Amagus Stream Cipher

Karl Zander – pvial00@gmail.com

Abstract.

Amagus is an encryption algorithm that obfuscates data so that no one is able to read it without the key. It is fast and formidable, maintaining a 1024-bit internal state and accepting variable key lengths.

1. Introduction

Amagus 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 1024-bit key stream register, and a key stream generator which outputs blocks of 8 pseudorandom bytes (64-bits) that are XORed with the input plaintext.

The cipher accepts variable key lengths from 256 bits to 1024 bits. A nonce or initialization vector is required for encryption. The length of the nonce must be 128-bits (could be variable in other implementations).

2. Design goals

The algorithm must not directly expose the register/state.

The algorithm must operate on all 1024 bits of the state/register per encryption round.

The algorithm must adhere to the strict avalanche effect and produce proper confusion and defusion.

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

The key setup process for Amagus is very simple. The key is loaded into the state register 64 bits at a time. Then the first two 64 bit registered are XOR'ed with the nonce. Lastly, the round function is run on the state 10 times or however many rounds are specified in the implementation, 10 rounds is standard.

4. Round function

Amagus has a fast round function that utilizes ARX operations and is broken up in two phases.

1. Mixing phase

   - The mixing phase does one of a few operations on alternating words in the state. The first word is added to a word, the second word is XOR'ed with a word, the third word is XOR'ed is the XOR of the word and another rotated left so many bits. This pattern continues throughout the rest of the state.

2. Transposition/diffusion phase

   - In this phase the state columns are alternated through the same function.

The actual round function in C code:

```
void *amagus_F(struct amagus_state *state) {
   int r;
   for (r = 0; r < amagus_rounds; r++) {
      state->r[0] += state->r[6];
      state->r[1] ^= state->r[15];
      state->r[2] = amagus_rl((state->r[2] ^ state->r[12]), 9);
      state->r[3] += state->r[9];
      state->r[4] ^= state->r[11];
      state->r[5] = amagus_rr((state->r[5] ^ state->r[10]), 6);
      state->r[6] += state->r[13];
      state->r[7] ^= state->r[8];
      state->r[8] = amagus_rl((state->r[8] ^ state->r[3]), 11);
      state->r[9] += state->r[1];
      state->r[10] ^= state->r[4];
      state->r[11] = amagus_rr((state->r[8] ^ state->r[7]), 7);
      state->r[12] += state->r[0];
      state->r[13] ^= state->r[2];
      state->r[14] = amagus_rl((state->r[14] ^ state->r[0]), 3);
      state->r[15] += state->r[5];

      state->r[15] += state->r[6];
      state->r[2] ^= state->r[15];
      state->r[14] = amagus_rl((state->r[14] ^ state->r[12]), 9);
      state->r[4] += state->r[9];
      state->r[13] ^= state->r[11];
      state->r[6] = amagus_rr((state->r[6] ^ state->r[10]), 6);
      state->r[12] += state->r[13];
      state->r[8] ^= state->r[8];
      state->r[11] = amagus_rl((state->r[11] ^ state->r[3]), 11);
      state->r[10] += state->r[1];
      state->r[1] ^= state->r[4];
      state->r[3] = amagus_rr((state->r[3] ^ state->r[7]), 7);
      state->r[5] += state->r[0];
      state->r[7] ^= state->r[2];
      state->r[9] = amagus_rl((state->r[9] ^ state->r[0]), 3);
      state->r[0] += state->r[5];
   }
}
```

5. Output function

The output function used by Amagus is straightforward. All state words are XOR'ed together to produce a 64 bit output word which is then XOR'ed with the input data.

6. Cryptanalysis

TBD

7. Statistical Properties

Amagus was subjected to the Diehard battery of tests (dieharder). Overall, Amagus passed the tests. Tests were conducted on streams of 1 gigabyte of data with 100 different key, nonce pairs. Amagus was also subjected to NIST Statistical Test Suite battery of tests and passed.