MAG alternating methods notes

0.0 Introduction

It should be noted that the MAG endeavor is still in an early stage and was not specifically intended for eSTREAM project. The main objective of MAG paper [1] (and source code) is to promote algorithm's selection criteria peculiarities as a new approach for random number generation. Consequently a lot of implementation details (especially cryptographic details) are still undefined. From that perspective the distinguishing attack (DA) described by [2] is a good starting point to improve MAG cipher design. Following section will describe DA attack in short (and quantify predictability of DA) and propose MAG design changes to avoid DA.

1.0 Distinguishing Attack

DA shows how MAG secure stream can be differentiated from a random stream. The differentiation comes from dependencies between adjacent cells evolution wise (see figure 1). The update of a single cell is formulated by equation (2) [2]

$$R_i' = R_i \oplus C'. \qquad (2)$$

where R's are adjacent cells evolution wise and C is carry is updating element formulated by equation (1) [2]

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

To make MAG stream secure it is proposed that one -> many relationship (a hash) may serve to conceal the internal state of evolution. The simplest solution was to use mod 256 (grey fields figure 1). And that 256 mod choice of hashing was used to find dependencies in hashed stream and consequently to distinguish hashed stream from a random one. Specifically, by xoring two neighboring cells evolution wise, the corresponding carry part can be obtained by equation (a) [2].

$$C' \bmod 256 = R_0 \oplus R_0' \bmod 256 \qquad \text{(a)}$$

That information is used to skip the calculation of the stream cell array wise, leaving only one branching to guess. (equation (3) [2])

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

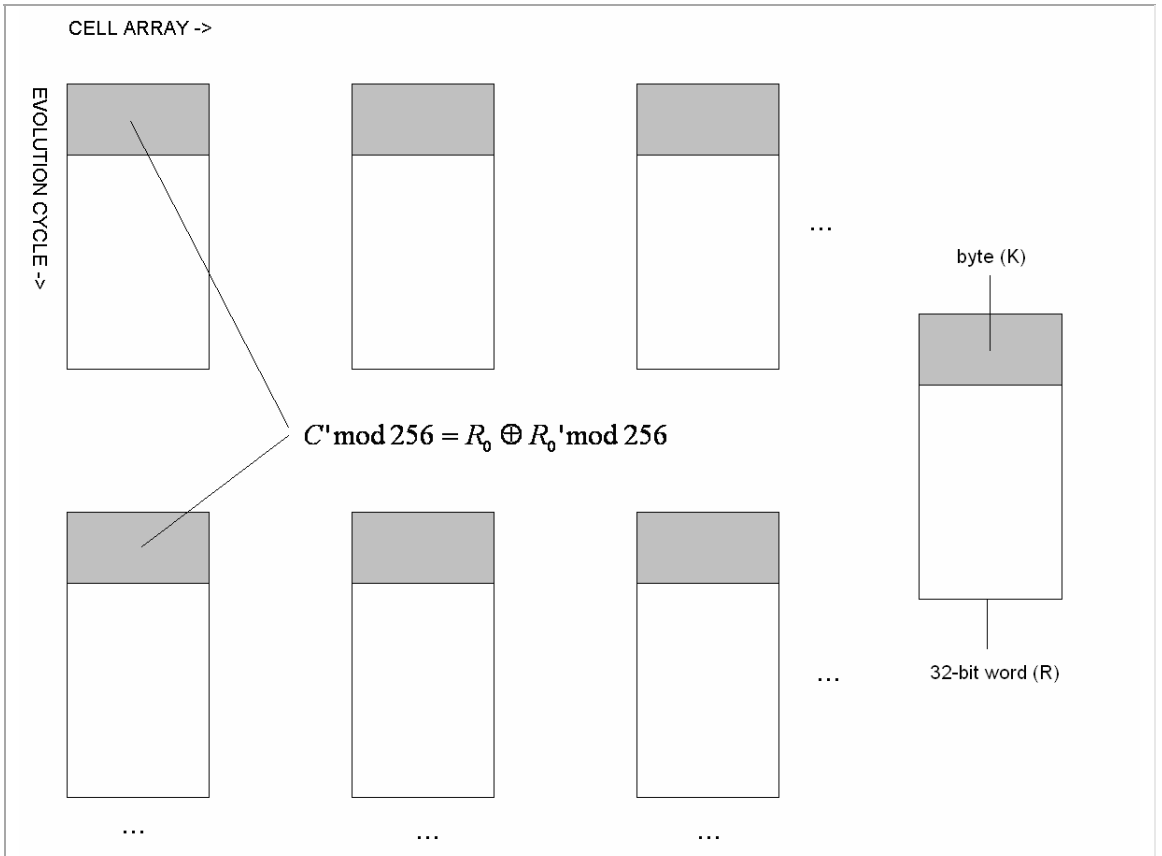

Figure 1 mod 256 variant

It is stated that DA has some predicting feature. But the predicting feature is never explained in more detail or quantified. Most importantly, it is not estimated how the DA improves over brute force attack.

There are some figures concerning predicting probabilities. From the equation no 3 it appears that the next unknown byte can be predicted with ½ probability, meaning that next byte probability is ¼ and so on. To predict one evolution cycle (127 bytes) probability of getting it right is on par with the brute force attack. Beyond one evolution cycle worth of stream prediction cost exceeds brute force cost. It can be concluded that prediction cost of DA is exponential $O(2^n)$.

Also DA claims that a part of internal state may be recovered, namely branching path of the algorithm. Stating that and questioning the use of sparse functions, authors of DA see substantial weakness in algorithm. “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.”[2]

First about revealing some part of the internal state: The usability of the branching path knowledge is highly questionable because the probability of branching is ½ (equation no (3) [2]). Seeing weakness in the revealed branching path of MAG is equivalent to the statement that statistics somehow improves predictability of the fair coin flipping.

From that perspective (equation 3) if only the branching path is used as an PRNG output, that stream should satisfy two abstract requirements for secure PRNG [3 BS].

Second objection is sparse use of operations. MAG design is based on the computational irreducibility paradigm, more precisely on use of selection criteria as legitimate math (anti-math) operation. It is not clear how sparse use of operations is relevant to the design of MAG (unless it was addressed to the nonchalant use of mod 256).

2.0 Changes to avoid attack

It would be easy to argue security of the branching path as a PRNG alone, but performance will be similar to the LFSR (one bit output for several operations).

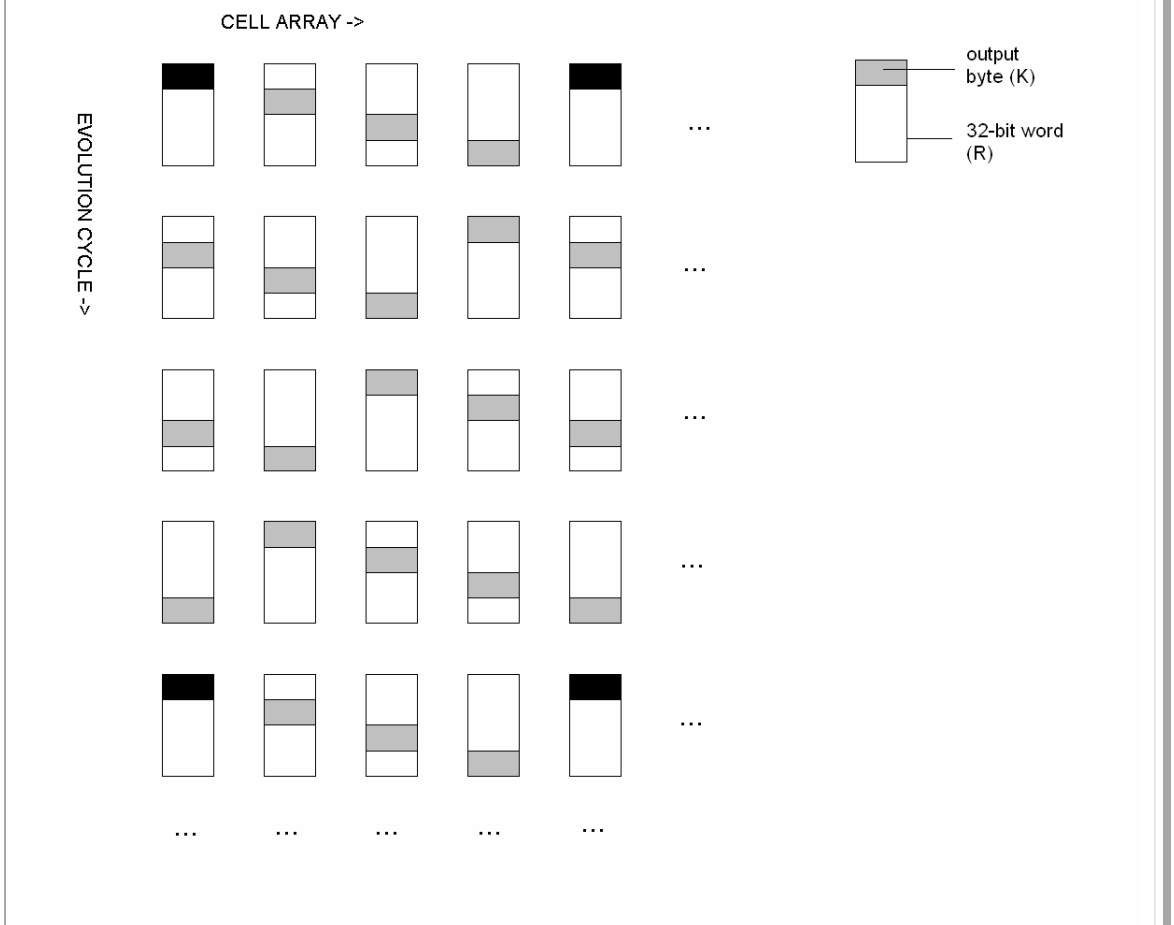

Figure 2 quarter word alternating variant

However it is still easy to avoid DA and improve performance. Instead of mod 256 (taking byte from the same spot figure 1) the output byte should be harvested from different positions within 32-bit words (see figure 2). With that change performance is still one byte output for several operations, but DA is not feasibile anymore. Xoring output bytes aligned evolution wise (black cells) can not expose corresponding carry element because there is three intermediate steps which remain unknown.

It looks that three covered steps is more than enough to keep secret the internal state of the evolution and to avoid DA. Consequently it is possible to keep secret internal state, to be resistant to the DA and to double performance of the algorithm. See figure 3.

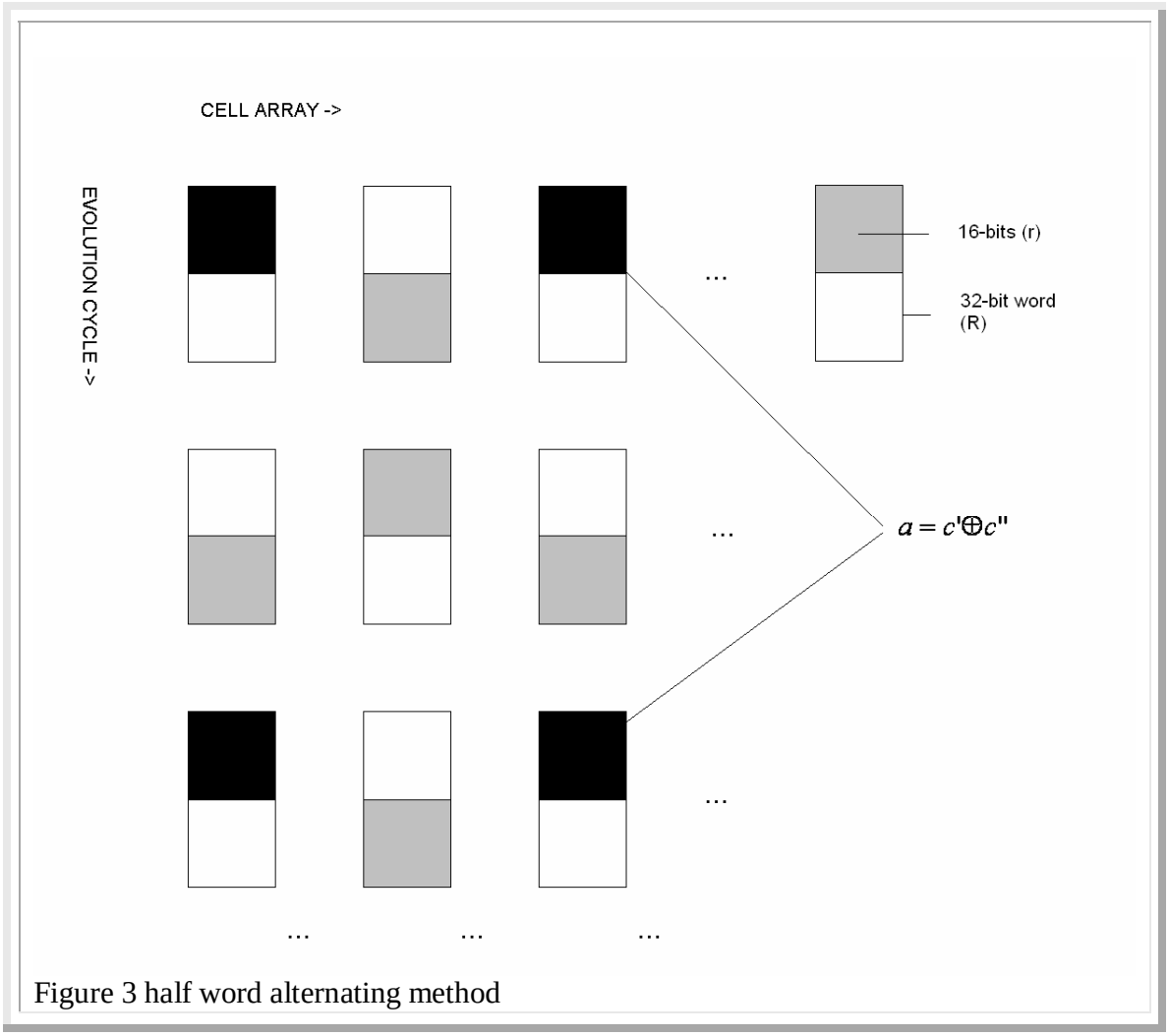

Figure 3 half word alternating method

Having knowledge of a half of the cell, does not permit the calculation of evolution because complete knowledge of the state is needed. The relationship between cells is concealed as well, because there is one unknown steep (white field). The relationship

exploited by DA is now changed ("Eq. 2 defines how to reveal the corresponding carry, namely $C' \bmod 256 = R_0 \oplus R_0' \bmod 256$ "[2]).

If *c* is enumerated as a half of *C* (carry element) and *r* is enumerated as a half of *R* (cell), the relation (xoring product evolution wise) between observable cells halves is

$r_0' \oplus (r_0' \oplus c' \oplus c'') = a$ => $a = c' \oplus c''$ equation (b)

where the carry halves (*c'* and *c"*)are unknown. From there it is not possible to arrive to equation 3. Also equation 3 requires consequent mod 256 of the cells which are not possible to get by half word alternating output scheme.

3.0 Hardware and software performance notices

Following section contains some pointers concerning future hardware and software implementations in the light of hashing changes.

As was argued in MAG paper the software performance should be in the RC4 performance neighborhood. But that performance is suggested for 32-bit environment. It should be noted that performance of MAG is scalable twofold.

First, the level of security needed (a key length) has not impact to the MAG performance, although the memory requirements rises linearly. For example 256 bit security needs array of 256 elements, 512 bit security needs 512 elements array and so on for the same computational cost (indifference to the key length).

Second, the choice of the word width environment directly determines performance. For 32-bit word systems output is 16 bits (half-alternating variant), for 64-bit word systems output is 32 bits and so on for the same number of evolution operations.

4.0 Summary

The DA shows how MAG output can be distinguished from a random output. It exploits mod 256 hashing method proposed by MAG. The mod 256 hash enables DA to skip considerable amount of calculation and to guess only one branch. Even with that amount of information, the predicting power is limited because prediction cost rises exponentially and it is not cumulative (amount of known stream is irrelevant). The internal state and the key are not compromised by DA in any way. It is easy to avoid DA because it relies on mod 256 peculiarities (sampling byte from the same spot). The fact is that the 256 mod hashing is not fundamental to the overall scheme. Consequently it is easy to avoid DA because there is a numerous ways of how to hide internal state of algorithm including the quarter alternating (figure 2) and half alternating (figure 3) scheme proposed (source code can be found [4]).

5.0 References

[1] Rade Vuckovac. *MAG My Array Generator*. ECRYPT Stream Cipher Project Report 2005/001, 2005, http://www.ecrypt.eu.org/stream.
[2] Simon Künzli and Willi Meier. *Distinguishing Attack on MAG*. ECRYPT Stream Cipher Project Report 2005/001, 2005, http://www.ecrypt.eu.org/stream.

[3] PRNG requirements:

-Passing all empirical tests imaginable.

-Prediction of the next bit in sequence is impossible with unrestricted knowledge of all previous bits in sequence.

B. Schneier *Applied Cryptography protocols, algorithms and source code in C*; Second Edition; QA76.9.A25S35 1996; ISBN 0-471-12845-7

[4] **http://geocities.com/radence_v/**