Research of the Methods for Improving Performance for Cryptographically Strong BBS Pseudorandom Bit Sequences Generators

Andrii Malohlovets[1], Volodymyr Maxymovych[2]

[1]Department of Information Technology Security, Lviv Polytechnic National University, UKRAINE, Lviv, S. Bandery street 12, E-mail: maloglovets@gmail.com

[2]Department of Information Technology Security, Lviv Polytechnic National University, UKRAINE, Lviv, S. Bandery street 12, E-mail: volodymyr.maksymovych@gmail.com

Abstract – The aim of this work is the research of methods for improving the performance on cryptographically strong BBS generators, their practical realization and analysis of their efficiency.

Key words – BBS, improving performance, computational complexity, efficiency of parallelizing, K-stream generator

I. Introduction

Blum Blum Shub (B.B.S.) is a pseudorandom number generator proposed in 1986 by Lenore Blum, Manuel Blum and Michael Shub that is derived from Michael O. Rabin's oblivious transfer mapping. Blum Blum Shub can be represented by the formula:

$$x_{n+1} = x_n^2 \ mod \ M, \qquad (1)$$

where M = pq is the product of two large primes p and q. At each step of the algorithm, some output is derived from x_{n+1}.

The seed x0 should be an integer that is co-prime to M (i.e. p and q are not factors of x0) and not 1 or 0.

The Blum Blum Shub generator has the possibility of calculating any x_i value directly (via Euler's theorem):

$$x_n = x_0^{2^n \bmod ((p-1)(q-1))} \bmod M. \qquad (2)$$

There is a proof reducing its security to the computational difficulty of solving the quadratic residuosity problem. When the primes are chosen appropriately, and O(log log M) lower-order bits of each xn are output, then in the limit as M grows fast, distinguishing the output bits from random should be at least as difficult as solving the Quadratic residuosity problem modulo M.

II. Theoretical Improvement of BBS Generator.

For such type of formula:

$$x^y \ mod \ N, – \qquad (3)$$

computational complexity is

$$O(M(n)*[log_2 y]), \qquad (4)$$

where $M(n)$ – is the complexity of the chosen multiplication algorithm.

Computational complexity of one step (formula 1) is

$$O(M(n)) \qquad (5)$$

Computational complexity for formula 3 is

$$O(M(n)*([log_2 M]+[log_2 N])). \qquad (6)$$

If

$$G = [log_2 M]+[log_2 N]. \qquad (7)$$

Expected results of improved two-stream generator:

$$y = 2 * N, \qquad (8)$$

in comparison to original, one-stream:

$$y = N + G. \qquad (9)$$

Generator diagram is shown in Fig. 1.

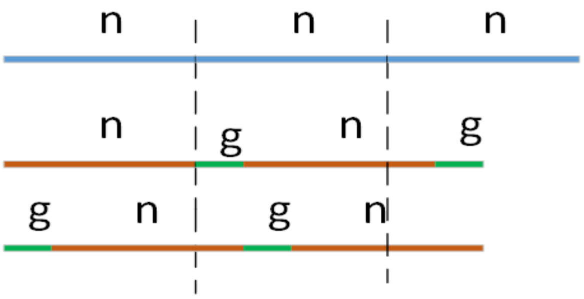

Fig. 1. Generator diagram

Graph ratio of the number generated bits to N is shown in Fig. 2.

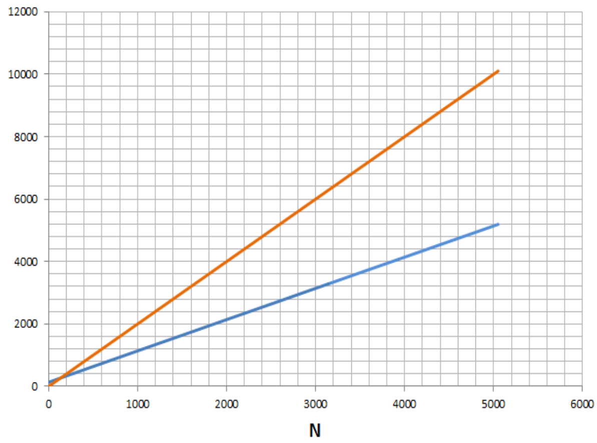

Fig. 2. Graph ratio of the number generated bits to N

Computational complexity for three-stream generator is:

$$O(M(n)*([log_2 M]+[log_2 2*N])). \qquad (10)$$

If

$$G = [log_2 M]+[log_2 2*N] \qquad (11)$$

Expected results of improved three-stream generator:

$$y = 3 * N, \qquad (12)$$

in comparison to original generator:

$$y = N – 3 * G. \qquad (13)$$

Computational complexity for parallelizing into K streams is

$$O(M(n)*([log_2 M]+[log_2 (K-1)*N])). \qquad (14)$$

If

$$G = [log_2 M]+[log_2 (K-1)*N] \qquad (15)$$

Expected results of improved generator

$$y = K * N, \qquad (16)$$

in comparison to original

$y = N + G *$ [redacted]. (17)

Efficiency of parallelizing into K streams is

[redacted]. (18)

III. Results of Improved BBS Generator

Statistical portraits of the generated bit sequences for original and two-stream generators are shown in Fig. 3 and 4 appropriately.

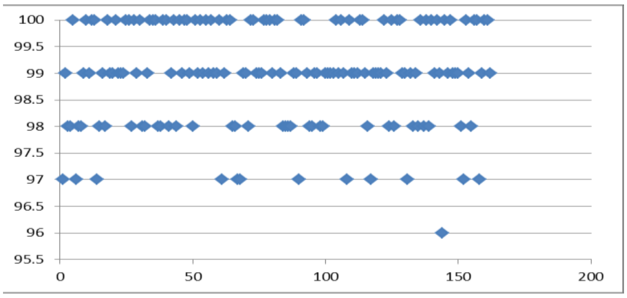

Fig. 3. Statistical portraits of the generated bit sequences for original generators

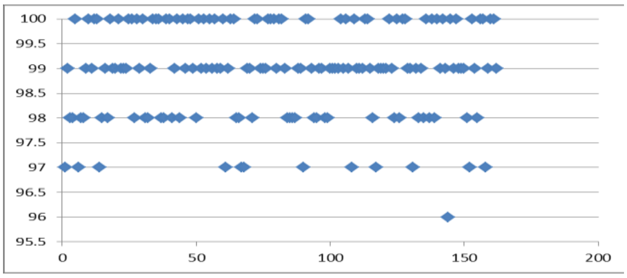

Fig. 4. Statistical portraits of the generated bit sequences for two-stream generators

Time elapsed for generating 1.5 * 10^6 bits by original, two-stream and three-stream generators BBS is shown in Fig. 5.

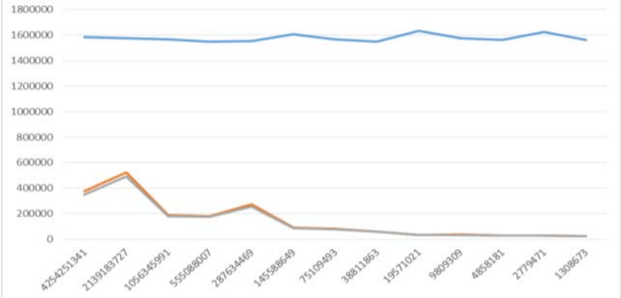

Fig. 5. Time elapsed for generating 1.5 * 10^6 bits by original, two-stream and three-stream generators BBS

Time elapsed for generating 1.5 * 10^6 bits by two-stream generator BBS with different sizes of N is shown in Fig. 6.

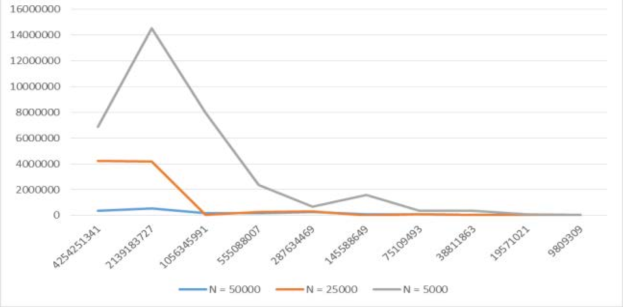

Fig. 6. Time elapsed for generating 1.5 * 10^6 bits by two-stream generator BBS with different sizes of N.

Time elapsed for generating 1.5 * 10^6 bits by three-stream generator BBS with different sizes of N is shown in Fig. 7.

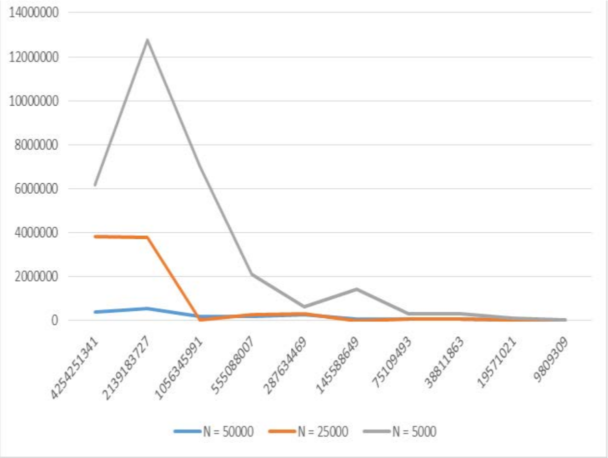

Fig. 7. Time elapsed for generating 1.5 * 10^6 bits by three-stream generator BBS with different sizes of N.

Conclusion

Theoretical calculation improvement of the BBS generator is expected to provide such results:

- The efficiency of two-stream generator comparing to the original one is 1.946 times higher.
- Recommended size of N should be 40 times higher than the number of key bits.
- Efficiency of parallelizing tends to maximum value, number of streams, if N >> [$\log_2$ M].

Results of practical realization are summed up as follows:

- The efficiency of two-stream generator comparing to the original one is 4.198 times higher.
- The efficiency of three-stream generator comparing to the original one is 4.546 times higher.
- Efficiency of parallelizing depends on how low is the result of calculating the formula:

$$2^N \bmod ((p-1)(q-1)). \qquad (19)$$

References

[1] Blum, Lenore; Blum, Manuel; Shub, Mike (1982). "Comparison of Two Pseudo-Random Number Generators". Advances in Cryptology: Proceedings of CRYPTO '82. Plenum: 61–78.

[2] Lenore Blum, Manuel Blum, and Michael Shub. «A Simple Unpredictable Pseudo-Random Number Generator», SIAM Journal on Computing, volume 15, pages 364—383, May 1986.

[3] Lenore Blum, Manuel Blum, and Michael Shub. «A Simple Unpredictable Pseudo-Random Number Generator», SIAM Journal on Computing, volume 15, pages 364—383, May 1986.

[4] Andrew Rukhin, et. al. «A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications», NIST ITL Special Publication 800-22 (Apr 2010)