Does ZK-Crypt version 1 flunk a repetition test?

Daniel J. Bernstein *

djb@cr.yp.to

ZK-Crypt version 1 is one of the stream ciphers submitted to eSTREAM, the ECRYPT Stream Cipher Project.

An October 2005 paper "The distinguishing attack on ZK-Crypt cipher" claims that the ZK-Crypt output streams are detectably biased: distinguishable from uniform with 97% reliability after 2^{28} output bits, for example. The ZK-Crypt authors disputed this attack in February 2006, first at the SASC 2006 workshop and then as part of the documentation of ZK-Crypt version 2.

Two papers at the SASC workshop reported that ZK-Crypt version 1 also flunks some simple IV-diffusion tests. It seems clear that ZK-Crypt version 1 is being withdrawn in favor of version 2. But this does not mean that the dispute regarding the October 2005 paper should be left unresolved. The community should properly assign credit for breaking ZK-Crypt version 1, and should be prepared to properly evaluate subsequent papers along the lines of the October 2005 paper. Did the October 2005 paper break ZK-Crypt version 1?

A closer look reveals that the October 2005 paper falls far below normal standards of scientific verifiability. The authors never define the distinguishing function that they claim to have found. They present a table claimed to be output from software implementing the distinguishing function, but they do not provide the software. They describe a parametrized "book stack test," but many critical parameters are unspecified. They rely on an "obvious" statement regarding the effect of the "book stack test" on uniform random data, but that statement is false for many parameter choices, as pointed out by the ZK-Crypt authors.

On the other hand, it is clear that the "book stack test" is a reinvention of a compression method normally called "move to front," which tries to exploit irregularities in the distribution of distances between repeated words. One can see all such irregularities without deciphering the October 2005 paper: simply draw a graph of the distribution of distances between repeated output words from ZK-Crypt version 1, and compare it to a graph of the distribution of distances between repeated output words from a perfect cipher.

I carried out this experiment and, from the graphs, immediately saw the problem with ZK-Crypt: the output words tend to repeat at smaller distances than output words from a perfect cipher. The first 10 million 4-byte output words of ZK-Crypt version 1 almost always (i.e., for almost all keys and IVs) have more than 12000 repetitions, whereas 10 million independent uniform random 4-byte

* Permanent ID of this document: b6f4ca45a01f114782fe80341e9b60a9. Date of this document: 2006.03.02. This document is final and may be freely cited.

words almost always do not. In short, ZK-Crypt version 1 flunks a very simple repetition test.

Here is a program that, given `/dev/urandom` as input, tries 20 (key, IV) pairs; for each pair, it prints the number of repetitions in the first 10 million 4-byte output words of ZK-Crypt version 1. The program uses the ZK-Crypt authors' `zk-crypt.c`, which in turn uses `ecrypt-config.h`, `ecrypt-machine.h`, `ecrypt-portable.h`, `ecrypt-sync.h`, and `ecrypt-zk-crypt.h`; I copied all of these files from version 162 of the eSTREAM benchmark suite. My first run of the program printed 12355, 12336, 12261, 12282, 12368, 12235, 12247, 12243, 12121, 12428, 12148, 12293, 12311, 12271, 12125, 12286, 12256, 12198, 12293, 11974.

```
#include <stdio.h>
#include "zk-crypt.c"

#define WORDS 10000000
unsigned char out[WORDS * 4];
int pos[WORDS];

int cmp(const void *first,const void *second)
{
  int p1 = *(int *) first;
  int p2 = *(int *) second;
  if (out[4 * p1] > out[4 * p2]) return 1;
  if (out[4 * p1] < out[4 * p2]) return -1;
  if (out[4 * p1 + 1] > out[4 * p2 + 1]) return 1;
  if (out[4 * p1 + 1] < out[4 * p2 + 1]) return -1;
  if (out[4 * p1 + 2] > out[4 * p2 + 2]) return 1;
  if (out[4 * p1 + 2] < out[4 * p2 + 2]) return -1;
  if (out[4 * p1 + 3] > out[4 * p2 + 3]) return 1;
  if (out[4 * p1 + 3] < out[4 * p2 + 3]) return -1;
  if (p1 > p2) return 1;
  if (p1 < p2) return -1;
  return 0;
}

void out_fill(void)
{
  static ECRYPT_ctx c;
  static u8 k[16];
  static u8 iv[16];
  static u8 in[WORDS * 4];
  int i;

  for (i = 0;i < 16;++i) k[i] = getchar();
  for (i = 0;i < 16;++i) iv[i] = getchar();
```

```
  ECRYPT_init();
  ECRYPT_keysetup(&c,k,128,128);
  ECRYPT_ivsetup(&c,iv);
  ECRYPT_encrypt_bytes(&c,in,out,WORDS * 4);
}

main()
{
  int i;
  long long total;
  int loop;

  for (loop = 0;loop < 20;++loop) {
    out_fill();
    for (i = 0;i < WORDS;++i) pos[i] = i;
    qsort(pos,WORDS,sizeof(pos[0]),cmp);
    total = 0;
    for (i = 0;i + 1 < WORDS;++i)
      if (memcmp(out + pos[i] * 4,out + pos[i+1] * 4,4) == 0)
        ++total;
    printf("%lld\n",total);
    fflush(stdout);
  }
  return 0;
}
```

For comparison, the following program uses `/dev/urandom` data in place of ZK-Crypt output words, and carries out the same test. My first run of the program printed 11614, 11571, 11945, 11556, 11677, 11496, 11656, 11561, 11547, 11650, 11660, 11667, 11609, 11650, 11571, 11615, 11522, 11713, 11549, 11692.

```
#include <stdio.h>

#define WORDS 10000000
unsigned char out[WORDS * 4];
int pos[WORDS];

int cmp(const void *first,const void *second)
{
  int p1 = *(int *) first;
  int p2 = *(int *) second;
  if (out[4 * p1] > out[4 * p2]) return 1;
  if (out[4 * p1] < out[4 * p2]) return -1;
  if (out[4 * p1 + 1] > out[4 * p2 + 1]) return 1;
  if (out[4 * p1 + 1] < out[4 * p2 + 1]) return -1;
```

```
  if (out[4 * p1 + 2] > out[4 * p2 + 2]) return 1;
  if (out[4 * p1 + 2] < out[4 * p2 + 2]) return -1;
  if (out[4 * p1 + 3] > out[4 * p2 + 3]) return 1;
  if (out[4 * p1 + 3] < out[4 * p2 + 3]) return -1;
  if (p1 > p2) return 1;
  if (p1 < p2) return -1;
  return 0;
}

void out_fill(void)
{
  int i;
  for (i = 0;i < WORDS * 4;++i) out[i] = getchar();
}

main()
{
  int i;
  long long total;
  int loop;

  for (loop = 0;loop < 20;++loop) {
    out_fill();
    for (i = 0;i < WORDS;++i) pos[i] = i;
    qsort(pos,WORDS,sizeof(pos[0]),cmp);
    total = 0;
    for (i = 0;i + 1 < WORDS;++i)
      if (memcmp(out + pos[i] * 4,out + pos[i+1] * 4,4) == 0)
        ++total;
    printf("%lld\n",total);
    fflush(stdout);
  }
  return 0;
}
```