Create account Log in

Article Talk Read Edit View history Search

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Čeština
Deutsch
Español
فارسی
Français
Հայերեն
Italiano
日本語
Polski
Português
Русский
Slovenščina
Svenska
ไทย
Türkçe
Edit links

Blum Blum Shub

From Wikipedia, the free encyclopedia

This article has multiple issues. Please help **improve it** or discuss these issues on the **talk page**. [hide]

- This article includes a list of references, but **its sources remain unclear** because it has **insufficient inline citations**. *(September 2013)*
- This article **relies too much on references to primary sources**. *(September 2013)*

Blum Blum Shum (**B.B.S.**) is a pseudorandom number generator proposed in 1986 by Lenore Blum, Manuel Blum and Michael Shub [1] that is derived from Michael O. Rabin's oblivious transfer mapping.

Blum Blum Shum takes the form

$$x_{n+1} = x_n^2 \bmod M.$$

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 output is commonly either the bit parity of x_{n+1} or one or more of the least significant bits of x_{n+1}.

The seed x_0 should be an integer that is co-prime to *M* (i.e. *p* and *q* are not factors of x_0) and not 1 or 0.

The two primes, *p* and *q*, should both be congruent to 3 (mod 4) (this guarantees that each quadratic residue has one square root which is also a quadratic residue) and gcd(*φ*(*p* − 1), *φ*(*q* − 1)) should be small (this makes the cycle length large).

An interesting characteristic of the Blum Blum Shub generator is the possibility to calculate any x_i value directly (via Euler's Theorem):

$$x_i = \left(x_0^{2^i \bmod \lambda(M)}\right) \bmod M.$$

where λ is the Carmichael function. (Here we have $\lambda(M) = \lambda(p \cdot q) = \operatorname{lcm}(p-1, q-1)$.

Contents [hide]

Security [edit]

There is a proof reducing its security to the computational difficulty of solving the Quadratic residuosity problem.[1] When the primes are chosen appropriately, and O(log log *M*) lower-order bits of each x_n are output, then in the limit as *M* grows large, distinguishing the output bits from random should be at least as difficult as solving the Quadratic residuosity problem modulo *M*.

Example [edit]

Let $p = 11$, $q = 19$ and $s = 3$ (where s is the seed). We can expect to get a large cycle length for those small numbers, because $\gcd(\varphi(p-1), \varphi(q-1)) = 2$. The generator starts to evaluate x_0 by using $x_{-1} = s$ and creates the sequence x_0, x_1, x_2, $\ldots x_5$ = 9, 81, 82, 36, 42, 92. The following table shows the output (in bits) for the different bit selection methods used to determine the output.

Even parity bit	Odd parity bit	Least significant bit
0 1 1 0 1 0	1 0 0 1 0 1	1 1 0 0 0 0

References [edit]

1. ^ *a* *b* Blum, Lenore; Blum, Manuel; Shub, Mike (1 May 1986). "A Simple Unpredictable Pseudo-Random Number

Generator". *SIAM Journal on Computing* **15** (2): 364–383. doi:10.1137/0215025.

General

- Blum, Lenore; Blum, Manuel; Shub, Mike (1982). "Comparison of Two Pseudo-Random Number Generators". Advances in Cryptology: Proceedings of CRYPTO '82. Plenum. pp. 61–78.
- Geisler, Martin; Krøigård, Mikkel; Danielsen, Andreas (December 2004). "About Random Bits". available as PDF and Gzipped Postscript

External links [edit]

- GMPBBS , a GPL'ed GMP-based implementation of Blum Blum Shub in C by Maria Morisot with implementations in Java and PHP also.
- An implementation in Java
- Randomness tests

Categories: Pseudorandom number generators | Cryptographically secure pseudorandom number generators