol 2259 of LNCS, pp. 1-24, Springer-Verlag, 2001.
7. 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.
8. T. Isobe, T. Ohigashi, H. Kuwakado M. Morii, “How to Break Py and Pypy by a Chosen-IV Attack,” eSTREAM, ECRYPT Stream Cipher Project, Report 2006/060.
9. S. Paul, B. Preneel, G. Sekar, “Distinguishing Attacks on the Stream Cipher Py,” *Fast Software Encryption 2006* (M. Robshaw, ed.), vol. 4047 of *LNCS*, pp. 405-421, Springer-Verlag, 2006.
10. S. Paul, B. Preneel “On the (In)security of Stream Ciphers Based on Arrays and Modular Addition,” *Asiacrypt 2006* (X. Lai and K. Chen, eds.), vol. 4284 of *LNCS*, pp. 69-83, Springer-Verlag, 2006.
11. G. Sekar, S. Paul, B. Preneel, “Weaknesses in the Pseudorandom Bit Generation Algorithms of the Stream Ciphers TPypy and TPy,” available at `http://eprint.iacr.org/2007/075.pdf`.
12. G. Sekar, S. Paul, B. Preneel, “New Weaknesses in the Keystream Generation Algorithms of the Stream Ciphers TPy and Py,” available at `http://eprint.iacr.org/2007/230.pdf`.

13. G. Sekar, S. Paul, B. Preneel, "Related-key Attacks on the Py-family of Ciphers and an Approach to Repair the Weaknesses," available at `http://www.cosic.esat.kuleuven.be/publications/article-932.pdf`.
14. Y. Tsunoo, T. Saito, T. Kawabata, H. Nakashima, "Distinguishing Attack against TPypy," *Selected Areas in Cryptography 2007* (to appear).
15. H. Wu, B. Preneel, "Differential Cryptanalysis of the Stream Ciphers Py, Py6 and Pypy," *Eurocrypt 2007* (Moni Naor, ed.), vol. 4515 of *LNCS*, pp. 276-290, Springer-Verlag, 2007.

## A Effect of Any Non-zero Constant Rotation in TPy6-A and TPy6-B

The distinguishing attacks on Py6 (and hence TPy6) presented in [10] are based on the fact that, when certain conditions on the elements of array $P$ are satisfied then $s_{r(i)} = s_{r+2(j)}$, where $r$ denotes the round and $i$, $j$ $(0 \leq i, j \leq 31)$ denote the bit positions.

We now examine the effect of constant rotation (say $c$) in step 4 of the PRBG of TPy6 (see Algorithm 1).

$$\begin{aligned} s_{r(i)} &= ROTL32(s_{r-1} + Y_r[P_r[18]] - Y_r[P_r[57]], c)_i && (28)\\ &= (s_{r-1} + Y_r[P_r[18]] - Y_r[P_r[57]])_{i-c \bmod 32}. && (29)\end{aligned}$$

Let $k$ denote $i - c \bmod 32$. Therefore,

$$s_{r(i)} = s_{r-1(k)} \oplus Y_r[P_r[18]]_k \oplus Y_r^c[P_r[57]]_k \oplus e_{r(k)},$$

where $e$ denotes the carry term generated in (29) and $e_{r(0)} = 1$.

Similarly, if $l$ denotes $j - c \bmod 32$, we have,

$$s_{r+2(j)} = s_{r+1(l)} \oplus Y_{r+2}[P_{r+2}[18]]_l \oplus Y_{r+2}^c[P_{r+2}[57]]_l \oplus e_{r+2(l)}. \quad (30)$$

Again, we have

$$s_{r+1(l)} = s_{r(m)} \oplus Y_{r+1}[P_{r+1}[18]]_m \oplus Y_{r+1}^c[P_{r+1}[57]]_m \oplus e_{r+1(m)}, \quad (31)$$

where $m$ denotes $l - c \bmod 32$, and

$$s_{r(m)} = s_{r-1(n)} \oplus Y_r[P_r[18]]_n \oplus Y_r^c[P_r[57]]_n \oplus e_{r(n)}, \quad (32)$$

where $n$ denotes $m - c \bmod 32$. Substituting (31) and (32) in (30), we get that the expression for $s_{r(i)} \oplus s_{r+2(j)}$ contains the term $s_{r-1(k)} \oplus s_{r-1(n)}$. Now, it follows that if $k \neq n$, it is very likely that the terms $s_{r(i)}$ and $s_{r+2(j)}$ are not correlated. Besides, there are a number of $Y$-terms at different bit-positions and the terms do not cancel out if $i \neq j$.

Now, $n = j - 3c \bmod 32$ and $k = i - c \bmod 32$. Hence, when $i = j$, we have $c \neq 0$ in order that $k \neq n$ be satisfied. Thus, with $c = 19$, we expect that there will be no correlations in the output stream such that a distinguisher can be built with data complexity less than that of exhaustive search. The constant 19 is not influenced by any factors and may be replaced by any non-zero constant.