

Attacking the IV Setup of Py and Pypy

Last modified August 29, 2006

Hongjun Wu and Bart Preneel

Katholieke Universiteit Leuven, ESAT/SCD-COSIC
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
{wu.hongjun,bart.preneel}@esat.kuleuven.be

Abstract. In this paper we show that Py and Pypy are practically insecure due to the flaw in their IV setup. With about 2^{16} IVs (with special difference between the IVs), there are two identical keystreams.

1 Introduction

Py [1] and Pypy [2] are stream ciphers submitted to the ECRYPT eSTREAM project. A distinguishing attack on Py was found by Paul, Preneel and Sekar [4]. In that attack, the keystream can be distinguished from random with $2^{89.2}$ outputs. Later, the attack was improved by Crowley [3], and the data required in the attack is reduced to 2^{72} .

In order to resist the distinguishing attack on Py, the designers decided to discard half of the outputs, i.e., the first output of the two outputs at each step is discarded. The new version is called Pypy.

The IV setup of Py and Pypy are identical. In this paper, we show that there is serious flaw in the IV setup of Py and Pypy. For IVs with special difference, two keystreams can be identical for every 2^{16} IVs.

This paper is organized as follows. In Sect. 2, we illustrate the IV setup of Py and Pypy. Section 3 gives the attack on the IV setup. Section 4 concludes this paper.

2 The IV Setup of Py and Pypy

The initializations of Py and Pypy are identical. The initialization consists of two stages: key setup and IV setup. In key setup, the key is expanded into the internal state of the cipher. In the IV setup, the IV is introduced into the state. In the following, we will ignore the description of the key setup and only introduce the IV setup. (The key setup is described in [1].)

The following description of the IV setup is from [1]. P is an array with 256 8-bit elements. Y is an array with 260 32-bit elements, s is 32-bit. $YMININD = -3$, $YMAXIND = 256$. EIV is a byte array with the same size as the IV. The table 'internal_permutation' is a constant permutation table with 256 elements. 'ivsize' is the size of the IV in bytes. After the key setup, the array Y is affected by the key value. Then the following IV setup is carried out.

```

/* 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(YMAXINDi);
    u8 s0 = P(s&0xFF);
    EIV(i) += s0;
    s = ROTL32(s, 8) ^ (u32)s0;
}

/*updating the rolling array 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;

```

3 Attack the IV Setup

At the beginning of the IV setup, only 15 bits of the IV (IV[0] and IV[1]) are applied to initialize the array P and s (the least significant bit of IV[1] is not used). For an IV pair, if those 15 bits are identical, then the resulting P are the same. Then we notice that the IV is applied to update the values of s and EIV as follows.

```
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;
}
for(i=0; i<ivsizeb; i++)
{
    s = s + iv[i] + Y(YMAXINDi);
    u8 s0 = P(s&0xFF);
    EIV(i) += s0;
    s = ROTL32(s, 8) ^ (u32)s0;
}
```

In the following, we give two types of IV pairs that result in identical keystreams.

3.1 IVs different at two bytes

We illustrate the attack with an example. Suppose that two IVs, iv_1 and iv_2 , are different at only two bytes $iv_1[i] \oplus iv_2[i] = 1$, the least significant bit of $iv_1[i]$ is 1, and $iv_1[i+1] \neq iv_2[i+1]$. And suppose that $i \geq 1$. Let us trace how the difference in IV affects the s and EIV .

```
s = s + iv[i] + Y(YMININD+i);
u8 s0 = P(s&0xFF);
EIV(i) = s0;
s = ROTL32(s, 8) ^ (u32)s0;
```

At the end of this step, $EIV_1[i] \neq EIV_2[i]$. Let $\beta_1 = EIV_1[i]$, and $\beta_2 = EIV_2[i]$. At the end of this step, $s_1 - s_2 = 0x100 + \delta_1$, where $\delta_1 = (\beta_1 \oplus x) - (\beta_2 \oplus x)$ where x is a 32-bit number (the $ROTL32(s,8)$). Then we look at the next step.

```
s = s + iv[i+1] + Y(YMININD+i+1);
u8 s0 = P(s&0xFF);
EIV(i) = s0;
s = ROTL32(s, 8) ^ (u32)s0;
```

Because $iv_1[i+1] \neq iv_2[i+1]$, if $iv_2[i+1] - iv_1[i+1] = \delta_1$, then s_1 and s_2 become identical with high chance. Let $s_1 = s_2$ with probability p_1 . Running the simulation, $p_1 = 2^{-10.6}$. If $s_1 = s_2$, then $EIV_1[i+1] = EIV_2[i+1]$, and in the following steps $i+2, i+3, \dots, i+ivsize-1$, the s_1 and s_2 remain the same, and $EIV_1[i+2] = EIV_2[i+2]$, $EIV_1[i+3] = EIV_2[i+3]$, \dots , $EIV_1[i+ivsize-1] = EIV_2[i+ivsize-1]$.

Then $IV[i]$ and $IV[i+1]$ are used again to update s and EIV .

```

s = s + iv[i] + Y(YMAXIND-i);
u8 s0 = P(s&0xFF);
EIV(i) += s0;
s = ROTL32(s, 8) ^ (u32)s0;

```

At the end of this step, $EIV_1[i] = EIV_2[i]$ with probability $\frac{1}{255}$. Let $\gamma_1 = s0_1$, and $\gamma_2 = s0_2$. For $EIV_1[i] = EIV_2[i]$, we require that $\gamma_2 - \gamma_1 = \beta_1 - \beta_2$. At the end of this step, $s_1 - s_2 = 0x100 + \delta_2$, where $\delta_2 = (\gamma_1 \oplus y) - (\gamma_2 \oplus y)$ where y is a 32-bit number (the $ROTL32(s,8)$). Note that δ_1 and δ_2 are correlated since $\gamma_2 - \gamma_1 = \beta_1 - \beta_2$. Then we look at the next step.

```

s = s + iv[i+1] + Y(YMAXIND-i-1);
u8 s0 = P(s&0xFF);
EIV(i) += s0;
s = ROTL32(s, 8) ^ (u32)s0;

```

At the end of this step, if $iv_2[i+1] - iv_1[i+1] = \delta_2$, then s_1 and s_2 become identical with high chance. Note that $iv_2[i+1] - iv_1[i+1] = \delta_1$, and δ_1 and δ_2 are correlated, so $iv_2[i+1] - iv_1[i+1] = \delta_2$ with with probability larger than 2^{-8} . Let $s_1 = s_2$ with probability p'_1 . Running the simulation, $p'_1 = 2^{-5.6}$. Once the two s are identical, $EIV_1[i+1] = EIV_2[i+1]$, and in the following steps $i+2, i+3, \dots, i+ivsize-1$, s_1 and s_2 remain the same, and $EIV_1[i+2] = EIV_2[i+2]$, $EIV_1[i+3] = EIV_2[i+3]$, \dots , $EIV_1[i+ivsize-1] = EIV_2[i+ivsize-1]$.

Thus after introducing the IV to update s and EIV , $s_1 = s_2$ and $EIV[1] = EIV[2]$ with probability $p_1 \times \frac{1}{255} \times p'_1 \approx 2^{-24.2}$.

Experiment 1. We use 2^{14} random 128-bit keys in the attack. For each key, we randomly generate 2^{16} pairs of 128-bit IV with the difference at only two bytes: $iv_1[6] \oplus iv_2[6] = 1$, $iv_1[7] \neq iv_2[7]$. Then we found that 111 pairs of those 2^{30} keystream pairs are identical. For example, for the key (08 da f2 35 a3 d5 94 e2 85 cc 68 ba 7e 10 8a b4), and the IV pair (6e e7 09 b1 35 85 2f 07 1a fe 3f 50 a8 84 30 11) and (6e e7 09 b1 35 85 2e 80 1a fe 3f 50 a8 84 30 11), those two keystreams are identical, and the first 16 keystream bytes of Pypy are (6f eb ca 18 54 3f 59 96 b6 17 8a 54 6e bd 45 1f).

From the experiment, we found that for an IV pair with the required difference, the two keystreams are identical with probability about $\frac{111}{2^{30}} = 2^{-23.2}$, about twice of the theoretical value.

The IV difference at two bytes. In the above analysis, the difference is chosen as $iv_1[i] \oplus iv_2[i] = 1$, $iv_1[i+1] \neq iv_2[i+1]$ ($i \geq 1$). For this type of IV difference, we can generalize it so that $iv_1[i]$ and $iv_2[i]$ can choose other differences instead of 1. **As long as $(iv_1[i] - iv_2[i]) \bmod 256 = 1$ or 255 , $iv_1[i+1] \neq iv_2[i+1]$ ($i \geq 2$), then there is chance that the two keystreams can be identical.**

For example, if $iv_1[i] \oplus iv_2[i] = 3$, the two least significant bits of $iv_1[i]$ are 01 or 10, and $iv_1[i+1] \neq iv_2[i+1]$ ($i \geq 2$), then two identical keystreams appear with probability $2^{-23.2}$. In average, if $iv_1[i] - iv_2[i] = 1$, and $iv_1[i+1] \neq iv_2[i+1]$ ($i \geq 2$), then two identical keystreams appear with probability $2^{-26.4}$.

3.2 IVs different at 3 bytes

In the above attack, we deal with the $IV[i]$ and $IV[i+1]$, and use the difference at $IV[i+1]$ to eliminate the difference introduced by $IV[i]$ in s . In the following, we introduce another type of difference to deal with the situation that the difference at $IV[i+1]$ cannot eliminate the difference introduced by $IV[i]$ in s . The solution is to introduce the difference at $IV[i+4]$.

We illustrate the attack with an example. Suppose that two IVs, iv_1 and iv_2 , are different at only three bytes $iv_1[i] \oplus iv_2[i] = 0x80$, the most significant bit of $iv_1[i]$ is 1, $iv_2 \neq iv_2[i+1]$, $iv_2[i+4] \oplus iv_1[i+4] = 0x80$, and the most significant bit of $iv_1[i+4]$ is 0. And suppose that $i \geq 2$. Let us trace how the difference affects the s and EIV.

```

s = s + iv[i] + Y(YMININD+i);
u8 s0 = P(s&0xFF);
EIV(i) = s0;
s = ROTL32(s, 8) ^ (u32)s0;

```

At the end of this step, $EIV_1[i] \neq EIV_2[i]$, and $s_1 - s_2 = 0x8000 + \delta_1$, where δ_1 is the difference of two different 8-bit numbers. Then we look at the next step.

```

s = s + iv[i+1] + Y(YMININD+i+1);
u8 s0 = P(s&0xFF);
EIV(i) = s0;
s = ROTL32(s, 8) ^ (u32)s0;

```

Because $iv_1[i+1] \neq iv_2[i+1]$, $s_1 - s_2 = 0x8000$ with chance $p_2 = 2^{-8}$. If $s_1 - s_2 = 0x8000$, then $EIV_1[i+1] \oplus EIV_2[i+1] = 0$.

Since $v_1[i+2] = v_2[i+2]$, at the end of the step of introducing $IV[i+2]$, $EIV_1[i+2] = EIV_2[i+2]$, and $s_1 - s_2 = 0x800000$ with probability about 1.

Since $v_1[i+3] = v_2[i+3]$, at the end of the step of introducing $IV[i+3]$, $EIV_1[i+3] = EIV_2[i+3]$, and $s_1 - s_2 = 0x80000000$ with probability about 1. Now consider the next step.

```

s = s + iv[i+4] + Y(YMININD+i+4);
u8 s0 = P(s&0xFF);
EIV(i) = s0;
s = ROTL32(s, 8) ^ (u32)s0;

```

At the end of this step, the probability that $EIV_1[i+4] = EIV_2[i+4]$, and $s_1 = s_2$ is 1. So for the above 5 steps, $s_1 = s_2$ with probability p_2 . Once the two s are identical, in the following steps $i+5, i+6, \dots, i+ivsize-1$, the s_1 and s_2 remain the same, and $EIV_1[i+5] = EIV_2[i+5]$, $EIV_1[i+6] = EIV_2[i+6]$, \dots , $EIV_1[i+ivsize-1] = EIV_2[i+ivsize-1]$.

Then $IV[i]$ and $IV[i+1]$ are used again to update s and EIV. With similar analysis, we can show that at the end of the updating, $EIV_1 = EIV_2$, $s_1 = s_2$ with probability about $(p_2)^2 \times \frac{1}{255} \approx 2^{-24}$. (As shown in the next subsection, this probability is about $2^{-22.9}$.)

The IV difference at three bytes. In the above analysis, the difference is chosen at only three bytes, $iv_1[i] \oplus iv_2[i] = 0x80$, the most significant bit of $iv_1[i]$ is 1, $iv_2 \neq iv_2[i+1]$, $iv_2[i+4] \oplus iv_1[i+4] = 0x80$, and the most significant bit of $iv_1[i+4]$ is 0 ($i \geq 2$). For this type of IV difference, we can generalize it so that $iv_1[i]$ and $iv_2[i]$ can choose other differences instead of $0x80$. In fact, **once we set the difference as $iv_1[i] - iv_2[i] = iv_2[i+4] - iv_1[i+4]$, $iv_1[i+1] \neq iv_2[i+1]$ ($i \geq 2$), then the two keystreams can be identical with probability close to that given above.** For two IVs different only at three bytes, if $iv_1[1] \oplus iv_2[1] = 1$, $iv_1[2] \neq iv_2[2]$, and $iv_1[1] - iv_2[1] = iv_2[5] - iv_1[5]$, then this IV pair is also weak. We will give the detailed analysis of the generalized differences later.

3.3 Improve the attack

We are able to reduce the amount of IVs required to generate identical keystreams. The idea is to generate more IV pairs from a group of IV. For the IV pair with two-byte difference $iv_1[i] \oplus iv_2[i] = 1$, $iv_1[i+1] \neq iv_2[i+1]$, if $iv[2]$ choose all the 256 values, then we can obtain $255 \times 255 = 2^{15.99}$ IV pairs with the required differences from 512 IVs. Thus with 512 chosen IVs, the probability that there is one pair of identical keystreams becomes $2^{15.99} \times 2^{-23.2} \approx 2^{-7.2}$. With about $2^{7.2} \times 512 = 2^{16.2}$ IVs, identical keystreams can be obtained.

Experiment 2. We use 2^{16} random 128-bit keys in the improved attack. For each key, we generate 512 128-bit IVs with the values of the least significant bit of $iv[4]$ and the eight bits of $iv[5]$ choosing all the 512 possible values, while all the other 119 IV bits remain unchanged for each key (but those 119 IV bits are random from key to key). Then we obtain $255 \times 255 = 2^{15.99}$ IV pairs with the required difference. Among these $2^{16} \times 2^{15.99} \approx 2^{32}$ IV pairs, 447 IV pairs result in identical keystreams.

The above experiment shows that with $2^{16} \times 512 = 2^{25}$ selected IVs, 447 IVs result in identical keystreams. It shows that two identical keystreams appear for every $\frac{447}{2^{25}} = 2^{16.2}$ IVs.

For the IV pair with three-byte difference, the similar improvement as above can be applied.

Experiment 3. We use 2^{16} random 128-bit keys in the improved attack. For each key, we generate 512 128-bit IVs with the values of the most significant bit of $iv[4]$ and the eight bits of $iv[5]$ choosing all the 512 possible values, and the most significant bit of $iv[8]$ is different from the most significant bit of $iv[4]$, while all the other 118 IV bits remain unchanged for each key (but those 118 IV bits are random from key to key). Then we obtain $255 \times 255 = 2^{15.99}$ IV pairs with the required difference. Among these $2^{16} \times 2^{15.99} \approx 2^{32}$ IV pairs, 570 IV pairs result in identical keystreams.

The above experiment shows that with $2^{16} \times 512 = 2^{25}$ selected IVs, 570 IVs result in identical keystreams. It shows that two identical keystreams appear for every $\frac{570}{2^{25}} = 2^{15.9}$ IVs.

4 Conclusion

The attacks show that the Py and Pypy are practically insecure. In the application, if the IVs are generated from counter, or if the IV is short (such as 3 or 4 bytes), then the special IVs (with the differences as illustrated in this paper) appear with high chance, and identical keystreams can be obtained with large probability.

References

1. E. Biham, J. Seberry, "Py: A Fast and Secure Stream Cipher Using Rolling Arrays." Available at <http://www.ecrypt.eu.org/stream/ciphers/py/py.ps> .
2. E. Biham, J. Seberry, "Pypy: Another Version of Py." Available at <http://www.ecrypt.eu.org/stream/papersdir/2006/038.pdf>
3. P. Crowley, "Improved Cryptanalysis of Py." Available at <http://www.ecrypt.eu.org/stream/papersdir/2006/010.pdf> .
4. S. Paul, B. Preneel, S. Sekar, "Distinguishing Attack on the Stream Cipher Py." *Fast Software Encryption - FSE 2006*, to appear.