Attacks on a Lightweight Cipher Based on a Multiple Recursive Generator

Lu Xiao and Gregory G. Rose

Qualcomm Incorporated
5775 Morehouse Dr, San Diego, CA 92122, USA
{lxiao, ggr}@qualcomm.com

March 18, 2009

Abstract

At IEEE GLOBECOM 2008, a lightweight cipher based on a Multiple Recursive Generator (MRG) was proposed for use in resource limited environment such as sensor nodes and RFID tags. This paper proposes two efficient attacks on this MRG cipher. A distinguishing attack is firstly introduced to identify the use of an MRG cipher that has a modulus suggested by its designers. It requires 2^{18} words of ciphertext and the most significant bit of each corresponding plaintext word. Then an efficient known plaintext attack is proposed to construct the cipher's current state and generate subkeys used for all subsequent encryption. The known plaintext attack, when targeted at the MRG ciphers optimized for efficiency, only requires $2k$ words of known plaintext and trivial computation where k is the MRG order. Even the ciphers based on complicated and inefficient MRGs can be attacked with low complexity, e.g., in the magnitude of 2^{12} words of known plaintext for all MRG ciphers with order 47, regardless of which MRG modulus is used. These two attacks indicate that the examined MRG cipher structure is seriously flawed.

1 Introduction

A Multiple Recursive Generator (MRG) usually generates a sequence of numbers with a very long period. A block cipher based on MRG was proposed at IEEE GLOBECOM 2008 [9], which uses initial MRG seeds as cipher key and generates a sequence to be used for encrypting 32-bit plaintext each time. The cipher designers suggest the use of $m = 2^{31} - 1$ as MRG modulus and select a particular case with order $k = 47$ for performance analysis [9]. The MRG cipher is claimed to be very useful and secure enough for resource limited environment such as sensor nodes and RFID tags.

In cryptanalysis, a distinguishing attack aims to identify the use of a specific cipher. Usually it examines the statistical patterns in plaintext and/or ciphertext in order to tell its difference from a random mapping. Although not intending to discover a key, a successful distinguishing attack indicates existence of security weakness in a cipher. The distinguish attack in Section 3 works with 2^{18} ciphertext words and the Most Significant Bits (MSBs) of corresponding plaintext words. These ciphertext words do not have to be encrypted with the same key or computed sequentially. In many applications, a plaintext word's MSB is much easier to obtain or guess than the whole 32 bit content. For example, if the plaintext is English language text in ASCII, all the MSBs will be zero. The distinguishing attack will still be applicable, with reduced efficiency, if there is any significant bias in these MSBs.

A known plaintext attack assumes that attackers can collect certain amount of plaintext and its corresponding ciphertext encrypted using an unknown key. This attack typically aims to deduce the cipher key with less complexity than a brute force attack. For example, linear cryptanalysis in [8] successfully breaks block cipher DES with 2^{47} known plaintexts while a brute force attack takes 2^{56} key searches. Known plaintext can be obtained through eavesdropping a communication channel and certain inferencing. For example, X.509 digital certificates [1] are supposed to be public information and widely used in IKE [6] and TLS [3]. If a cipher is used by IKE or provides lower layer protection for TLS (e.g., data link layer protection with EAP-TLS [11] as payload), the encrypted words corresponding to these certificates can be collected. Our known plaintext attack on those efficient MRG ciphers actually requires known plaintext less than a regular X.509 certificate.

In this paper, we firstly propose a distinguishing attack to reliably distinguish an MRG cipher using modulus $2^{31}-1$ [9] from encrypting with random sequence. It requires 2^{18} words of ciphertext and each corresponding plaintext word's MSB. Then a known plaintext attack is proposed on a generic MRG cipher, regardless of which modulus is used. When the MRG is optimized for efficiency, the attack requires only $2k$ words of sequential plaintext and its corresponding ciphertext, where k is the order of MRG used for this cipher (e.g., $k = 47$ in the cipher illustrated in [9]). With $48k$ decryption operations, the encryption system's state can be restored, the key is recovered, and all subsequent ciphertext can be decrypted. By comparison, a brute force attack needs 2^{32k} attempts to search k initial MRG seeds, 32 bits each. The attack also works on MRG ciphers with maximum number of distinct coefficients. Any MRG cipher with a practical order ($k \leq 47$) can be attacked successfully, with usually no more than 2^{12} words of known plaintext and trivial computation.

2 Summary of the MRG Cipher

As Figure 1 illustrates, the MRG cipher has 4 steps for encryption as the following.

- *Step 1: pseudorandom number generation.* An MRG is used to generate a sequence of numbers $X_i = (\alpha_1 X_{i-1} + \ldots + \alpha_k X_{i-k}) \bmod m$, $(i \geq k)$. The k different MRG seeds $X_0, \ldots, X_{k-1}$ are used as cipher key and initialize the state of encryption system.

The MRG coefficients $\alpha_1, \ldots, \alpha_k, m$ are publicly known values. Note that the MRG is simply a linear feedback shift register defined over the field $GF(m)$.

- *Step 2: key mixture.* A 32-bit plaintext word is divided into 4 bytes. Each byte within this word will be added with the corresponding byte in MRG output X_k. The addition is defined as byte wise addition modulo 256. The result is denoted by C. The MRG output X_i $(i \geq k)$ is called subkey in this paper to distinguish the cipher key seeded to the MRG.
- *Step 3: computation of a permutation* Π *from the MRG output* X_k. Denote $X_k = X_{k,0}||X_{k,1}||X_{k,2}||X_{k,3}$ and $\Pi = \pi_0||\pi_1||\pi_2||\pi_3$, where " || " denotes concatenation. $\pi_0 = X_{k,0} \bmod 4$; $\pi_i = n \bmod 4$, where n is the smallest integer bigger or equal to $X_{k,i}$ so that $\pi_i \notin \{\pi_0, \ldots, \pi_{i-1}\}$. The permutation Π is dynamically computed for each subsequent subkey X_i $(i > k)$.
- *Step 4: output permutation.* The permutation Π is used to shuffle bytes within the intermediate value C obtained from Step 2. Each i-th byte of the ciphertext C' comes from the π_i-th byte of the intermediate value C.

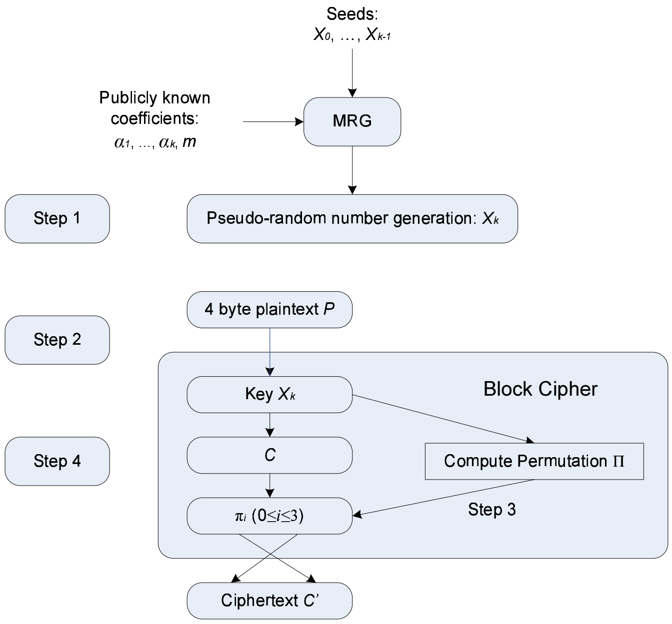

Figure 1: MRG Based Cipher [9]

Although not clearly specified, the MRG designers suggest the use of modulus $2^{31} - 1$ for efficiency improvement in cipher description [9][1]: *because* $m = 2^{31} - 1$ *and* $\alpha_1 = \ldots = \alpha_k < 2^{32}$ *are popular values for the modulo and parameter sizes, respectively, we make the following restrictions:* $m < 2^{32}$ *and* $\alpha_1, \ldots, \alpha_k < 2^{32}$.

[1]The modulus is denoted by p, instead of m, in [9]. This paper uses p to denote probability or plaintext.

3 A Distinguishing Attack

When the MRG cipher uses a modulus of 31 bits (i.e., $m = 2^{31} - 1$) as suggested in [9], each subkey X_i $(= x_{30} \dots x_0)$ also has 31 bits. The MSB of the intermediate value C $(= c_{31} \dots c_0)$ is possible to change from plaintext P $(= p_{31} \dots p_0)$ only if there is a carry bit from its lower bit position: $c_{31} = p_{31} \oplus \textit{carry-bit}_{30}$. Although the permutation Π shuffles byte order of the ciphertext C', MSB c_{31} must be transposed to a shuffled ciphertext byte's MSB.

Denote $P_c(i)$ as the probability that the i-th bit in a byte of C has a carry bit out. Since $c_i = p_i \oplus x_i \oplus \textit{carry-bit}_{i-1}$, the i-th bit location has carry out when $p_i || x_i || \textit{carry-bit}_{i-1}$="011", "101" or when $p_i || x_i$="11". Thus,

$$P_c(i) = \frac{1}{2} P_c(i-1) + \frac{1}{4}, \dots, P_c(0) = \frac{1}{4}$$

$$\Rightarrow P_c(i) = \frac{1}{2} - \frac{1}{2^{i+2}} \ .$$

After byte-wise modulo addition, C's MSB changes from p_{31} with probability $P_c(6) = \frac{1}{2} - \frac{1}{256}$. So $c_{31} = p_{31}$ with probability $1 - P_c(6) = \frac{1}{2} + p_b$, where $p_b = \frac{1}{256}$.

The attack is based on the probability bias p_b of C's MSB. Given a random sequence used for encrypting plaintext in a perfect manner, the MSBs of a ciphertext word's 4 bytes all match the corresponding plaintext word's MSB with probability 1/16. Encrypted using this MRG cipher, a ciphertext word has its 4 bytes' MSBs identical to the MSB of its plaintext word, with probability $(1/8) * (1/2 + p_b)$. The following theorem in [7] is used to estimate the number of sufficient samples to distinguish the MRG cipher:

Theorem 1 (See detailed proof in [7]) *Let M, N be distributions, and suppose the event e happens in M with probability p and in N with probability p(1+q). Then for small p and q, $O(1/pq^2)$ samples suffice to distinguish M from N with a constant probability of success.*

In our case, the event e is that the 4 bytes in ciphertext have their MSBs match the MSB of corresponding plaintext word. According to the above theorem:

$$p = \frac{1}{16}, \ p(1+q) = \frac{1}{8}(\frac{1}{2} + p_b)$$

then

$$q = \frac{1}{128}, \ \frac{1}{pq^2} = 2^{18} \ .$$

Thus, a distinguishing attack works on the MRG with samples in the magnitude of 2^{18}. Each sample needs the MSBs of a ciphertext word's 4 bytes and the MSB of its plaintext word. These ciphertext words do not have to be encrypted with the same key or computed sequentially. A word's MSB is often fixed when its content follows certain format (e.g., English language text in ASCII) or varies slightly (e.g., air pressure data collected by a wireless sensor). Thus, it is not hard to collect enough ciphertext and MSBs of plaintext for this attack.

We simulate this attack for 1000 times on 2^{18}. Denote M_e and N_e as random variables of event e in distributions using random source for encryption and using MRG cipher for encryption, respectively. We use a cryptographic random number generator, CryptGenRandom(.) provided by Microsoft CryptoAPI, to emulate a random source. With 2^{18} samples, the mean and standard deviation are 16382 and 121.68 for M_e; 16515 and 124.95 for N_e. These results are close to their theoretic values from binomial distribution with t samples:

$$E[M_e] = tp = 2^{18}p = 16384, \sigma[M_e] = \sqrt{tp(1-p)} = 123.94,$$

$$E[N_e] = tp(1+q) = 16512, \sigma[N_e] = \sqrt{tp(1+q)(1-p(1+q))} = 128.47.$$

Figure 2 illustrates the first 100 experimental tests with 2^{18} samples. It has been sufficient enough to tell the difference between M_e and N_e.

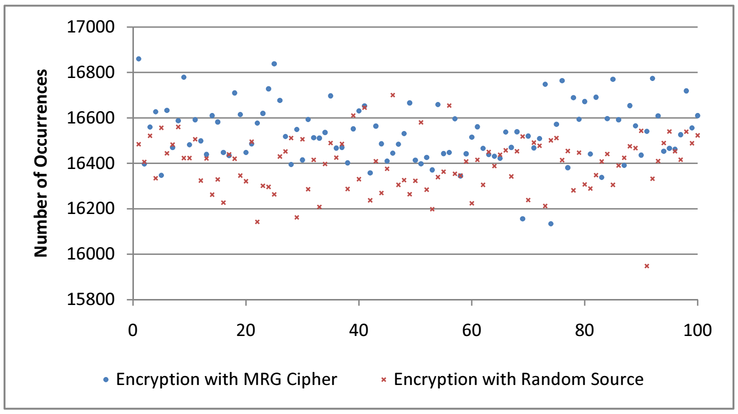

Figure 2: Distributions with 2^{18} Samples

To make the difference more significant, we run the test with 2^{22} samples. As illustrated in Figure 3, with 16 times more samples, we can distinguish an MRG cipher with more confidence.

4 A Known Plaintext Attack

The MRG cipher takes k seeds as its cipher key. An exhaustive key search requires 2^{32k} attempts with at least k words of known plaintext and ciphertext.

An improved attack is to exhaustively search subkeys $\{X_i\}$ sufficient to derive the current MRG state, which requires $2^{32}k$ encryption (or decryption) operations and at least k words of known plaintext and ciphertext. The attack can be accelerated by looking up a dictionary indexed by 32-bit plaintext and its ciphertext. As a result, k table lookups are needed with a dictionary containing 2^{64} words. A chosen plaintext attack can reduce dictionary size to 2^{32} words. However, as discussed later, these attacks may lead to multiple candidates for

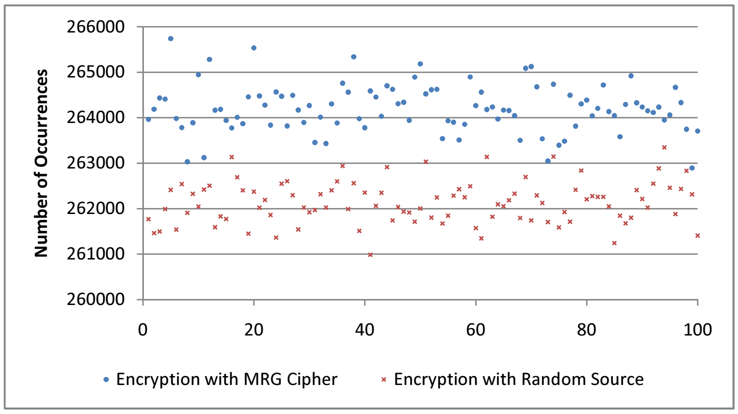

Figure 3: Distributions with 2^{22} Samples

each subkey, corresponding to the same plaintext and ciphertext. It makes verification more complicated when many MRG coefficients are nonzero.

This section illustrates an attack much more efficient than these generic attacks.

4.1 General Description

The cipher can be divided into two phases: (I) pseudorandom sequence generation using an MRG; (II) encryption of each 32-bit word using one MRG output word X_i ($i \geq k$) as subkey. We start with subkey X_k guessing. When we have candidates for $k+1$ sequential subkeys, we can use the following equation to remove wrong candidates through further subkey deducing.

$$X_i = (\alpha_1 X_{i-1} + \ldots + \alpha_k X_{i-k}) \bmod m, \ \ (i \geq 2k)$$

Although the coefficients for multiplication are not specified clearly, performance analysis in [9] indicates that any MRG with order larger than 47 is not practical (even $k = 47$ is much larger than current secure designs). We first attack the cipher in phase (II) and allow only a small number of candidates to survive from each subkey's deducing. Then we analyze the complexity to reduce spurious subkeys in phase (I) with different MRG coefficients.

4.2 Deducing Subkey X_k

The permutation Π, used in MRG cipher, shuffles the 4 bytes within the intermediate result C to form the ciphertext C'. Only 24 (i.e., 4!) permutations are possible for Π. In a known plaintext attack, we can deduce 24 C candidates from the permutations, denoted as Π^s ($0 \leq s \leq 23$), and the given ciphertext C'. Denote the C candidate derived from the permutation Π^s as C^s.

Since modulo addition is used for key mixture, we have each byte $C_i^s = P_i + X_{k,i} \bmod 256$, where C_i^s and P_i denote the i-th bytes ($0 \leq i \leq 3$) in C^s and P, respectively. Thus, each X_k

candidate, denoted by X_k^S, can be computed byte by byte:

$$X_{k,i}^S = (C_i^s - P_i) \bmod 256,\ 0 \le i \le 3.$$

With 24 candidates of C, 24 candidates for X_k can be deduced[2]. As defined in [9], subkey X_k determines the permutation Π for current block. Since each candidate X_k^s is deduced without the knowledge of how its permutation is computed, deducing X_k^s and computing Π^s are independent in this attack unless $X_k^s = X_k$. As a result, each candidate other than the right guess has a probability of $1/24$ to match Π^s assumed previously at the beginning of the attack. Those candidates other than X_k are called spurious subkeys if they can survive from permutation verification. In addition, because X_k comes from MRG modulo operation with m, only those candidates less than m can survive. Then the expected number of spurious subkeys, denoted as E_{sp}, is computed through binomial distribution, $Y_{sp} \sim B(t, p)$:

$$E_{sp} = E[Y_{sp}] = tp = (24-1)(\frac{1}{24} \cdot \frac{m}{2^{32}}) = \frac{23}{24} \cdot \frac{m}{2^{32}} < \frac{23}{24}\ .$$

The right candidate X_k certainly matches its permutation. In total, the expected number of X_k^s to survive permutation verification is:

$$E_{sv} = E_{sp} + 1 < \frac{47}{24}\ .$$

With each word of known plaintext and ciphertext, we reduce its subkey X_k candidates from 2^{32} to a variable up to 24, with expectation E_{sv}.

The pseudo code in Figure 4 deduces subkey X_k after examining 24 possible permutations. This procedure can be repeated for sequential words of known plaintext and ciphertext, to deduce subkey candidates for X_i $(i > k)$.

As the critical step of this attack, the process of subkey deducing has been implemented. All coefficients for MRG are not clearly specified in [9]. Thus, we assume that MRG outputs are as good as those from a cryptographic random number generator. In each subkey deducing process, the plaintext word and subkey X_k are simulated with CryptGenRandom(.) provided by Microsoft CryptoAPI.

We first assume that m is a prime close to 2^{32} such that the condition ($X_k <$ MRG-modulus) cannot remove any spurious subkey. In total, there are $2^{32}m$ possible combinations of plaintext and subkey. It is costly to exhaustively study their statistics of subkey deducing behaviors. As shown in Table 1, the statistical results become stable as 2^{16} or more plaintext/subkey combinations are generated randomly and tested. The mean of surviving candidate count is about 2.186 and the standard deviation is about 1.457 during the 2^{32} subkey deducing tests. The real subkey can always be deduced. About 39.1% of per word subkey deducing tests end up with a uniquely deduced subkey. The mean of surviving candidate count is slightly larger than its theoretical value E_{sv} ($\approx 47/24$). It is caused by unevenly distributed mapping from X_k to the permutation Π. This small difference does not noticeably affect the effectiveness of the attack.

[2] In this analysis, we ignore the possibility that the 24 candidates may not be unique, as some of the bytes of the subkey might be equal.

```
BYTE pi[24][4] = {
        {0, 1, 2, 3}, {0, 1, 3, 2}, {0, 2, 1, 3}, {0, 2, 3, 1},
        {0, 3, 1, 2}, {0, 3, 2, 1}, {1, 0, 2, 3}, {1, 0, 3, 2},
        {1, 2, 0, 3}, {1, 2, 3, 0}, {1, 3, 0, 2}, {1, 3, 2, 0},
        {2, 0, 1, 3}, {2, 0, 3, 1}, {2, 1, 0, 3}, {2, 1, 3, 0},
        {2, 3, 0, 1}, {2, 3, 1, 0}, {3, 0, 1, 2}, {3, 0, 2, 1},
        {3, 1, 0, 2}, {3, 1, 2, 0}, {3, 2, 0, 1}, {3, 2, 1, 0} }

procedure MRG-Subkey-Deducing(BYTE pt[4], BYTE ct[4])
begin
  BYTE      Xk[4]
  INTEGER   i, j

  for i ← 0 to 23 do
  begin
    for j ← 0 to 3 do
    begin
      Xk[pi[i][j]]  ←  (ct[j] − pt[pi[i][j]]) mod 256
    end
    if (pi[i][0...3]=Π-Compute(Xk)) and (Xk<MRG-modulus)
    /* Π-Compute(.) performs Step 3 in Section 2 */
    begin
      output Xk as one surviving subkey candidate
    end
  end
end
```

Figure 4: Pseudo Code for Subkey Deducing

When modulus $m = 2^{31}-1$ (as suggested in [9]), the condition ($X_k <$ MRG-modulus) can invalidate about half of subkey candidates. Then the expected number of surviving subkeys $E_{sv} = \frac{23}{24}\cdot\frac{m}{2^{32}}+1 \approx \frac{23}{48}+1 = \frac{71}{48}$. We apply this condition check to the previous tests, assuming that $m = 2^{31}-1$. The mean of surviving subkey count decreases to 1.727 and the standard deviation is 1.083 during the 2^{32} subkey deducing tests. Thus, the attack removes spurious subkeys more effectively, when targeted at those efficient MRGs with modulus $m = 2^{31}-1$.

4.3 Removing Spurious Subkeys

For each key between X_k and X_{2k-1} inclusively, we can remove spurious subkey combination by verifying:

$$X_{i+k} = (\alpha_1 X_{i+k-1} + \ldots + \alpha_k X_i) \bmod m,\ \ (k \le i \le 2k-1).$$

Table 1: Experimental Results of Subkey Deducing (when $m \approx 2^{32}$)

# of candidates left for X_k	Distribution [t samples], denoted as $D[t]$							
	1st $D[47]$	2nd $D[47]$	1st $D[94]$	2nd $D[94]$	$D[2^8]$	$D[2^{16}]$	$D[2^{24}]$	$D[2^{32}]$
0	0	0	0	0	0	0	0	0
1	0.468	0.298	0.362	0.362	0.430	0.392	0.391	0.391
2	0.319	0.362	0.255	0.404	0.320	0.332	0.334	0.334
3	0.106	0.149	0.202	0.128	0.105	0.120	0.121	0.121
4	0.043	0.106	0.128	0.043	0.070	0.079	0.078	0.078
5	0.043	0.021	0.032	0.032	0.043	0.041	0.040	0.040
6	0	0.064	0.011	0.011	0.023	0.023	0.023	0.023
7	0.021	0	0	0	0.008	4.99E-3	5.11E-3	5.13E-3
8	0	0	0	0	0	3.20E-3	3.14E-3	3.18E-3
9	0	0	0	0	0	0	0	0
10	0	0	0.011	0.011	0	5.07E-3	5.46E-3	5.45E-3
11	0	0	0	0	0	0	0	0
12	0	0	0	0.011	0	2.44E-4	2.36E-4	2.33E-4
13	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	2.26E-6	2.36E-6
17	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	1.55E-6	1.63E-6
19	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	6.56E-7	6.02E-7
21	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	4.17E-7	2.97E-7
Mean	1.957	2.383	2.319	2.181	2.078	2.188	2.186	2.186
Std. Dev.	1.288	1.392	1.453	1.682	1.329	1.455	1.457	1.457

A candidate for X_{i+k} is computed for each combination of existing subkey candidates for $X_i, \ldots, X_{i+k-1}$. Such a new candidate can survive subkey deducing, with probability close to $E_{sv}/2^{32}$ assuming m is close to 2^{32}. Thus, each combination containing spurious subkeys can be easily invalidated if the number of all combinations are far less than $2^{32}/E_{sv}$. The total number of these combinations can be estimated by $(E_{sv})^n$, where n is the number of nonzero coefficients in $\alpha_1, \ldots, \alpha_k$.

The value of n directly determines the efficiency of MRG, because n multiplications normally are needed to generate one subkey (unless many coefficients are identical).

4.3.1 Attack on MRG Ciphers Optimized for Efficiency

The MRG cipher proposed in [9] may use an efficient MRG with 2 to 4 nonzero coefficients in $(\alpha_1, \ldots, \alpha_k)$, as analyzed in [2]. For example, an MRG may consist of 2 nonzero coefficients for multiplication:

$$X_i = (\alpha_t X_{i-t} + \alpha_k X_{i-k}) \bmod m, \ \ (i \geq k, 1 \leq t < k, \alpha_t \neq 0, \alpha_k \neq 0).$$

This type of MRG has better performance than those with more nonzero coefficients.

It is trivial to verify that $(E_{sv})^2 << 2^{32}/E_{sv}$. For each subkey between X_k and X_{3k-1} inclusively, spurious subkeys can be removed by verifying:

$$X_{i+k} = (\alpha_t X_{i+k-t} + \alpha_k X_i) \bmod m, \ \ (k \leq i \leq 2k-1, 1 \leq t < k, \alpha_t \neq 0, \alpha_k \neq 0).$$

Once subkeys X_k to X_{2k-1} are uniquely identified, we obtain sufficient information of MRG's current state. All subsequent subkeys can be easily derived from MRG computation.

Ciphers with those MRGs consisting of 3 or 4 nonzero coefficients for multiplication can be attacked in a similar way because $(E_{sv})^3, (E_{sv})^4 << 2^{32}/E_{sv}$.

4.3.2 Attack on Complicated MRG Ciphers

The most complicated MRGs are perhaps those having maximum number of distinct nonzero values for all coefficients $\alpha_1, \ldots, \alpha_k$. We assume that its m is close to 2^{32}. Such an MRG is extremely inefficient but helpful to estimate the upper bound of the attack's complexity for all MRG ciphers.

Give k sequential subkeys to deduce, the number of events to uniquely identify a subkey is a random variable Y in a binomial distribution: $Y \sim B(k, p)$, where p is the probability that all 23 spurious subkey maps to wrong permutations. Thus, $p = (\frac{23}{24} \cdot \frac{m}{2^{32}})^{23} \approx (\frac{23}{24})^{23} \approx 0.376$. Its experimental value through statistical test is 0.391, as shown in Table 1. Y is not less than a threshold y with probability:

$$Pr(Y \geq y) = 1 - \sum_{i=0}^{y-1} \binom{k}{i} p^i (1-p)^{k-i} \ .$$

The number of all candidates for tuple $[X_i, \ldots, X_{i+k-1}]$, where $i \geq k$, can be estimated by $(E_{sv})^{k-y}$. When $(E_{sv})^{k-y} \frac{E_{sv}}{2^{32}} = 2^{-32}(E_{sv})^{k-y+1}$ is significantly small, all spurious keys can be reliably removed. Table 2 shows the relation between $Pr(Y \geq y)$ and $2^{-32}(E_{sv})^{k-y+1}$ when $k = 47$ and E_{sv} is adjusted to 2.186 (i.e., experimental mean in Table 1) from $\frac{47}{24}$. For example, when $y = 26$, $2^{-32}(E_{sv})^{k-y+1} = 0.006908$, indicating that spurious subkeys can be reliably removed; $Pr(Y \geq y) = 0.01003$, indicating the number of samples needed to uniquely identify 26 subkeys among 47 sequential subkeys is in the magnitude of $1/Pr(Y \geq y) = 99.7$. Thus, the required known plaintext is in the magnitude of $k/Pr(Y \geq y) = 4686 \approx 2^{12}$ words.

Table 2: Probabilities

y	$Pr(Y \geq y)$	$2^{-32}(E_{sv})^{48-y}$
20	2.869E-01	7.538E-01
21	1.953E-01	3.448E-01
22	1.244E-01	1.577E-01
23	7.391E-02	7.216E-02
24	4.090E-02	3.301E-02
25	2.103E-02	1.510E-02
26	1.003E-02	6.908E-03
27	4.430E-03	3.160E-03
28	1.807E-03	1.446E-03
29	6.799E-04	6.613E-04
30	2.353E-04	3.025E-04
31	7.469E-05	1.384E-04
32	2.169E-05	6.331E-05

4.4 Summary of the Attack's Complexity

4.4.1 Attack on MRG Ciphers Optimized for Efficiency

The attack requirements for the two phases are:

- Deducing subkeys: $2k$ sequential words of known plaintext and ciphertext are needed to deduce all surviving subkey candidates for $X_k, \ldots, X_{3k-1}$. Deducing each subkey needs to consider 24 candidates. In total, $24 * 2k = 48k$ decryption operations are needed without MRG computation.
- Removing spurious subkeys: $(E_{sv})^3 k$ subkey combinations need to be verified using MRG computation.

Since $(E_{sv})^3 k < 48k$, such an attack requires less than $48k$ decryption operations and $2k$ words of sequential known plaintext and ciphertext.

4.4.2 Attack on Complicated MRG Ciphers

As described previously, we need to select a convenient y to trade off between complexity and successful rate. When y satisfies the requirement $(E_{sv})^{k-y+1} << 2^{32}$:

- Deducing subkeys: the length of known plaintext can be estimated using $k/Pr(Y \geq y)$. Subkey deducing is required for these known plaintext words.
- Removing spurious subkeys: $(E_{sv})^{k-y}$ subkey combinations need to be verified using MRG.

For example, an MRG cipher with all 47 distinct nonzero coefficients can be successfully attacked with about 2^{12} known plaintext words when $y = 26$. The workload is about $(E_{sv})^{k-y} = 2^{24}$ MRG operations plus $k/Pr(Y \geq y) = 2^{12}$ encryptions. Similar attacks on other MRG ciphers with order 47 have less complexity than this worst case.

5 Cipher Performance

Although called block cipher, the MRG cipher works more similarly to a word oriented stream cipher. The MRG cipher is designed for use cases in a resource limited environment. However, even an efficient MRG with two identical nonzero coefficients (thus extremely insecure) takes one 32-bit multiplication and one modulo operation. These two operations are not efficient in either hardware or software. The MRG takes k variables (e.g., 188 bytes when $k = 47$) in memory.

By comparison, cipher RC4 only takes 258 bytes in memory for keystream generation. Many word oriented stream ciphers (e.g., SNOW[4] and SOBER[5] families of ciphers) take less memory, require no multiplication or modulus operations, and demonstrate good performance over various platforms. For example, a software implementation of SNOW 2.0 only takes 6.75 cycles/byte [10].

6 Conclusion

Our attacks work on the MRG cipher with manageably low space and time complexity. Therefore, the cipher structure proposed in [9] is seriously flawed. It should not be used for cryptographic applications even in resource restricted systems.

References

[1] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. RFC 5280: Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile, May 2008. Available at `www.ietf.org/rfc/rfc5280.txt`.

[2] L. Y. Deng. Efficient and portable multiple recursive generators of large order. In *ACM Transactions on Modeling and Computer Simulation*, volume 15, pages 1–13, 2008.

[3] T. Dierks and C. Allen. RFC 2246: The TLS protocol version 1, January 1999. Available at `www.ietf.org/rfc/rfc2246.txt`.

[4] P. Ekdahl and T. Johansson. A new version of the stream cipher SNOW. In *Proceedings of Selected Areas in Cryptography - SAC 2002*, volume 2595 of *Lecture Notes in Computer Science*, pages 47–61. Springer-Verlag, 2001.

[5] P. Hawkes and G. G. Rose. Primitive specification for SOBER-128. 2003. Available at `eprint.iacr.org/2003/081`.

[6] E. C. Kaufman. RFC 4306: Internet key exchange (IKEv2) protocol, 2005. Available at `www.ietf.org/rfc/rfc4306.txt`.

[7] I. Mantin and A. Shamir. A practical attack on broadcast RC4. In *Proceedings of Fast Software Encryption - FSE 2001*, volume 2355 of *Lecture Notes in Computer Science*, pages 152–164. Springer-Verlag, 2001.

[8] M. Matsui. Linear cryptanalysis method for DES cipher. In *Proceedings of Advances in Cryptology - EUROCRYPT'93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer-Verlag, 1994.

[9] A. Olteanu, Y. Xiao, F. Hu, and B. Sun. A lightweight block cipher based on a multiple recursive generator. In *Proceedings of IEEE GLOBECOM 2008*, December 2008.

[10] B. Preneel, B. V. Rompay, S. Örs, A. Biryukov, L. Granboulan, E. Dottax, M. Dichtl, M. Schafheutle, P. Serf, S. Pyka, E. Biham, E. Barkan, Dunkelman, J. Stolin, M. Ciet, J.-J. Quisquater, F. Sica, H. Raddum, and M. Parker. Performance of optimized implementations of the NESSIE primitives. Technical report, NESSIE, February 2003. Available at `www.cosic.esat.kuleuven.be/nessie/deliverables/D21-v2.pdf`.

[11] D. Simon, B. Aboba, and R. Hurst. RFC 5216: The EAP-TLS authentication protocol, March 2008. Available at `www.ietf.org/rfc/rfc5216.txt`.