Rekeying Issues in the MUGI Stream Cipher

Matt Henricksen and Ed Dawson

Information Security Institute,
Queensland University of Technology,
GPO Box 2434, Brisbane,
Queensland, 4001, Australia.
{m.henricksen, e.dawson}@qut.edu.au

Abstract. MUGI [15] is a word-based stream cipher designed for 64-bit architectures. It uses a 128-bit master key and a 128-bit initialization vector to populate a large non-linear feedback shift register (NLFSR) and additional non-linear state (NLS). In standard benchmarks on 32-bit processors, MUGI suffers from poor key agility because it is implemented on an architecture for which it is not designed, and because its NLFSR is too large relative to the size of its master key. This paper proposes a variant of MUGI, entitled MUGI-M, to enhance key agility, and concludes with an analysis of its security and performance characteristics.

Key words: stream cipher, MUGI, MUGI-M, key initialization, key agility

1 Introduction

MUGI [15] is a Pseudo Random Number Generator (PRNG) designed for use as a stream cipher. It uses a 128-bit master key and a 128-bit initialization vector. Its design strength of 128 bits is commensurate with the length of the key.

MUGI's structure is based on the PANAMA PRNG [5], which can be used either as a stream cipher or hash function. A schematic generalization of PANAMA and MUGI is shown in Figure 1. The update function Υ is composed of a linear sub-function λ and a non-linear sub-function ρ. The function λ updates the buffer, using input from both the buffer and the state. The function ρ updates the state, using additional input from the buffer. An output filter f operating on the state produces the keystream.

MUGI is targeted to 64-bit architectures, which means that in terms of speed, it is currently non-competitive with many recent word-based stream ciphers. On the Intel Pentium 4, it has a throughput of 25.2 cycles per byte, compared to 3.7, 6.7 and 9.2 cycles per byte respectively for Rabbit [3], Dragon [4], and Turing [13]. This is a situation that will almost certainly change when 64-bit architectures finally become commonplace. MUGI's mediocre performance in software is not due entirely to the mismatch between the algorithmic requirements and implementation characteristics. It has a large state space, which can lead to poor key agility, through a complex and lengthy key initialization process.

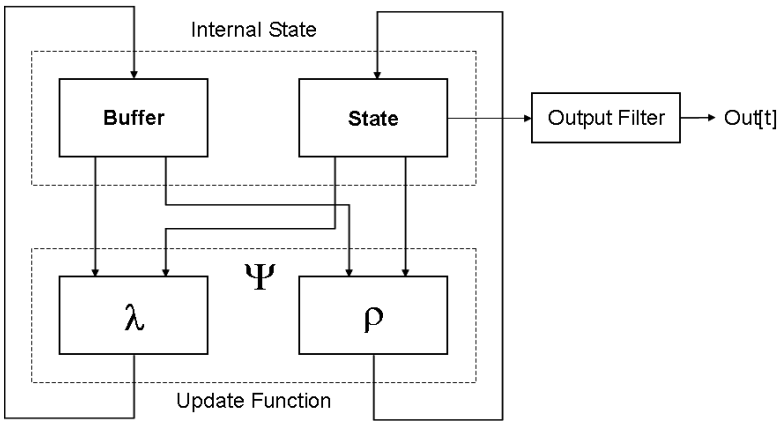

Fig. 1. Generalization of the PANAMA and MUGI Structures

In this paper, we show how to improve MUGI's key agility for both 32- and 64-bit architectures. In Section 2, we describe the MUGI keystream generation and key initialization algorithms. In Section 3, we review previous cryptanalysis of MUGI, which leads to an interesting insight on the role of the buffer in the cipher. In Section 4, we discuss a peculiarity with the key initialization algorithm. In Section 5, we analyze further the performance of MUGI relative to other word-based stream ciphers, and suggest strategies that could be used to improve it, culminating in an algorithm for a "modified MUGI" in Section 6. In Section 7 we perform a security and implementation analysis for the new algorithm. In Section 8, we summarize the contribution of paper.

2 The MUGI Algorithm

The MUGI algorithm uses 64-bit words. MUGI's internal state contains a 3-stage Non-linear Feedback Shift Register (NLFSR) denoted a, and a 16-stage Linear Feedback Shift Register (LFSR), denoted b. The output filter produces 64 bits of the output from state a at each iteration.

The non-linear function ρ is a target-heavy Feistel network structure:

$$\begin{aligned} a_0[t+1] &= a_1[t] \\ a_1[t+1] &= a_2[t] \oplus F(a_1[t], b_4[t]) \oplus C_1 \\ a_2[t+1] &= a_0[t] \oplus F(a_1[t], b_{10}[t] \lll 17) \oplus C_2 \end{aligned}$$

where C_1 and C_2 are known constants, $(M \lll k)$ indicates leftwise k-bit rotation of M, and F is a function that uses the components of the round function of the Advanced Encryption Standard [6]. Note that the 192-bit state receives at most 128 bits of new material each time ρ is called. Each of the state words is used in

a different way: a_0 is used to provide new material to the buffer; a_1 is used for mixing in the F function; and a_2 is used for output and feedback.

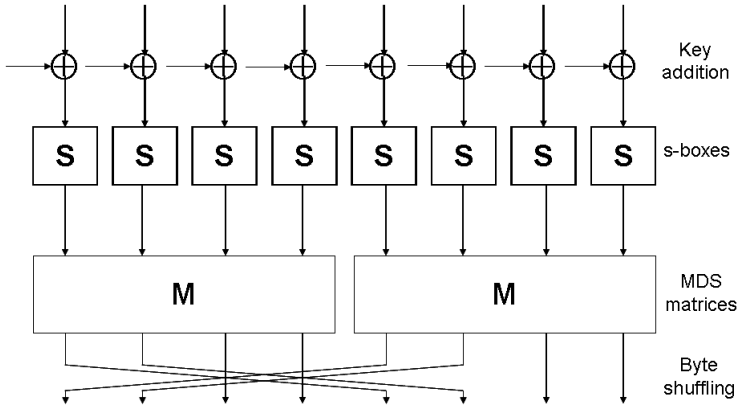

Fig. 2. MUGI F Function

The details of the F function are shown in Figure 2. The function has four layers. In the first layer, which resembles key addition in an Substitution Permutation Network (SPN), eight bytes from a buffer word are added to each of eight state bytes. In the second layer, the state is modified by eight parallel applications of the AES s-box. The third layer contains a repeated Maximum Distance Separable (MDS) matrix. The final layer consists of a word-based permutation. The polynomials used in the MDS are identical to those used in AES.

Denoting stage i $(0 \leq i \leq 15)$ of the buffer as b_i and stage j $(0 \leq j \leq 2)$ of the state as a_j, the details of function λ are as follows:

$$
\begin{aligned}
&b_i[t+1] = b_{i-1}[t] (i \neq 0, 4, 10) \\
&b_0[t+1] = b_{15}[t] \oplus a_0[t] \\
&b_4[t+1] = b_3[t] \oplus b_7[t] \\
&b_{10}[t+1] = b_9[t] \oplus (b_{13}[t] \lll 32)
\end{aligned}
$$

where $b_i[t+1]$ and $a_i[t+1]$ are the content of stage i of buffer b and respectively state a after the completion of t iterations.

Output Filter

Each application of the Υ function produces new values within the state a. The output filter selects the 64-bit block a_2 to output as the keystream.

Initialization and Rekeying

The initialization process of MUGI consists of five phases. All must be executed in full during rekeying of a master key. Only phases three, four and five are executed during rekeying of an initialization vector.

Phase 1: Master Key Injection The 192-bit MUGI state a is initialized using the 128-bit master key K. The key is divided into two segments $K_0 \parallel K_1$ and state a set, using known constant C_0, as follows:

$$a_0 = K_0$$
$$a_1 = K_1$$
$$a_2 = (K_0 \lll 7) \oplus (K_1 \ggg 7) \oplus C_0$$

Phase 2: State Mixing and Buffer Initialization The non-linear state function ρ is used to mix the state a a total of sixteen times, using a null buffer. After each iteration, a stage in the NLFSR buffer is filled with key-dependent material from the state word a_0. The last stage in the buffer is filled first; therefore, the first stage is filled using key material which has undergone the most mixing:

$$b(K)_{15-i} = (\rho^{i+1}(a[-48], 0))_0 \quad 0 \leq i \leq 15$$

Phase 3: Initialization Vector Injection The 128-bit initialization vector $I = I_0 \parallel I_1$ is added to the mixed state a in a similar way to the key injection.

$$a[-32]_0 = a[-33]_0 \oplus I_0$$
$$a[-32]_1 = a[-33]_1 \oplus I_1$$
$$a[-32]_2 = a[-33]_2 \oplus (I_0 \lll 7) \oplus (I_1 \ggg 7) \oplus C_1$$

Phase 4: Further State Mixing The state is again mixed, using a null buffer, sixteen times. At the end of this phase, the state is represented by:

$$a[-16] = \rho^{16}(a[-32], 0)$$

Phase 5: State and Buffer Mixing The rekeying procedure is finished by iterating the state and buffer sixteen times using the Υ function, and discarding the resulting keystream.

$$a[0] = \Upsilon^{16}(a[-16]), b(K))$$

3 Related Work

In [14] the designers of MUGI analyze their cipher. They claim that MUGI is immune to linear cryptanalysis because the minimum number of active s-boxes within an approximation is 22, and the maximum linear probability of

an s-box is 2^{-6}. Consequently the maximum probability of an approximation is 2^{-132}; this is insufficient to attack the cipher given its design strength of 128 bits. They leverage the studied properties of the AES round function to claim immunity against a resynchronization attack that uses differential, linear or integral cryptanalysis.

In [7], it is shown that MUGI is not vulnerable to a linear masking attack due to the difficulty in finding a biased linear combination of the inputs and outputs of the non-linear function ρ. Also the large size of the state (1,216 bits) precludes a time-memory-data attack. The dependence of the state and buffer upon each other makes discovery of divide and conquer and correlation attacks non-trivial, and to date, none have been discovered. They note that MUGI passes all common statistical tests.

In [12], Mihaeljevic studies a variant of MUGI in which MDS matrices are excluded from the F component of the ρ update function. Because MUGI uses the AES s-box, which is well known to produce over-defined and sparse equations, the simplified MUGI can be subjected to an XL attack. However, the report [12] does not produce any definite conclusions about the complexity of the attack, except that increasing the length of the key could increase the design strength above the attack complexity (which would make the attack successful). Also Mihaeljevic [12] need not exclude linear operations like the MDS from the attack; these enable the production of additional equations which should reduce the complexity of the attack, although increase the difficulty in rendering the over-defined equations.

In [10], Golic analyses the linear function λ using a system of recurrences in b_4 and b_{10} , and solved using generating functions. From this, he discovers the period of the subsequences related to the recurrences is equal to or less than 48, and the linear complexity is 32. These properties are considered too small for use in a cryptographic application, although no attack has been forthcoming on this basis. Golic studies a simplified MUGI in which the buffer is made autonomous by decoupling the feedback from the state. Linear cryptanalysis is applied to both the simplified and full versions of MUGI — in both cases, the attack succeeds when compared to the large state size, but requires greater complexity than brute forcing the key. The attack is much easier on the simplified version, proving the success of the non-linear feedback between the buffer and the state. Golic finds that the algorithm is immune to the XL attack due to the large state and complex rekeying algorithm.

In [2], Biryukov and Shamir analyze the non-linear state (NLS) of MUGI. They find that the security of MUGI is very sensitive to small changes in the design of the ρ function, and the output filter, both of which operate on the NLS. For example, they describe practical attacks in which the output filter selects from state words a_0 or a_1, or when a_2 is chosen by the filter after the evaluation of ρ. The work of [10] in determining buffer recurrences in b_4 and b_{10} greatly simplifies the complexity of this last attack. The main part of the paper concerns an attack that allows the contents of the non-linear state to be recovered knowing only words b_4 and b_{10} of the buffer, given only three output

words and a time complexity of 2^{32}. However, guessing these buffer words is equivalent in effort to guessing the secret key. Also, knowledge of the state at any point in time does not automatically allow determination of the state at a future point, since it is quickly mixed with unknown buffer words.

4 An Observation on Key Initialization

As seen in Section 2, MUGI rekeying involves five phases. In phase two, the fifteenth word of the buffer (b_{15}) is assigned the output $(\rho^1(a,0))_0$, which is the value of the state variable a_0 after a single invocation of the ρ function. In the ρ function, the a_0 word is modified simply by replacing its value with that of a_1 (that is, one third of the state is not changed by the ρ function). Since each buffer word is only updated once in the second phase, at the end of phase two, b_{15} contains the unmodified key word K_1, which entered the state as a_1.

Stages three and four of the initialization do not touch the buffer at all, meaning that at the start of the final stage, after thirty-two rounds of the ρ function, half of the key material is present in the buffer in its unmixed state. An attacker has to work backwards through only sixteen rounds of ρ to obtain K_1. While there is no known way of doing this faster than brute force, this is still significantly less effort than is suggested by the lengthy and complex initialization process.

5 Improving Key Agility of MUGI

Compared to many other contemporary ciphers, MUGI has a large ratio of key size to state size. This can be seen in Table 1, which is ordered by increasing ratio of key to state size.

Table 1. Key to State Size of Modern Word Based Stream Ciphers

Cipher	Key Size (bits)	State Size (bits)	Ratio
Helix [9]	256	160	**1:0.6**
Turing [13]	256	544	**1:2.1**
SNOW [8]	256	576	**1:2.2**
Rabbit [3]	128	513	**1:4.0**
Dragon [4]	256	1,088	**1:4.2**
MUGI [15]	128	1,216	**1:9.5**
RC4 [1]	128	2,048	**1:16.0**
Scream [11]	128	2,432	**1:19.0**
HC-256 [16]	256	65,536	**1:256.0**

One implication of a large state size is reduced key agility, since the key initialization algorithm needs to touch each element of the state. A rule of thumb

observed in SNOW, Dragon, HC-256 and MUGI, all of which mix the internal state using the update function, is that the function should be called twice for each element in the state. Scream chains each element in its masking table by iterating the update function four times on the previous element. Consequently, MUGI, Scream and HC-256, all of which have large states, also have lengthy key initialization functions and are poor performers in terms of key agility. While Dragon and MUGI have comparable state sizes, Dragon's key is twice the length, providing better security per byte of state. Its update function is much faster, so the key initialization algorithm, at a throughput of 11 cycles/byte, is completed in approximately twenty percent of the time required by MUGI.

There are two obvious strategies that can be considered to improve the performance of MUGI. The first is to migrate the cipher from a 64- to 32-bit design, by halving the size of each of the components, including the stages in the NLFSR and the words within the non-linear state. This has the added advantage that the design of MUGI now matches the architecture on which it is most likely to be implemented. It has the fatal weakness that the non-linear state naturally houses a 96-bit rather than 128-bit key. This key size is too small. Also the reduction in size of components necessitates rethinking the design of the core function F, which contains eight 8×8 s-boxes and two 32×32-bit MDS matrices. Using eight 4×4 s-boxes increases the maximum characteristic probability across four rounds from 2^{-132} to 2^{-50}, and using four 8×8 s-boxes increases the maximum probability across four rounds to 2^{-100}. In both cases, this is a significant loss of security. In this case the trade-off of security to benefit efficiency is inappropriate.

An alternative strategy is to leave the non-linear state and its ρ update function as they are, and act upon the deficiencies of the buffer. By reducing the buffer to 8×64-bit stages, for a total state size of $512 + 192 = 704$ bits, the speed of the rekeying strategy is increased significantly, the speed of the update function is slightly increased, and the security is marginally decreased. The state size is still more than five times the size of a 128-bit master key. This is the strategy that will be adopted in the modification of MUGI.

Shrinking the buffer involves altering the taps used for feedback, and also the indices to stages used by the non-linear filter function. As the size of the buffer is halved, it is a natural progression to also halve the indices of the taps and stages, leaving their order unaltered. One effect of this strategy is that some stages receive feedback from adjacent stages.

Another improvement is to remove phase four of the keying scheme. This mixes the non-linear state sixteen times. Consequently, by the end of the initialization, each element of the non-linear state and the buffer has been modified forty-eight and thirty-two times respectively. By removing this stage, each element of the non-linear state and buffer has been altered sixteen times. This brings the cipher into line with the design principles of other ciphers, and the rule of thumb that each element of the state should be touched by a non-linear function (at least) twice.

To remove the property discussed in Section 4, we change the state word that is fed into the buffer in phase two. If a_1 is used as feedback to the buffer, then the state word a_0 reflects the contents of the buffer word last modified. This is a benign property, since it is destroyed immediately upon commencement of phase three. But using a_2 as feedback in phase two avoids this relationship, with the obvious proviso that as it is used post-initialization to generate output, its role in providing feedback to the buffer is localized to the key initialization algorithm.

6 An Improvement: the MUGI-M algorithm

In the modified algorithm, denoted MUGI-M, the only changes that effect the update sub-function ρ are the changes in the buffer words used as inputs:

$$\begin{aligned} a_0[t+1] &= a_1[t] \\ a_1[t+1] &= a_2[t] \oplus F(a_1[t], b_2[t]) \oplus C_1 \\ a_2[t+1] &= a_0[t] \oplus F(a_1[t], b_5[t] \lll 17) \oplus C_2 \end{aligned}$$

The update sub-function λ operates on the buffer as follows:

$$\begin{aligned} b_i[t+1] &= b_{i-1}[t] (i \neq 0, 2, 5) \\ b_0[t+1] &= b_7[t] \oplus a_0[t] \\ b_2[t+1] &= b_1[t] \oplus b_3[t] \\ b_5[t+1] &= b_4[t] \oplus (b_6[t] \lll 32) \end{aligned}$$

The initialization process of MUGI-M consists of four phases. All must be executed in full during rekeying of a master key. Only phases three and four are executed during rekeying of an initialization vector.

Phase 1: Master Key Injection The 128-bit MUGI-M state a is initialized as per Phase 1 of the MUGI algorithm.

Phase 2: State Mixing and Buffer Initialization The non-linear state function ρ is used to mix the state a a total of eight times, using a null buffer. After each iteration, a stage in the buffer is filled with key-dependent material from the state word a_2. The last stage in the buffer is filled first; therefore, the first stage is filled using key material which has undergone the most mixing:

$$b(K)_{7-i} = (\rho^{i+1}(a[-16], 0))_2 \quad 0 \leq i \leq 7$$

Phase 3: Initialization Vector Injection The 128-bit initialization is added to the mixed state a as per Phase 3 of the MUGI algorithm.

Phase 4: State and Buffer Mixing The rekeying procedure finishes by iterating the state and buffer eight times using the combined Υ function, and discarding

the resulting keystream.

$$a[0] = \Upsilon^8(a[-8]), b(K))$$

Test vectors for this algorithm are presented in Appendix A. Code is available from the authors upon request.

7 Analysis of MUGI-M

Table 2 shows the contrast in efficiency between MUGI and MUGI-M on the Intel Pentium 4 (Northwood) processor. In particular, there is an improvement in MUGI-M of 200% in the speed of rekeying an initialization vector, and 170% in full rekeying. There is a modest 30% increase in the speed of the keystream generation, due likely due to reduced register pressure and smaller buffer loops.

Table 2. Efficiency of MUGI and MUGI-M on the Intel Pentium 4

Cipher	Keystream Generation	Key Initialization (IV)	Key Initialization (Full)	Ratio
Cycles per iteration				
MUGI	181	4987	7540	**1:27.6:41.7**
MUGI-M	140	1652	2784	**1:11.8:20.0**
Cycles per byte				
MUGI	25.2	36.8	55.7	**1:1.5:2.2**
MUGI-M	19.4	12.2	20.6	**1:0.6:1.1**
Ratio	**1.3:1**	**3.0:1**	**2.7:1**	

The attacks discussed in Section 3 are ineffective against MUGI for the following reasons: the effectiveness of the highly non-linear state function ρ, which leverages the properties of the AES block cipher; the large size of the buffer; the feedback between the internal state and the buffer; and the complex rekeying strategy. None of the attacks rely on properties of the buffer other than its size. Golic [10] argues that the properties of the buffer, when considered autonomously, are cryptographically poor. This argument is deflected by the fact that the buffer is coupled to the non-linear state, and that it is unrealistic to map the buffer properties directly to those of the whole cipher. However, from this it can be claimed that by changing the location of the taps in the buffer, we are not altering any special properties of the buffer, which was constructed in an ad-hoc manner. We are aiming to repair the performance of MUGI rather than engender it with additional security properties. In the remainder of this section, the resistance of MUGI-M against classes of individual attacks is considered.

Block-cipher style attacks rely on the properties of the non-linear function: for example, the maximum differential and linear probabilities across the function. Given that only the size of the buffer, and the location of its taps have been changed, the analysis of MUGI in [7] remains unchanged. The analysis relies extensively on the properties of the 64-bit F function, which is a modified AES round function. It is well-known that this function is resistant against differential and linear attacks. This is because the s-boxes in the F function have a maximum probability of 2^{-6}, although almost half of the s-box characteristics have a probability of 2^{-7}. To launch a successful attack against the F function requires a differential that incorporates fewer than ten active s-boxes, as $2^{-7\times10} < 2^{-64}$. The analysis in [7] of the intertwined MDS matrices indicates that they guarantee at least eight active s-boxes over four rounds. If a differential style attack can be launched against MUGI, it will need to use fewer than six words of keystream. The F function exhibits a vulnerability to integral cryptanalysis across no fewer than four, and no more than nine rounds. The synchronous nature of the cipher means that the attacker does not have sufficient control over the inputs to launch it on either MUGI or MUGI-M. The resilience of MUGI-M against block-cipher style attacks appears to be the same as that of MUGI. If an attack of this style affects one, it will presumably affect the other.

Linear cryptanalysis The self-evaluation report of MUGI [14] includes an analysis of linear cryptanalysis incorporating both the non-linear state and the buffer. This form of linear cryptanalysis consists of two phases: the first determines a linear approximation of ρ. In the second, a path is searched to acquire an approximation that consists only of output bits (as the internal state is not available to the attacker). For MUGI-M, the first phase remains unaltered from that of MUGI: if an approximation can be found that includes fewer than twenty-two active s-boxes, linear cryptanalysis may be possible. The second phase does not depend upon the length of the buffer; since the nature of the buffer has not been fundamentally altered, the analysis of MUGI applies equally to MUGI-M.

Time-Memory-Data trade-off attacks MUGI-M is immune to time-memory-data trade-off attacks because it has a small key size relative to the size of the buffer. For a brute-force equivalent attack with $T = 2^{128}$, $M^2 \times D^2 = 2^{896}$. Assuming that a limit is placed on generating 2^{128} bits of keystream under one key, then to launch an attack requires 2^{287} gigabytes of memory. This is clearly infeasible.

Divide and conquer attacks A successful divide and conquer attack on MUGI in which the components are autonomous, and that determines the contents of the components sequentially, has a complexity of $2^{192} + 2^{1024}$ (rather than the brute-force complexity of $2^{192} \times 2^{1024}$). The shorter buffer length of MUGI-M reduces this complexity to $2^{192} + 2^{512}$. This analysis ignores the fact that the components are not autonomous, and that the complexity may be much

higher. The complexity of the attack needs to be less than 2^{127} to be considered successful, given the 128-bit design strength of MUGI. Therefore, divide and conquer attacks are very unlikely to succeed against MUGI-M.

Correlation attacks A correlation attack on MUGI or MUGI-M requires a measure of correlation between the NLS and the NLFSR. No measure has been found in either cipher, due to the absence of a perceivable bias in the non-linear filter, and to the feedback between the NLS and the NLFSR. A correlation attack against MUGI-M seems unlikely.

Guess and determine attacks have been successful against a number of word-based ciphers. In a guess and determine attack against a PANAMA-style cipher, a cryptanalyst can adopt one of three approaches: fix elements within the non-linear state and use them guess the contents of the NLFSR; fix elements within the NLFSR and use them to guess the contents of the NLS; or a hybrid approach in which elements from both components are guessed.

MUGI has shown resistance to guess and determine attacks because of the high non-linearity in the ρ function, and the large sizes of both the state and the buffer. Adopting either of the first two approaches outlined is fruitless, because the material guessed exceeds the number of bits in the master key, so a hybrid approach needs to be adopted. While this may be possible, no guess and determine attack has been possible, because no simple relationship between the non-linear state and the buffer has been discovered. As the buffers in MUGI and MUGI-M are similar in structure and size (relative to the master key size), and the ρ function is essentially unchanged, a guess and determine attack on one of the ciphers is likely to apply (with modifications) to the other.

Linear masking attacks depend on two factors: finding a linear approximation to the non-linear filter, and finding a linear combination of the buffer that causes the bias in the non-linear filter to vanish. To date, no effective bias has been discovered in the non-linear filter ρ of MUGI, which is unaltered in MUGI-M. We do not expect that MUGI-M is vulnerable to linear masking attacks.

Algebraic attacks depend upon developing systems of equations on the non-linear components of ciphers. In MUGI-M, the sole non-linear component is the AES s-box, which is well-known to be over-defined. The linear components of the non-linear filter and buffer allow extra equations to be added to the system. In principle, MUGI-M is vulnerable to an XL attack, with a complexity similar to that on MUGI, which shares the same non-linear filter. However, in both cases, the complexity of the XL attack exceeds the design strength of the 128-bit master key [12], and is therefore not practical.

Rekeying Attacks MUGI-M appears to be secure from rekeying attacks, despite the fact that the key initialization algorithm mixes the non-linear state

sixteen instead of forty-eight times, and the buffer sixteen instead of thirty-two times. The level of mixing per buffer stage remains the same.

Also the attacker has no control over any stage in the buffer, except indirectly through the non-linear state. No raw key material enters the buffer at any time.

Consider a resynchronization attack using multiple master keys, in which there are differences between the keys. For extra freedom, the attacker is allowed to control the difference in the initial a_2 state word. Because the F function is optimized against differential cryptanalysis, and because each of the stages in the buffer is chained to previous stages, the attacker very quickly loses the ability to track differences within the keystream. No differentials through the F function are possible after it has been iterated four times. After the population of b_6 and b_7 in phase two of the rekeying, subsequent words are affected by at least four iterations of the F function and therefore activate too many s-boxes for an effective related-key attack to be launched. In phase four, b_6 and b_7 are filled with material dependent upon all buffer words, so the low non-linearity present in these words in phase two is not a weakness.

8 Summary

In this paper we have reviewed past cryptanalysis of the MUGI stream cipher, and pointed out a peculiarity in the key initialization algorithm, whereby one key word was visible in the buffer after thirty-two out of forty-eight iterations of the update function. We determined that MUGI had poor key agility, compared to other word-based stream ciphers because its design targets 64-bit architectures, which are not yet commonly available, and because its large state size requires a lengthy key initialization process. The state size is large relative to the key size, so does not serve well the security-efficiency trade-offs in MUGI's design.

We suggested a variant of the MUGI algorithm, MUGI-M, in which the size of the buffer was halved, and the key initialization algorithm reduced from forty-eight to sixteen steps. This resulted in an improvement of 200% in the speed of rekeying an initialization vector, and 170% in full rekeying. We analysed the new variant with respect to security and determined that it remains secure against attacks, principally because we made no significant alterations to the non-linear filter, because each stage in the buffer is sufficiently modified by the key initialization algorithm, and because the buffer is still large relative to the key size. This alteration will serve the security-performance trade-off of MUGI well, both now and in the future, when 64-bit architectures, for which MUGI was designed, become commonplace.

Acknowledgements

Many thanks to Minna Yao and the anonymous referees for their feedback on this paper.

References

1. Anonymous. RC4 algorithm revealed. Posting to sci.crypt usenet group on 14 September, 1994. Available at ftp://idea.sec.dsi.unimi.it/pub/security/crypt/code/rc4.revealed.gz.
2. Alex Biryukov and Adi Shamir. Analysis of the non-linear part of MUGI. In Serge Vaudenay, editor, *Proceedings of the 12th International Workshop on Fast Software Encryption*, Lecture Notes in Computer Science. Springer-Verlag, 2005. To appear.
3. Martin Boesgaard, Mette Vesterager, Thomas Pedersen, Jesper Christiansen, and Ove Scavenius. Rabbit: a new high-performance stream cipher. In Joan Daemen and Vincent Rijmen, editors, *Proceedings of the 9th International Workshop on Fast Software Encryption*, volume 2365 of *Lecture Notes in Computer Science*, pages 325–344. Springer-Verlag, 2003.
4. Kevin Chen, Matt Henricksen, Leonie Simpson, William Millian, and Ed Dawson. Dragon: A fast word based cipher. In *Information Security and Cryptology - ICISC '04 - Seventh International Conference*, 2004. To appear in Lecture Notes in Computer Science.
5. Joan Daemen and Craig Clapp. Fast hashing and stream encryption with PANAMA. In Serge Vaudenay, editor, *Proceedings of the 5th International Workshop on Fast Software Encryption*, volume 1372 of *Lecture Notes in Computer Science*, pages 60–74. Springer-Verlag, 1998.
6. Joan Daemen and Vincent Rijmen. Rijndael. In *Proceedings from the First Advanced Encryption Standard Candidate Conference, National Institute of Standards and Technology (NIST)*, August 1998. Available at http://csrc.nist.gov/encryption/aes/.
7. Ed Dawson, Gary Carter, Helen Gustafson, Matt Henricksen, William Millan, and Leonie Simpson. Evaluation of the MUGI psuedo-random number generator. Technical report, CRYPTREC, Information Technology Promotion Agency (IPA), Tokyo, Japan, 2002. Available at www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1035_IPA-MUGI_report_final.pdf.
8. Patrik Ekdahl and Thomas Johansson. Snow - a new stream cipher, 2000. Available at http://www.it.lth.se/cryptology/snow/.
9. Niels Ferguson, Doug Whiting, Bruce Schneier, John Kelsey, Stefan Lucks, and Tadayoshi Kohno. Helix: fast encryption and authentication in a single cryptographic primitive. In Joan Daemen and Vincent Rijmen, editors, *Proceedings of the 9th International Workshop on Fast Software Encryption*, volume 2365 of *Lecture Notes in Computer Science*, pages 345–361. Springer-Verlag, 2003.
10. Jovan Golic. Security evaluation of MUGI. Technical report, CRYPTREC, Information Technology Promotion Agency (IPA), Japan, Tokyo, 2002.
11. Shai Halevi, Don Coppersmith, and Charanjit Jutla. Scream: a software-efficient stream cipher. In Joan Daemen and Vincent Rijmen, editors, *Proceedings of the 9th International Workshop on Fast Software Encryption*, volume 2365 of *Lecture Notes in Computer Science*, pages 195–209. Springer-Verlag, 2003.
12. Mihodrag Mihaeljevic. Report on security evaluation of MUGI stream cipher. Technical report, CRYPTREC, Information Technology Promotion Agency (IPA), Tokyo, Japan, 2002.
13. Gregory Rose and Philip Hawkes. Turing: a fast stream cipher. In Joan Daemen and Vincent Rijmen, editors, *Proceedings of the 9th International Workshop on Fast Software Encryption*, volume 2365 of *Lecture Notes in Computer Science*, pages 307–324. Springer-Verlag, 2003.

14. Dai Watanabe, Soichi Furuya, Hirotaka Yoshida, and Kazuo Takaragi. MUGI psuedorandom number generator, self evaluation, 2001. Available at http://www.sdl.hitachi.co.jp/crypto/mugi/index-e.html.
15. Dai Watanabe, Soichi Furuya, Hirotaka Yoshida, and Kazuo Takaragi. A new keystream generator MUGI. In Joan Daemen and Vincent Rijmen, editors, *Proceedings of the 9th International Workshop on Fast Software Encryption*, volume 2365 of *Lecture Notes in Computer Science*, pages 179–194. Springer-Verlag, 2003.
16. Hongjun Wu. A New Stream Cipher HC-256, 2004. Available at http://eprint.iacr.org/2004/092.pdf.

A Test vectors for MUGI-M

The following little-endian test vectors for MUGI-M are presented:

```
Key = 00000000000000000000000000000000
IV  = 00000000000000000000000000000000

Keystream =
0E850A7AD4E94A1C 5C97E7FBA492CC60 34738F8D04904D47
79CE86DC89D2E684 34050A91BC2555D0 8C8310A3E543DE40
F2B6B9F612381372 11036D8E55485B69 5323E5B6F05CBF32
389675E756BF490E D61618C9FAAFE00F 51BC3DA8A4C70E50
44147EFBDA308F4A D0AD8E5C38E85FD5 8F397AEA286A7761
C64694622A3599E5

Key = 0E850A7AD4E94A1C5C97E7FBA492CC60
IV  = 34738F8D04904D4779CE86DC89D2E684

Keystream =
8BB9E439BD1B632F C614E04066FAEA66 1820B17F2D7216B6
8986D48391441B8F E1B8A6D4C6A81815 B91207DC6138669A
2428795E4B67258A 7D6E0786559E0F32 E0B9DC8B34C5A6D8
C59E1BB3FD1ACA53 4395FF4AF7C9A1AC DFFDE7F86661D94D
7A37A985291598A1 AB554E72C2C7EAD2 C9125F4ACAEBE3B4
66DB2836BF75CC34

Key = 8BB9E439BD1B632FC614E04066FAEA66
IV  = 1820B17F2D7216B68986D48391441B8F

Keystream =
F4EB67A12774D27D 6FE1F36A696E8D20 0017C6166A273176
A06F58F0FAEE1B5E C1A8F9081E85FE55 A2FC5569966650F8
C44F926DFEDD99D0 5B6ECCE80E4C2057 67A9F58EED1CABF5
0500EF8D4429B3F4 90F58F5C42F74028 8C4B9D15AA7DFCE1
668491546DC4D799 4D040BCFEB46706E 365E136FC31B8204
BF9CE27566C138B1
```