Tweaking the IV Setup of the Py Family of Stream Ciphers — The Ciphers TPy, TPypy, and TPy6

Eli Biham* Jennifer Seberry†

January 25, 2007

Abstract

Py, Pypy, and Py6 are secure stream ciphers, which were designed especially for the eSTREAM project. Recently, equivalent IV's were found for all members of this family, where, once found, allow attackers to recover some key dependent data.

In this note we tweak the mistake in the design of the IV setup of Py, Pypy, and Py6, so that equivalent IVs will not exists. The names of the Tweaked variants are prepended with T to denote that they are the tweaked variants – TPy is the tweaked Py, TPypy is the tweaked Pypy, and TPy6 is Tweaked Py6 (pronounced TRoo, TRoopy, and TRoo6).

Key setup and encryption speeds of the tweaked variants are unchanged compared to the originals, as no change is made in the key setup and stream generation. The tweaked IV setup may be very slightly slower than the original, as the modification is very small.

As a result, the tweaked versions, TPy and TPy6 are the fastest eSTREAM candidates, and the TPypy supplies an even higher level of security with speed similar to the second fastest unbroken eSTREAM candidate, but with a much faster key and IV setups.

The stream ciphers Py and Py6 were proposed in [1] as very fast and secure submissions to the eSTREAM project (which is part of Ecrypt [4]). Pypy was added later to the Py family in [2].

In [5] it was claimed that a distinguishing attack on Py was found. The requirements and complexity of this attack, however, were not within the security claims of Py. Note that in a follow up paper [3] the claims were correctly addressed. Pypy does not have the features that this attack is based on, and thus similar attacks cannot be applied to Pypy.

Recently, an attack on the IV setup of the Py family was found [6], that allows to find equivalent IVs very quickly, i.e., to find pairs of IVs which lead to the same internal state. Using such pairs of IVs, it was found possible to recover some key-dependent information.

*Computer Science department, Technion, Haifa 32000, Israel. biham@cs.technion.ac.il, http://www.cs.technion.ac.il/~biham/.

†Centre for Computer Security Research, SITACS, University of Wollongong, 2522, NSW, Australia. j.seberry@uow.edu.au, http://www.uow.edu.au/~jennie/.

Cipher	Pypy and TPypy	Py and TPy	RC4
Pentium III	4.58	2.80	7.30
Core Duo	3.93	2.38	7.15

Table 1. Speed of Pypy, TPypy, Py, TPy and RC4 on Intel Pentium III and Core Duo (cycles/byte).

The authors of Py [1] claimed that there are no equivalent IVs in Py. When looking at the IV setup, and considering the two loops that mix the IV into EIV (on which the new attack is based), it becomes clear that the authors of Py, when tried to improved the IV setup, added the second loop in order to mix the IV better into the internal state. However, this extra mixing was a poor choice, as if only the first loop was proposed, the IV setup would not have the equivalent IVs problem.

In this paper, we tweak the IV setup of all the three members of the Py family to solve that mistake. In the tweak of Py and Pypy, we modify the second mixing loop of the IV, such that it will not lead to equivalent IVs, by ensuring that it is invertible (following the suggestion of [6]), in a way that we prefer over removing this second loop. We also change the final loop of the IV setup slightly, also in order to ensure invertibility, though we belive that the problem in the IV setup would be solved even without it. Note that in TRy6, unlike in Py6, the EIV rolling array is also doubled in size to ivsizeb*2. As a result, we limit the keys of TPy6 to be up to 64 bytes, and the IVs to be up to 32 bytes (both are larger than the minimal and recommended sizes).

The Speed of RC4, Py, Pypy, TPy, and TPypy (i.e., both the original and the tweaked versions) on several processors is given in Table 1 (the speed is measured by the NESSIE test suite under the NESSIE API for synchronous stream ciphers).

The new IV setup of TPy and TPypy is outlined in Figure 1. For comparison, the Original IV setup is outlined in Figure 2. The new IV setup of TPy6 is outlined in Figure 3.

References

[1] Eli Biham, Jennifer Seberry, **Py** *(Roo, רו): A Fast and Secure Stream Cipher using Rolling Arrays*, submitted to the eSTREAM project, 2005.

[2] Eli Biham, Jennifer Seberry, **Pypy** *(Roopy, רופי): Another Version of Py*, submitted to the eSTREAM project, 2006.

[3] Paul Crowley, *Improved cryptanalysis of Py*, presented in SASC 2006, 2006.

[4] Ecrypt, `http://www.ecrypt.eu.org/`.

[5] Souradyuti Paul, Bart Preneel and Gautham Sekar, *Distinguishing Attacks on the Stream Cipher Py*, Fast Software Encryption, FSE 2006.

[6] Hongjun Wu, Bart Preneel, *Attacking the IV Setup of Py and Pypy*, private communications, 2006.

```
/* Create an initial permutation */
u8 v= iv[0] ^ ((Y(0)>>16)&0xFF);
u8 d=(iv[1 mod ivsizeb] ^ ((Y(1)>>16)&0xFF))|1;
for(i=0; i<256; i++)
  {
    P(i)=internal_permutation[v];
    v+=d;
  }
/* Now P is a permutation */

/* Initial s */
s = ((u32)v<<24)^((u32)d<<16)^((u32)P(254)<<8)^((u32)P(255));
s ^= Y(YMININD)+Y(YMAXIND);
/* Mix iv[0..ivsizeb-1] into EIV[0..ivsizeb-1] and s */
for(i=0; i<ivsizeb; i++)
  {
    s = s + iv[i] + Y(YMININD+i);
    u8 s0 = P(s&0xFF);
    EIV(i) = s0;
    s = ROTL32(s, 8) ^ (u32)s0;
  }
/* Again, but with the last words of Y, and using EIV instead of iv */
for(i=0; i<ivsizeb; i++)
  {
    /* Tweak: Changed assignment to s to ensure invertibility */
    s = s + EIV((i+ivsizeb-1) mod ivsizeb) + Y(YMAXIND-i);
    u8 s0 = P(s&0xFF);
    EIV(i) += s0;
    s = ROTL32(s, 8) ^ (u32)s0;
  }
/* Updating the Rolling Arrays and $s$ */
for(i=0; i<260; i++)
  {
    u32 x0 = EIV(0) = EIV(0)^(s&0xFF);
    rotate(EIV);
    swap(P(0),P(x0));
    rotate(P);
    /* Tweak: Changed assignment to Y(YMININD) and s to ensure invertibility*/
    s = ROTL32(s,8)+Y(YMAXIND);
    Y(YMININD)+=s^Y(x0);
    rotate(Y);
  }

s=s+Y(26)+Y(153)+Y(208);
if(s==0)
  s=(keysizeb*8)+((ivsizeb*8)<<16)+0x87654321;
```

Figure 1. The (Tweaked) IV Setup of TPy and TPypy.

```
/* Create an initial permutation */
u8 v= iv[0] ^ ((Y(0)>>16)&0xFF);
u8 d=(iv[1 mod ivsizeb] ^ ((Y(1)>>16)&0xFF))|1;
for(i=0; i<256; i++)
  {
    P(i)=internal_permutation[v];
    v+=d;
  }
/* Now P is a permutation */

/* Initial s */
s = ((u32)v<<24)^((u32)d<<16)^((u32)P(254)<<8)^((u32)P(255));
s ^= Y(YMININD)+Y(YMAXIND);
for(i=0; i<ivsizeb; i++)
  {
    s = s + iv[i] + Y(YMININD+i);
    u8 s0 = P(s&0xFF);
    EIV(i) = s0;
    s = ROTL32(s, 8) ^ (u32)s0;
  }
/* Again, but with the last words of Y, and update EIV */
for(i=0; i<ivsizeb; i++)
  {
    s = s + iv[i] + Y(YMAXIND-i);
    u8 s0 = P(s&0xFF);
    EIV(i) += s0;
    s = ROTL32(s, 8) ^ (u32)s0;
  }
/* Updating the Rolling Arrays and $s$ */
for(i=0; i<260; i++)
  {
    u32 x0 = EIV(0) = EIV(0)^(s&0xFF);
    rotate(EIV);
    swap(P(0),P(x0));
    rotate(P);
    Y(YMININD)=s=(s^Y(YMININD))+Y(x0);
    rotate(Y);
  }

s=s+Y(26)+Y(153)+Y(208);
if(s==0)
  s=(keysizeb*8)+((ivsizeb*8)<<16)+0x87654321;
```

Figure 2. The (Original) IV Setup of Py and Pypy.

```
u8 v= iv[0] ^ ((Y(0)>>16)&0xFF);
u8 d=(iv[1%ivsizeb] ^ ((Y(1)>>16)&0xFF))|1;
for(i=0; i<64; i++)
  {
    P(i)=v&0x3F;
    v+=d;
  }
v&=0x3F; /* Back to the original v */
/* Initial s */
s = ((u32)v<<24)^((u32)d<<16)^((u32)P(62)<<8)^((u32)P(63));
s ^= Y(YMININD)+Y(YMAXIND);
/* Mix iv[0..ivsizeb-1] into EIV[0..2*ivsizeb-1] and s */
for(i=0; i<ivsizeb; i++)
  {
    s = s + iv[i] + Y(YMININD+i);
    u8 s0 = P(s&0x3F);
    EIV(i) = s0;
    /* Tweak: */
    u8 s1 = P((s>>2)&0x3F);
    EIV(i+ivsizeb) = s1;
    s = ROTL32(s, 12) ^ (u32)s0 ^ ((u32)s1)<<6;
  }
/* Again, but with the last words of Y, and update EIV */
for(i=0; i<ivsizeb*2; i++)
  {
    /* Tweak: Changed assignment to s to ensure invertibility */
    s = s + EIV((i+ivsizeb*2-1) mod (ivsizeb*2)) + Y(YMAXIND-i);
    u8 s0 = P(s&0x3F);
    EIV(i) += s0;
    EIV(i) &= 0x3F; /* EIV should be 6 bit-long in TPy6, used as an index to P and Y */
    s = ROTL32(s, 6) ^ (u32)s0;
  }
/* Updating the Rolling Arrays and $s$ */
for(i=0; i<PYSIZE*8; i+=8)
  {
    u32 x0 = EIV(0) = EIV(0)^(s&0x3F);
    rotate(EIV);
    swap(P(0),P(x0));
    rotate(P);
    /* Tweak: */
    s = ROTL32(s,8)+Y(YMAXIND);
    Y(YMININD)+=s^Y(x0);
    rotate(Y);
  }
s=s+Y(i,8)+Y(i,21)+Y(i,48);
if(s==0)
  s=keysize+(ivsize<<16)+0x87654321;
```

Figure 3. The (Tweaked) IV Setup of TPy6.