Dark Stream Cipher

Karl Zander – pvial@kryptomagik.com, karl.zander00@gmail.com

Abstract.

Dark is an encryption algorithm that obfuscates data so that no one is able to read it without the key. It is small and fast, maintaining a 256-bit internal key stream register and utilizes a non-invertible output function.

1. Introduction

Dark is a fast encryption algorithm aimed at providing no attack vector other than brute force.

Dark consists of a key setup function which produces a 256-bit key stream register, and a key stream generator which outputs blocks of 4 pseudorandom bytes (32-bits) that are XORed with the input plaintext.

The cipher accepts a single key length, 256-bits. A nonce or initialization vector is required for encryption. The length of the nonce must be 128-bits.

2. Design goals

While designing Dark, I developed the following requirements:

The algorithm must not directly expose the register/state.

The algorithm must operate on all 256 bits of the state/register per encryption round and all 256 bits of the key must be required to decrypt even a single byte of data.

The algorithm must employ a public nonce or initialization vector that must not give the attacker more than 50% advantage in the key stream generation process.

The algorithm must produce a non-repeating stream (without period) of bytes with uniform characteristics.

The algorithm must pass known statistical testing for random number generators.

The algorithm must be extremely fast to implement in software.

3. Key Setup

Dark's key setup function prepares the 256-bit register for encryption. The register is represented as a 8 x 64-bit word array. First the key is loaded 4 bytes per register into the register array. Then the nonce is XORed with the first two elements in the register. Next, J (a pseudorandom incrementor) is computed by summing J and each element of the register. Afterwards, the encryption round function is applied to the register

To illustrate in more detail, the key stream array is initialized to the length of the key and set to zero. The key stream array is referred to as k[]. Each byte of the key stream is added mod 256 to zero byte array. During this process, j is computed which j is simply the sum of each progressive byte mod 256 plus a mod 256 counter.

```
void keysetup(unsigned char *key, unsigned char *nonce) {
   uint32_t n[4];
   r[0] = (key[0] << 24) + (key[1] << 16) + (key[2] << 8) + key[3];
   r[1] = (key[4] << 24) + (key[5] << 16) + (key[6] << 8) + key[7];
   r[2] = (key[8] << 24) + (key[9] << 16) + (key[10] << 8) + key[11];
   r[3] = (key[12] << 24) + (key[13] << 16) + (key[14] << 8) + key[15];
   r[4] = (key[16] << 24) + (key[17] << 16) + (key[18] << 8) + key[19];
   r[5] = (key[20] << 24) + (key[21] << 16) + (key[22] << 8) + key[23];
   r[6] = (key[24] << 24) + (key[25] << 16) + (key[26] << 8) + key[27];
   r[7] = (key[28] << 24) + (key[29] << 16) + (key[30] << 8) + key[31];

   n[0] = (nonce[0] << 24) + (nonce[1] << 16) + (nonce[2] << 8) + nonce[3];
   n[1] = (nonce[4] << 24) + (nonce[5] << 16) + (nonce[6] << 8) + nonce[7];
   n[2] = (nonce[8] << 24) + (nonce[9] << 16) + (nonce[10] << 8) + nonce[11];
   n[3] = (nonce[12] << 24) + (nonce[13] << 16) + (nonce[14] << 8) + nonce[15];

   r[4] = r[4] ^ n[0];
   r[5] = r[5] ^ n[1];
   r[6] = r[6] ^ n[2];
   r[7] = r[7] ^ n[3];

   for (int i = 0; i < 8; i++) {
      j = (j +  r[i]) & 0xFFFFFFFF;
   }
   F(j, ct);
}
```

4. Encryption round function

The encryption round function F() permutes the state/register before processing the next 32-bit block of plaintext. This is done using the following steps:

First, we establish that we will run this function on every array element of the register. Second, we reserve a 32-bit space in memory where the element is copied to for later processing.

```
uint32_t x;
x = r[i];
```

Then the register element is added to the element to the right of it and J (the pseudorandom incrementer) mod 2^32.

r[i] = (r[i] + r[(i + 1) & 0x07] + j) & 0xFFFFFFFF;

Next the element is XORed with the original value we saved in x.

r[i] = r[i] ^ x;

Then the element is rotated left 2 bits.

r[i] = rotate(r[i], 2);

J is then recomputed for the next round and increment the counter.

j = (j + r[i] + counter) & 0xFFFFFFFF;
counter = (counter + 1) & 0xFFFFFFFF;

5. Output function

Now that the register has been prepared by the round function, we are ready to encrypt data. To output 32-bits (8 bytes) from the register, we add registers 0 and 6, the result is XORed with register 1, that result is added by the 5th register, that result is XORed with the 2nd register, that result is added to the 4th element, that result is XORed with the 3rd register and finally, the final result is added by register 7 mod 2^32.

The output function could best be expressed by the following code:

output = (((((((r[0] + r[6]) ^ r[1]) + r[5]) ^ r[2]) + r[4]) ^ r[3]) + r[7]) & 0xFFFFFFFF;

The 32 output bits are unpacked to make 4 output bytes that XORed with the 4 input plaintext bytes. Extra unused bytes of the key stream are discarded once all input bytes have been processed.

5. Non-invertibility of the Output Function

The output function cannot be easily inverted because of the sheer infeasibility of calculating 8 x 32-bit words at any given point.

6. Cryptanalysis

The output of Dark has not been able to be distinguished from a random bit sequence nor can it be differentiated from an ideal cipher. Nothing can be learned from frequency analysis.

Under a known plaintext attack of n bytes, one may not invert the output function to regain the key state bytes. One may know n bytes of the plaintext and the remainder of the message will remain secure.

An attack to roll back the register state of the first encryption round to the point where the key was loaded is an effort of 2^544 steps. Brute forcing the key is a faster attack at 2^256 steps.

7. Statistical Properties

Dark was subjected to the Diehard battery of tests (dieharder). Overall, Dark passed the tests.

Tests were conducted on streams of 1 and 2 gigabytes of data with 100 different key, nonce pairs.

Dark was also subjected to NIST Statistical Test Suite battery of tests and passed.