A New Fast Stream Cipher: MAJE4

Sheena Mathew, K. Paulose Jacob

Abstract - A new fast stream cipher, MAJE4 is designed and developed with a variable key size of 128-bit or 256-bit. The randomness property of the stream cipher is analysed by using the statistical tests. The performance evaluation of the stream cipher is done in comparison with another fast stream cipher called JEROBOAM. The focus is to generate a long unpredictable key stream with better performance, which can be used for cryptographic applications.

Keywords - cryptography, stream cipher, pseudo random number generator (PRNG)

I. INTRODUCTION

Stream ciphers are an important class of symmetric encryption algorithms. Their basic design feature is the same as that for a One-Time-Pad cipher, which encrypts by XORing the plain text with a random key. But for a One-Time-Pad Cipher it is required to have a key of the same size as the plain text, which makes it impractical for most applications. While the stream ciphers require only a short random key. This key is expanded into a pseudo-random key stream, which is then XORed with the plain text to generate the cipher text. Again the same key stream is used to decrypt by XORing with the cipher text to form the plain text as shown in Fig. 1. The security of the stream cipher rests in the key. So the random number generators occupy a central place in cryptographic designs owing to their property of picking numbers unpredictably and in using these numbers to choose cryptographic keys [6-13].

Here the design goal of stream cipher is to efficiently generate pseudo-random bits, which are identical to truly random bits.

PRNGs (Pseudo Random Number Generators) used for cryptographic purposes are required to be
1. of maximum period to accommodate the long length of the transmitted message.
2. fast to speed up the process.
3. difficult to analyse, since analysis could penetrate the cryptographic system.
4. capable of producing a good distribution of values.

The cipher MAJE4 is designed for using a key of size 128 or 256 bit. MAJE4 produces a pseudorandom stream, which can be used to XOR the plain text of any length. The aim of the work is to design a secure stream cipher, which is highly secure in software.
II. DESIGN CONSIDERATIONS FOR MAJE4

1. The encryption sequence should have a large period. A pseudo random number generator uses a function that produces a deterministic stream of bits that eventually repeats itself. The longer the key, the longer it takes for a brute force attack and more difficult to do the cryptanalysis.

2. The key stream should approximate the properties of a true random stream as possible.

3. Suitable for hardware or software. Uses only primitive computational operations commonly found on micro-processors.

4. Simple and Fast. Use simple algorithm, which is easy to implement and eases the task of determining the strength of the algorithm.

5. Low memory requirement. To make it suitable for devices with restricted memory.

6. Mixed operators. The use of more than one arithmetic and / or Boolean operators complicates cryptanalysis. Use primitive operators like + and ^ since these operators do not commute and hence cryptanalysis becomes more difficult.

7. Variable number of rounds. An increase in the number of rounds increases cryptanalytic strength. Also it increases the encryption / decryption time. A variable number of rounds permit the user to make a compromise between security and execution speed.

III. DESCRIPTION OF MAJE4

The mathematical operators used are

1. Addition: Addition of words, denoted by +

2. Bitwise exclusive OR: This operation is denoted by ^.

3. Right shift operation: The right shift of word x right by y bits is denoted by x>> y.

All the above-mentioned design considerations were taken care while designing MAJE4 stream cipher. MAJE4 has a key length of 128 or 256-bit. Here the randomness property has been tested with the primary statistical tests like frequency test, serial test, runs test, poker test and auto correlation test. Since it uses only primitive computational operators like +, ^, >> etc, it is suitable for hardware and software implementations. MAJE4 uses an algorithm, which is easy to implement and fast also. The memory requirement for MAJE4 is less and hence it is suitable for devices having restricted memory. Here the nonlinearity is obtained by alternative usage of + and ^ operators, which complicates

A. Key setup

One can choose between a 128-bit key or 256-bit key.

256-bit key: The key is stored in eight 32 bit words key[0], key[1], key[2], key[3], key[4], key[5], key[6], and key[7].

128-bit key: The first four 32 bit words, i.e. key[0], key[1], key[2] and key[3] are considered for storing the key.

B. Algorithm

Steps:

1. Assign the key length kl either as 128-bit or 256-bit.

2. if kl = 128 then
   kln=2, div=4
   else
   kln=3, div=8

3. if kl = 128 then consider two lsb’s of key[0] and find the decimal equivalent of these two lsb’s and store in the variable ‘in’.
   else
   if kl = 256 then consider three lsb’s of Key[0] and find the decimal equivalent of these three lsb’s and store it in a variable ‘in’.

4. ran =key[0] ^ key[1n]

5. if kl = 128 then consider two lsb’s of ran and find the decimal equivalent of that and store in the variable ‘in1’.

6. if kl = 256 then consider three lsb’s of ran and find the decimal equivalent of that and store in the variable ‘in1’.

7. check the 16th bit in ran,
   if it is 1 then
   newran=(key[1n1] + key[1n1+1mod div] ^ (key[1n1+2 mod div] + key[1n1+3 mod div]))
   else
   newran=(key[1n1] ^ key[1n1+1mod div]) + (key[1n1+2 mod div] ^ key[1n1+3 mod div])

8. The output 32-bit word is newran, which can be used to XOR with the corresponding word in the plain text.

9. Advance all the keys as
   key[i] = key[i] * key[i] + key[i] >> 20

10. go to step 3

IV. RANDOMNESS TESTS

The analysis of MAJE4 is done mainly using the tests listed below [3]. These tests are commonly used for determining whether the binary sequences possess some specific characteristics that a truly random sequence is likely to exhibit.

1. Frequency Test
2. Serial Test
3. Poker Test
4. Runs Test
5. Autocorrelation Test

Frequency Test (monobit test): To test whether the number of 0’s and 1’s in the sequences are approximately the same, as would be expected for a random sequence.
Serial test (2 bit test): To determine whether the number of 00, 01, 10 and 11 as subsequences of s are approximately the same, as would be expected for a random sequence.

Poker test: Let m be a +ve integer. Divide the sequence into $n / m$ non-overlapping parts of length m. To test whether the number of each sequence of length m are approximately the same, as expected for a random sequence.

Runs test: To determine whether the number of runs of various lengths in the sequence is as expected in the random sequence.

Auto-correlation test: To check whether correlation between the sequence and the shifted version of it is approximately 0 when the number of shifts is not divisible by the period as expected for a random sequence.

The frequency test is for uniformity and the other tests are for independence.

V. RESULTS

Here Frequency, Serial, Poker and Runs tests are analysed using the Chi-square table and Autocorrelation test is analysed using Normal table, as specified for each randomness tests. The fast stream cipher MAJE4 successfully passes all the five statistical tests for every run. Tables 1, 2 and 3 show the results of the specified tests.

Table I: Statistical Analysis Using Autocorrelation Test

<table>
<thead>
<tr>
<th>Number of random numbers generated</th>
<th>Total no. of bits produced</th>
<th>128-bit key</th>
<th>256-bit key</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Autocorrelation test</td>
<td>Autocorrelation test</td>
</tr>
<tr>
<td>300</td>
<td>9600</td>
<td>2.0855</td>
<td>1.5007</td>
</tr>
<tr>
<td>500</td>
<td>16000</td>
<td>2.2158</td>
<td>1.8581</td>
</tr>
<tr>
<td>800</td>
<td>25600</td>
<td>2.0762</td>
<td>2.3439</td>
</tr>
<tr>
<td>1000</td>
<td>32000</td>
<td>2.4944</td>
<td>1.6045</td>
</tr>
<tr>
<td>2000</td>
<td>64000</td>
<td>1.6368</td>
<td>1.4902</td>
</tr>
</tbody>
</table>

Table II: Statistical Analysis Using Frequency, Serial, Poker And Runs Tests With 128 Bit Key

<table>
<thead>
<tr>
<th>Number of random numbers generated</th>
<th>Total no. of bits produced</th>
<th>128-bit key</th>
<th>256-bit key</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Frequency test</td>
<td>Serial test</td>
</tr>
<tr>
<td>500</td>
<td>16000</td>
<td>1.3690</td>
<td>3.6252</td>
</tr>
<tr>
<td>1000</td>
<td>32000</td>
<td>1.5961</td>
<td>3.5682</td>
</tr>
<tr>
<td>1500</td>
<td>48000</td>
<td>1.4083</td>
<td>3.4354</td>
</tr>
<tr>
<td>4000</td>
<td>128000</td>
<td>0.0632</td>
<td>1.2458</td>
</tr>
<tr>
<td>6000</td>
<td>192000</td>
<td>0.2475</td>
<td>1.8224</td>
</tr>
<tr>
<td>8000</td>
<td>256000</td>
<td>0.4100</td>
<td>2.9614</td>
</tr>
<tr>
<td>10000</td>
<td>320000</td>
<td>0.0903</td>
<td>3.7286</td>
</tr>
</tbody>
</table>

Table III: Statistical Analysis Using Frequency, Serial, Poker And Runs Tests With 256 Bit Key

<table>
<thead>
<tr>
<th>Number of random numbers generated</th>
<th>Total no. of bits produced</th>
<th>Statistical Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>256-bit key</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Frequency test</td>
</tr>
<tr>
<td>500</td>
<td>16000</td>
<td>0.3610</td>
</tr>
<tr>
<td>1000</td>
<td>32000</td>
<td>0.8820</td>
</tr>
<tr>
<td>1500</td>
<td>48000</td>
<td>0.3100</td>
</tr>
<tr>
<td>4000</td>
<td>128000</td>
<td>0.0031</td>
</tr>
<tr>
<td>6000</td>
<td>192000</td>
<td>0.0316</td>
</tr>
<tr>
<td>8000</td>
<td>256000</td>
<td>0.0082</td>
</tr>
<tr>
<td>10000</td>
<td>320000</td>
<td>0.0630</td>
</tr>
</tbody>
</table>

VI. PERFORMANCE EVALUATION

The summary of performance evaluation is presented here by comparing with JEROBOAM stream cipher and shown in Table 4.

A. Timing Analysis

From the timing analysis it can be noted that when we compare JEROBOAM 128-bit and MAJE4 128-bit, MAJE4 128-bit is almost 9 times faster as shown in Fig.2. Also the memory requirement is less in MAJE4. The evaluation is done using Pentium IV Processor, Linux Operating System and C compiler.

Table IV: Timing Analysis

<table>
<thead>
<tr>
<th>PRNGs used</th>
<th>Key length</th>
<th>No. of random numbers generated</th>
<th>No. of random bits per each random number</th>
<th>Total no. of bits produced (speed Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>JERO-BOAM</td>
<td>128-bit</td>
<td>26,80,000</td>
<td>16</td>
<td>40.89</td>
</tr>
<tr>
<td>MAJE4</td>
<td>128-bit</td>
<td>1,15,39,399</td>
<td>32</td>
<td>352.15</td>
</tr>
<tr>
<td>MAJE4</td>
<td>Variable 128-bit</td>
<td>58,34,000</td>
<td>32</td>
<td>178.03</td>
</tr>
<tr>
<td>MAJE4</td>
<td>Variable 256-bit</td>
<td>43,99,999</td>
<td>32</td>
<td>134.27</td>
</tr>
</tbody>
</table>

Fig. 2. Comparison of Number of Random Bits Produced per Second (in Mbps)
B. Memory Requirement

On comparing the memory required for executable files of JEROBOAM 128-bit and MAJE4 128-bit, MAJE4 was found consuming lesser space compared to JEROBOAM. The memory size required for optimised code for JEROBOAM is 6341 bytes, for MAJE4 128-bit is 5435 bytes and for MAJE4 128-bit or 256-bit is 5678 bytes.

VII. CONCLUSIONS

From the results of analysis and performance evaluation it can be concluded that MAJE4 is a reliable generator, which is much faster than JEROBOAM. All the five statistical tests are passed by this generator for all the random streams produced. Since the memory requirement is less, it can be used for devices with restricted memory for security purposes. Since it is faster and produce independent bits, this stream cipher can be used for applications that require encryption / decryption of a stream of data sent through the Internet.

REFERENCES