Distinguishing Attack on MAG

Simon Künzli and Willi Meier

FH Aargau, 5210 Windisch (Switzerland),
{s.kuenzli,w.meier}@fh-aargau.ch

Abstract. MAG is a synchronous stream cipher submitted to the ECRYPT stream cipher project. We present a very simple distinguishing attack (with some predicting feature) on MAG, requiring only 129 successive bytes of known keystream, computation and memory are negligible. The attack has been verified.

1 Brief Description

In the standard version of the stream cipher MAG [1], the internal state consists of 127 registers R_i of 32 bit size, as well as a carry register C of 32 bit size. The secret key is used to initialize all registers $R_0, \ldots, R_{126}$ and C (where the details of the key setup is not important for the attack). In order to produce the keystream, MAG is applied iteratively; a single iteration consists of an update and output period. The description of the update does not seem to be consistent in the paper and in the provided code; we will refer to the code (however, the attack is of very general nature and will may work for other versions). In update period i, the carry C and register R_i are modified. In a first step of the algorithm, two neighboring registers are compared in order to determine the operation for the carry update, and in a second step, the carry is used to update register R_i; more precisely,

$$C' = \begin{cases} C \oplus R_{i+1} & \text{if } R_{i+2} > R_{i+3} \\ C \oplus \bar{R}_{i+1} & \text{otherwise} \end{cases} \tag{1}$$

$$R'_i = R_i \oplus C'. \tag{2}$$

Here, $\oplus$ denotes the XOR operation, and $\bar{R}$ denotes the complement of R; updated variables are primed. Notice that a register R_i is updated only once in 127 iterations, whereas the carry C is updated in each step of iteration. We point out that comparison of registers is the only operation on words, whereas XOR and complementation are operations on bits. It remains to describe the (cryptographic) output of MAG: in output period i, the string $R_i \mod 256$ is sent to the keystream K_i (notice that addition of indices in R_i is performed modulo 127, whereas indices in K_i are continuous).

2 Distinguishing Attack

The goal of a distinguishing attack is to distinguish the keystream of the cipher from a truly random sequence. We assume that the attacker knows some part of the keystream (known-plaintext); the first 127 bytes of keystream K_i reveal the 8 least significant bits (lsb's) of all registers R_i, and the additional keystream byte K_{127} reveals the 8 lsb's of the updated register R'_0.

Given these 128 successive bytes of keystream $K_0, \ldots, K_{127}$, it is possible to compute two strings, one of them corresponding to the next keystream byte K_{128}: first, Eq. 2

defines how to reveal the corresponding carry, namely $C' \mod 256 = R_0 \oplus R'_0 \mod 256$. According to Eq. 1, the carry is updated by $C'' = C' \oplus R_1$ or by $C'' = C' \oplus \bar{R}_1$ (with equal probability). Finally, the register R_1 is updated by $R'_1 = C'' \oplus R_1$. These relations can be reduced modulo 256 (in order to make use of the known keystream bytes) and combined; using the fact that they also hold for other indices, we conclude

$$K_{i+128} = \begin{cases} K_i \oplus K_{i+127} \oplus K_{i+1} \oplus K_{i+2} & \text{with } \Pr = 1/2 \\ K_i \oplus K_{i+127} \oplus K_{i+1} \oplus \bar{K}_{i+2} & \text{with } \Pr = 1/2 \end{cases} \tag{3}$$

Prediction of K_{i+128} may be used to distinguish the keystream of the cipher from a truly random sequence: given the actual keystream K_{i+128}, the attacker may verify if it corresponds to one of the two results of Eq. 3. If not, the keystream is not produced by MAG. If yes, the keystream is produced by MAG with a probability of error corresponding to $\alpha = 1/128$. In order to reduce the error α (false positives), more keystream may be used to verify Eq. 3. Furthermore, the distinguisher may be used to recover some part of the state; each byte of keystream reveals one bit of information, namely the path of the branching. However, we did not study the state-recovery attack in more detail.

We conclude that the design of MAG has substantial weaknesses; revealing some part of the internal state, and sparse use of operations on words may be delicate choices of design for a secure stream cipher.

3 Example of an Attack

The attack was verified, using the code provided in [1]. In Tab. 1, we give an example of keystream produced by the standard implementation of MAG, initialized with the zero seed. According to the previous section, we verify the non-randomness of the last red-colored byte K_{128} (where the index counts from 0): Eq. 3 yields that either $K_{128} = \texttt{0x05} \oplus \texttt{0xF0} \oplus \texttt{0x53} \oplus \texttt{0x16} = \texttt{0xB0}$ or $K_{128} = \texttt{0x05} \oplus \texttt{0xF0} \oplus \texttt{0x53} \oplus \texttt{0xE9} = \texttt{0x4F}$; obviously, the first result is the appropriate one.

Table 1. Some example keystream produced by standard implementation of MAG for the zero seed.

0x05 53 16 29 77 23 33 5C 05 FC F8 57 26 1A 98 6B
0xAD 33 E2 2F 02 1B 3D 2E 82 44 82 E9 BF 8E C3 88
0x0F FE 88 21 2E 5D 6E EA 6B 62 1C 62 4D 7B 51 27
0x75 CE 34 53 CA 2A 32 B9 56 23 43 2C 19 5C 14 AE
0xC5 42 BA A8 59 11 8F 41 F0 48 2B 81 4D 52 C7 EA
0xB0 F5 BA 76 62 9B 93 7D 93 24 9C C2 7B 70 EE 3D
0x44 02 B8 E3 CF DF 36 7D EE F3 00 79 20 23 7A 60
0xB3 8B AD 3E 1B F4 BB 57 AF 99 53 AF 5C C7 88 F0
0xB0 23 6B 16 8E 3D 57 0D 0C A0 29 BD 19 F0 51 5B

References

1. Rade Vuckovac. MAG My Array Generator. ECRYPT Stream Cipher Project Report 2005/001, 2005, http://www.ecrypt.eu.org/stream.