Cryptographic Secure Pseudo-Random Bits Generation : The Blum-Blum-Shub Generator

Pascal Junod

August 1999

Contents

1 Introduction **3**

2 Concept of Pseudo-Random Bit Generator **4**

3 The Blum-Blum-Shub Generator **7**
3.1 Some number-theoretic preliminaries 7
3.2 Definition of the Blum-Blum-Shub PRBG 14

4 Security of the Blum-Blum-Shub Generator **15**
4.1 Introduction 15
4.2 The proof of security 15

List of Figures

1 Statistical experiment for the cryptographic security of a PRBG 5
2 Statistical experiment for the next-bit test 6
3 Description of the algorithm $B(n, x)$ 18

1 Introduction

Random numbers are critical in every aspect of cryptography. We need such numbers to encrypt e-mails, to digitally sign documents, for electronic payment systems, and so on.

Unfortunately, true random numbers are very difficult to generate, especially on computers that are typically designed to be deterministic. This brings us to the concept of *pseudo-random* numbers, which are numbers generated from some random internal values, and that are very hard for an observer to distinguish from true random numbers.

It is important to see the difference between the meaning of pseudo-random numbers in normal programming contexts, like simulation, e.g., where these numbers merely need to be reasonably random-looking and have good statistical properties (see [4]), and in the context of cryptography, where they *must* be indistinguishable from real random numbers, even to observers with huge amount of computational resources.

In the context of cryptography, a random number is a number that cannot be predicted by an observer before it is generated. Typically, if the number is to be in the range $[0..n-1]$, an observer cannot predict that number with probability "slightly" better than $1/n$. Or, we will see that the following is equivalent, if m random numbers are generated in a row, an observer given any $m-1$ of them still cannot predict the m'th with a probability significantly greater than $1/n$.

In this work, we present first the notion of *cryptographic secure pseudo-random bit generators (PRBG)* in a formal way by using two different definitions. Then a theorem of Yao proving the equivalence of these two definitions is treated. In a second part, *the Blum-Blum-Shub generator*, a very simple and provably secure PRBG, is presented, with all the mathematical background needed to understand it. In the third part, the proof of its security is treated in details.

2 Concept of Pseudo-Random Bit Generator

We give first an informal definition of a Pseudo-Random Bit Generator:

Definition 1 (Informal Definition)
A Pseudo-Random Bit Generator (PRBG) *is a deterministic algorithm which, given a truly-random binary sequence of length n, outputs a binary sequence of length $l(n) > n$ which appears to be random, with $l()$ being a polynomial. The input to the PRBG is called the* seed, *and the output is called a* pseudo-random bit sequence.

We now have to specify what "appears to be random" means. We give two formal, different definitions of this fact which are equivalent.

In a few words, the first definition says that a PRBG is said to pass all *poly-time statistical tests*, and therefore can be considered as a cryptographic secure PRBG, if no poly-time algorithm can distinguish between an output sequence of the generator and a truly random sequence with probability significantly greater than 1/2.

Definition 2 (Cryptographic secure PRBG, [7])
Let $g : \{0,1\}^n \longrightarrow \{0,1\}^{l(n)}$ be an efficient (computable in polynomial time) function ensemble, $l()$ being a polynomial with $l(n) > n$. Let X and Z be random variables uniformly distributed respectively on $\{0,1\}^n$ and on $\{0,1\}^{l(n)}$. Then g is a cryptographic secure PRBG, *if for all adversaries A running in polynomial time the success probability (or distinguishing probability)*

$$|P_X[A(g(X)) = 1] - P_Z[A(Z) = 1]| < \frac{1}{p(n)} \quad \forall p$$

where p is a polynomial.

The Figure 1 gives an illustration of the statistical experiment suggested by this definition. A truly random sequence or the output of the generator initialized with a random seed are given to the adversary, each with a probability of 1/2. Then the adversary decides in polynomial time which sequence it was.

We give now another definition, which says that a PRBG is said to pass the *next bit test* if there is no poly-time algorithm which, on input of the first $r \leq l(n) - 1$ bits of the sequence of an output sequence s, can predict the $(r+1)$-st bit of s with probability significantly greater than 1/2. In the following, $g(.)_{\{1,\ldots,I-1\}}$ is the notation for the first $I-1$ bits of the generator's output $g(.)$, and $g(.)_I$ represents the I-th bit of this output.

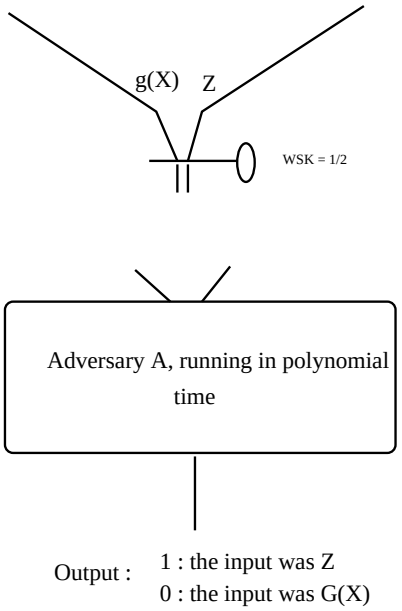

Figure 1: Statistical experiment for the cryptographic security of a PRBG

Definition 3 (Next bit unpredictable, [2])
Let $g : \{0,1\}^n \longrightarrow \{0,1\}^{l(n)}$ *be an efficient (computable in polynomial time) function ensemble, with* $l()$ *being a polynomial with* $l(n) > n$. *Let* X *and* I *be random variables uniformly distributed respectively on* $\{0,1\}^n$ *and on* $\{1,...,l(n)\}$. *Then* g *is a* next bit unpredictable PRBG, *if for all adversaries* A *running in polynomial time the success probability (prediction probability) of* A *for* g

$$P[A(I, g(X)_{\{1,...,I-1\}}) = g(X)_I] < \frac{1}{p(n)} \quad \forall p$$

where p *is a polynomial.*

Figure 2 explains how the next-bit test works: first a seed and a number $i-1$ of bits are randomly chosen; the adversary must then predict the i-th bit with the first $i-1$ bits as input in polynomial time.

The following theorem states that the two above definitions are equivalent :

Theorem 1 (Yao, [7])
A PRBG passes the next-bit test if and only if it passes all poly-time statistical tests.

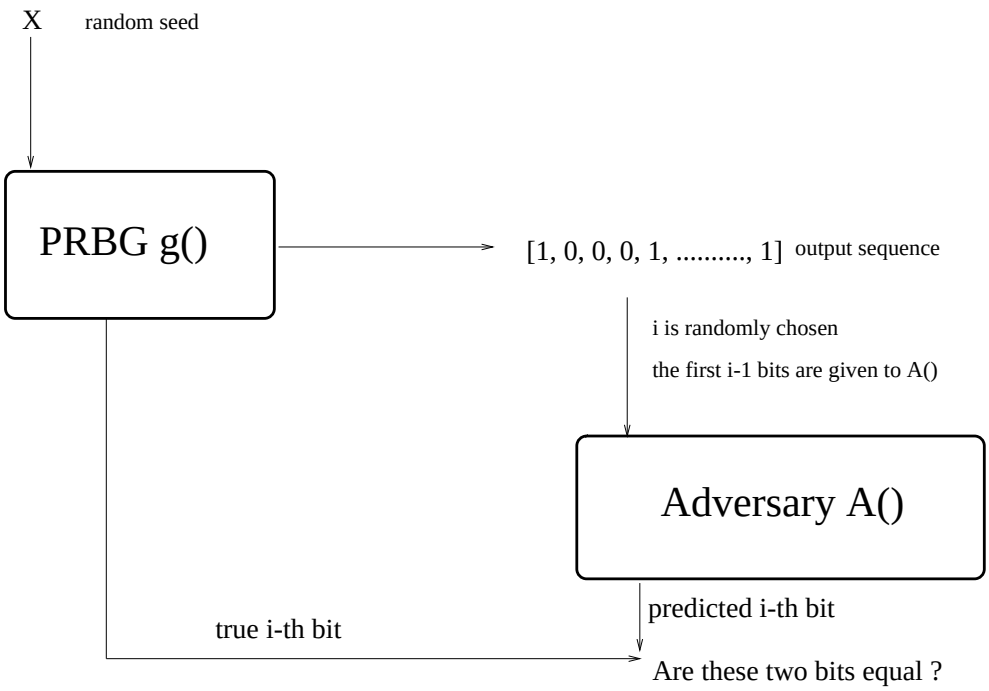

Figure 2: Statistical experiment for the next-bit test

Sketch of the proof :

Assume first that the PRBG does not pass the next-bit test. It is clear that such a poly-time algorithm is a statistical test that allows to distinguish the output string from a truly-random string : guess the last bit from the previous ones and check whether it is the same as the actual last bit.

On the other hand, assume that the PRBG passes the next-bit test, i.e., for every poly-bounded machine, every bit looks random, *given all the previous bits* (i.e., the guessing probability is close to $1/2$). This clearly implies that the entire string looks random (i.e., is chosen according to a uniform distribution).

□

3 The Blum-Blum-Shub Generator

In this section, we present the Blum-Blum-Shub PRBG generator, which was described in [1]. We need first some number theory background to understand its foundations. [6] was used as a reference book.

3.1 Some number-theoretic preliminaries

First of all, we recall that the Chinese Remainder Theorem (CRT) specifies a one-to-one transformation between elements a of $\mathbb{Z}_m$, where $m = m_1 \cdot m_2 \cdot ... \cdot m_k$ and lists $(r_1, r_2, ...r_k)$ of residues, when the moduli m_1, m_2, ..., m_k are pairwise relatively prime. We shall refer to the list $(r_1, r_2, ..., r_k)$ as the *CRT-transform* of a. The two main interesting properties of this transform are the following : first, the CRT-transform of the product of two numbers a_1 and a_2 in $\mathbb{Z}_m$ is the component-wise product of the CRT-transforms of a_1 and a_2; second, a is an invertible element in $\mathbb{Z}_m$ if and only if the moduli r_i are invertible elements of $\mathbb{Z}_{m_i}$ for all $1 \leq i \leq k$ respectively.

We define first the concepts of *quadratic residues* and of *Legendre symbol*:

Definition 4 (Quadratic Residues)
Let $n \in \mathbb{N}$. Then $a \in \mathbb{Z}_n^*$ is called a quadratic residue *modulo n if there exists $b \in \mathbb{Z}_n^*$ such that*

$$a \equiv b^2 \pmod{n}$$

The set of quadratic residues modulo n is denoted by QR_n. Furthermore,

$$QNR_n := \mathbb{Z}_n^* \setminus QR_n$$

is called the set of quadratic non-residues.

Example 1
For $\mathbb{Z}_{23}^*$, we have

$$QR_{23} = \{1, 2, 3, 4, 6, 8, 9, 12, 13, 16, 18\}$$

and

$$QNR_{23} = \{5, 7, 10, 11, 14, 15, 17, 19, 20, 21, 22\}$$

Definition 5 (Legendre symbol)
Let p be an odd prime. For $a \in \mathbb{Z}_p^*$, the Legendre symbol $\left(\frac{a}{p}\right)$ is defined as

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & p|a \\ 1 & a \in QR_p \\ -1 & a \notin QR_p \end{cases}$$

The following theorem shows how to compute the Legendre symbol of an element $a \in \mathbb{Z}_p^*$:

Theorem 2
Let p be an odd prime, and let $a \in \mathbb{Z}_p^*$. Then

$$\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod{p}$$

Proof :

Let $a \in QR_p$, i.e., $a = b^2$ in $\mathbb{Z}_p^*$ for some $b \in \mathbb{Z}_p^*$. Then

$$a^{\frac{p-1}{2}} \equiv (b^2)^{\frac{p-1}{2}} \equiv b^{p-1} \equiv 1 \pmod{p}$$

because of Fermat's little Theorem.
Let $a \in QNR_p$. Let g be a generator of $\mathbb{Z}_p^*$ (a cyclic group of order $p-1$). Then $a = g^t$ for some *odd* $t = 2s+1$ (otherwise, $a = g^t = g^{2s} = (g^s)^2$), and

$$a^{\frac{p-1}{2}} \equiv (g^t)^{\frac{p-1}{2}} \equiv (g^{2s})^{\frac{p-1}{2}} \cdot g^{\frac{p-1}{2}} \equiv g^{\frac{p-1}{2}} \pmod{p}$$

Now $(g^{\frac{p-1}{2}})^2 = 1$, hence $g^{\frac{p-1}{2}} \in \{-1, 1\}$. Because g is a generator of $\mathbb{Z}_p^*$, the order of g is equal to $p-1$, and $g^{\frac{p-1}{2}} = -1$.

□

Theorem 3
Let p be an odd prime. Then

$$|QR_p| = |QNR_p| = (p-1)/2$$

Proof :

Let g be a generator of $\mathbb{Z}_p^*$. By the proof of Theorem 2, $g^t \in QR_p$ holds if and only if $t \in \{0, 1, 2, ..., p-2\}$ is even, which is the case for half of the t's.

□

Another interesting property of the Legendre symbol is the following:

Theorem 4
Let p be an odd prime, a and b integers. Then

$$\left(\frac{a}{p}\right) \cdot \left(\frac{b}{p}\right) = \left(\frac{ab}{p}\right)$$

Proof :

This property is an immediate consequence of Theorem 2:

$$a^{\frac{p-1}{2}} \cdot b^{\frac{p-1}{2}} = (a \cdot b)^{\frac{p-1}{2}} \pmod{p}$$

□

We now define the *Jacobi symbol*, which is in fact the analogous of the Legendre symbol for composite moduli:

Definition 6 (Jacobi symbol)
Let n be an odd integer with prime factorization

$$n = \prod_i p_i^{e_i}$$

Let $a \in \mathbb{Z}_n^*$. Then the Jacobi symbol $\left(\frac{a}{n}\right)$ *is defined by*

$$\left(\frac{a}{n}\right) := \prod_i \left(\frac{a}{p_i}\right)^{e_i} \tag{1}$$

The Jacobi symbol has the following multiplicative property:

Theorem 5
Let n be an odd integer, and let a and b be integers. Then

$$\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right) \cdot \left(\frac{b}{n}\right)$$

Proof :

Using Definition 6 and Theorem 4, we have:

$$\left(\frac{ab}{n}\right) = \prod_i \left(\frac{ab}{p_i}\right)^{e_i} = \prod_i \left(\frac{a}{p_i}\right)^{e_i} \cdot \prod_i \left(\frac{b}{p_i}\right)^{e_i} = \left(\frac{a}{n}\right) \cdot \left(\frac{b}{n}\right)$$

which concludes the proof.

□

We are now interested in the number of square roots of each quadratic residue in a general multiplicative group. The three next lemmas are useful for proving Theorem 6.

Lemma 1
Let p be a prime and α a non-zero element of $\mathbb{Z}_p^*$; then we have

$$-\alpha = \alpha \Longleftrightarrow p = 2$$

Proof :

Let $R_p(x)$ be the reduction of x modulo p, often referred to as "$x \pmod p$", for integers x. We have:

$$\alpha = -\alpha \Rightarrow 2\alpha = 0 \Rightarrow R_p(2\alpha) = 0 \Rightarrow p \mid 2\alpha$$

For $1 \leq \alpha < p$, we have $p \mid 2$ and finally $p = 2$.
In the other way, we have

$$p = 2 \Rightarrow p \mid 2\alpha \Rightarrow R_p(2\alpha) = 0 \Rightarrow \alpha = -\alpha$$

□

Lemma 2
For p, an odd prime and for α, β, non-zero elements of $\mathbb{Z}_p^*$, we have

$$\alpha^2 = \beta^2 \Longleftrightarrow \alpha = \beta \vee \alpha = -\beta$$

Proof :

We have

$$\alpha^2 = \beta^2 \Leftrightarrow \alpha^2 - \beta^2 = 0 \Leftrightarrow (\alpha - \beta)(\alpha + \beta) = 0 \Leftrightarrow \begin{cases} \alpha = \beta \\ \alpha = -\beta \\ \alpha = \beta \wedge \alpha = -\beta \end{cases}$$

Suppose that $\alpha = \beta$ is true. Then, from $\alpha = -\beta$ it follows that also $\alpha = -\alpha$ is true, which is impossible for an odd prime because of Lemma 1. Thus, the two equations $\alpha = \beta$ and $\alpha = -\beta$ cannot hold simultaneously, from which we can conclude that $\alpha^2 = \beta^2 \Leftrightarrow \alpha = \beta \vee \alpha = -\beta$.

□

This lemma implies that each quadratic residue from $\mathbb{Z}_p^*$, p being an odd prime, has exactly 2 square roots α_1 and α_2 such that $1 \leq \alpha_1 \leq (p-1)/2$ and $(p+1)/2 \leq \alpha_2 \leq (p-1)$.

Lemma 3
Let $n = p_1 \cdot p_2 \cdot ... \cdot p_k$, where p_1, ..., p_k are distinct odd primes and $k \geq 2$. An element α of $\mathbb{Z}_n^*$ is a quadratic residue modulo n if and only if each component of its CRT transform with respect to the moduli $p_1, ..., p_k$ is a quadratic residue of $\mathbb{Z}_{p_i}^*$, where p_i is the modulus corresponding to that component.

Proof :

The CRT transform of $\alpha \in \mathbb{Z}_n^*$ with respect to $p_1, ..., p_k$ is

$$(\alpha \mod p_1, ..., \alpha \mod p_k)$$

Thus, because of the CRT characterization of the multiplication, we must prove that for $1 \leq i \leq k$:

$$\alpha \in QR_n \Longleftrightarrow \alpha \mod p_i \in QR_{p_i}$$

Proof of "$\Longrightarrow$" :
Let $\alpha = \beta^2 \pmod{n}$, then $R_n(\alpha) = R_n(\beta^2)$. Taking $R_{p_i}(.)$ on both sides, we get

$$R_{p_i}(\alpha) = R_{p_i}(R_n(\alpha)) = R_{p_i}(R_n(\beta^2)) = R_{p_i}(\beta^2) = R_{p_i}(R_{p_i}(\beta)^2)$$

We conclude that $R_{p_i}(\beta)$ is a square root of $R_{p_i}(\alpha)$ in $\mathbb{Z}_{p_i}^*$, so $\alpha \pmod{p_i}$ is a quadratic residue in $\mathbb{Z}_{p_i}^*$.
Proof of "$\Longleftarrow$" :
By assumption, we can write the CRT transform of α as $(\alpha_1^2, \alpha_2^2, ..., \alpha_k^2)$. By the properties of the CRT transform, this is the square of an element β with CRT transform $(\alpha_1, \alpha_2, ..., \alpha_k)$. Hence, $\alpha = \beta^2 \pmod{n}$ is a quadratic residue in $\mathbb{Z}_n^*$.

□

Theorem 6
Let n be an odd integer. If $a \in QR_n$, then the number of distinct square roots of a is exactly

$$2^k$$

where k is the number of distinct prime factors of n.

Proof :

From Lemma 3 we conclude that $(\pm\alpha_1, ..., \pm\alpha_k)$ are square roots of a quadratic residue in $\mathbb{Z}_n^*$. So, a quadratic residue has at least 2^k square roots. But a quadratic residue in $\mathbb{Z}_{p_i}^*$ has only two square roots (see Lemma 2), so there are exactly 2^k square roots.

□

Theorem 7
Let $n = p \cdot q$ be the product of two distinct odd primes. Exactly half the elements of $\mathbb{Z}_n^*$ have Jacobi symbol $+1$, the other half have Jacobi symbol -1. We denote these two sets respectively by $\mathbb{Z}_n^*(+1)$ and $\mathbb{Z}_n^*(-1)$. None of the elements of $\mathbb{Z}_n^*(-1)$ and exactly half of the elements of $\mathbb{Z}_n^*(+1)$ are quadratic residues.

Proof :

By Theorem 3, we know that exactly one half of the elements of $\mathbb{Z}_p^*$ and of $\mathbb{Z}_q^*$ are quadratic residues, respectively. Using Definition 6, we see that there is four possibilities, namely $(+1)\cdot(+1)$, $(+1)\cdot(-1)$, $(-1)\cdot(+1)$ and $(-1)\cdot(-1)$, to build the product for computing the Jacobi symbol of the elements of $\mathbb{Z}_n^*$. Only the first possibility gives a quadratic residue modulo n because of Lemma 3. The last possibility furnishes a quadratic non-residue whose Jacobi symbol is equal to 1, and the two others a quadratic non-residue with a Jacobi symbol equal to -1.

□

We now define the Blum primes, and we give the essential properties of these numbers which are interesting to understand the design of the Blum-Blum-Shub PRBG:

Definition 7
A prime number p with $p \equiv 3 \pmod 4$ is called a Blum prime number.

An important property of Blum primes is the following:

Theorem 8
Let p be an odd prime number. Then

$$-1 \in QNR_p \iff \quad p \text{ is a Blum prime}$$

Proof :

By Theorem 2, we can write

$$\left(\frac{-1}{p}\right) \equiv (-1)^{\frac{p-1}{2}} \pmod p$$

p being odd by assumption, it must be congruent to 1 or to 3 modulo 4. But $\frac{p-1}{2}$ is odd if and only if $p \equiv 3 \pmod 4$, which concludes the proof.

□

Theorem 9
Let $n = p \cdot q$ be the product of two Blum primes. Let $a \in QR_n$. Then a has exactly four square roots, exactly one of which is in QR_n itself.

Proof :

The first statement follows from Theorem 6.
By Theorem 8, the element $R_p(a) \in QR_p$ has two square roots in $\mathbb{Z}_p^*$, namely, if $a = g^{2s}$, $b = g^s$ and $-b$. Now,

$$(-b)^{\frac{p-1}{2}} = (-1)^{\frac{p-1}{2}} \cdot b^{\frac{p-1}{2}}$$

hence

$$\left(\frac{b}{p}\right) \neq \left(\frac{-b}{p}\right)$$

because p is a Blum prime (see Theorem 8). Hence exactly one of $\{b, -b\}$ is in QR_p. The same is true modulo q, and the four roots modulo n are the four "Chinese combinations" of the roots modulo p and q.
Clearly,

$$a \in QR_n \Leftrightarrow R_p(a) \in QR_p \wedge R_q(a) \in QR_q$$

Hence the statement follows by Lemma 3.

□

Definition 8
Let $n = p \cdot q$ be the product of two Blum primes. Let $a \in QR_n$. Then $\sqrt{a}$ denotes the square root of a such that $\sqrt{a} \in QR_n$.

Theorem 10
Let $n = p \cdot q$ be the product of two Blum primes. The function

$$\begin{array}{rccl} f: & QR_n & \longrightarrow & QR_n \\ & x & \longmapsto & x^2 \pmod{n} \end{array}$$

is a permutation.

Proof :

From Theorem 9 we know that each quadratic residue has *exactly* one square root which is a quadratic residue, hence this function is a bijection, or more precisely, because it is defined from a set to the same set, a permutation.

□

3.2 Definition of the Blum-Blum-Shub PRBG

The Blum-Blum-Shub PRBG (described in [1]) is based on the function defined in Theorem 10.

Definition 9
The Blum-Blum-Shub PRBG is the following algorithm :

- *Generate p and q, two big Blum prime numbers.*
- $n := p \cdot q$
- *Choose $s \in_R [1, n-1]$, the random seed.*
- $x_0 := s^2 \pmod{n}$
- *The sequence is defined as $x_i := x_{i-1}^2 \pmod{n}$ and $z_i := parity(x_i)$.*
- *The output is $z_1, z_2, z_3, ...$*

where $parity(x_i)$ is defined as $R_2(x_i)$.

Example 2
Let $n = p \cdot q = 7 \cdot 19 = 133$ and $s = 100$. Then we have $x_0 = 100^2 \pmod{133} = 25$. The sequence $x_1 = 25^2 \pmod{133} = 93$, $x_2 = 93^2 \pmod{133} = 4$, $x_3 = 4^2 \pmod{133} = 16$, $x_4 = 16^2 \pmod{133} = 123$ produces the output $1, 0, 0, 1$.

4 Security of the Blum-Blum-Shub Generator

4.1 Introduction

The cryptographic security of the Blum-Blum-Shub PRBG follows from an assumption on a number-theoretic problem, the so-called *quadratic residuosity problem*, which is defined as follows, together with the concept of *solver* for this problem (see [1]):

Definition 10 (Quadratic Residuosity Problem and Solver)
The quadratic residuosity problem *with parameters n and x is to decide for $x \in \mathbb{Z}_n^*(+1)$ whether x is a quadratic residue or not. A* solver *for the quadratic residuosity problem is a poly-time algorithm $A(n, x)$ which outputs a 1 if and only if x is a quadratic residue in $\mathbb{Z}_n^*(+1)$ and a 0 otherwise.*

The security of the Blum-Blum-Shub PRBG is based on the following assumption (see [1]):

Assumption 1
Let t be a positive integer, and n be the product of two distinct odd primes, $A(n, x)$ be a solver for the quadratic residuosity problem and $s := \lceil \log_2 n \rceil$ be the binary length of n. Then for s sufficiently large and for all but $1/s^t$ fraction of numbers n of length s, the probability that $A(n, x)$ decides correctly whether x is a quadratic residue in $\mathbb{Z}_n^*$ or not for n fixed and x selected uniformly from among all element of $\mathbb{Z}_n^*(+1)$, is less than $1 - 1/s^t$.

In their paper [1], Blum, Blum and Shub prove, assuming that the factors of n are necessary for deciding quadratic residuosity of an element $x \in \mathbb{Z}_n^*(+1)$, that these factors are necessary to have even an little advantage in guessing the parity of $x_{-1} := \sqrt{x_0}$, given the parameters n and x_0, in polynomial time. We can remark here that guessing the parity of the element to the left or to the right of a pseudo-random sequence's element is clearly equivalent. A pseudo-random sequence that "looks" random only in one direction is surely not a cryptographic secure one.

To prove that the generator is secure, "modulo" the quadratic residuosity assumption, Blum, Blum and Shub show first how an advantage in guessing the parity of an element to the left of the sequence can be converted in an advantage for determining quadratic residuosity. Then they use a result from Goldwasser and Micali to show the relation between their generator and the quadratic residuosity assumption. The goal of the next section is to give an overview of these constructions.

4.2 The proof of security

First we give the formal definitions of an $1/P$-*advantage* in guessing the parity of x_{-1} and in the quadratic residuosity problem. They come both

from [5].

In the following, I denotes an infinite set of indices; $N = \{N_k : k \in I\}$ denotes a family of non-empty sets N_k of nonnegative integers with $\forall n \in N_k$, n having binary length of exactly k. Furthermore, we assume that $A(n, x_0)$ is an algorithm which takes as input an integer $n = p \cdot q$ product of two Blum primes and an element $x_0 := x_{-1}^2 \pmod n \in QR_n$ and that it gives as output the parity of x_{-1}; $B(n, x)$ is an algorithm which takes as input n and an element $x \in \mathbb{Z}_n^*(+1)$ for which the quadratic residuosity has to be determined, and that $B(n, x)$ outputs a 1 if $x \in QR_n$ and a 0 if $x \in QNR_n$.

Definition 11 ($1/P$**-advantage in the parity of** x_{-1})
Let P be a polynomial. A poly-time algorithm $A(n, x)$ has a $1/P$-advantage for computing the parity of $x_{-1} := \sqrt{x_0}$ for the family N, if, for all but a finite number of indices $k \in I$, the following property holds $\forall n \in N_k$:

$$P[x \in QR_n | A(n, x_0) = parity(\sqrt{x_0} \pmod n)] \geq \frac{1}{2} + \frac{1}{P(k)}$$

Definition 12 ($1/P$**-advantage in the quadratic residuosity**)
Let P be a polynomial. A poly-time algorithm $B(n, x)$ has a $1/P$-advantage for determining quadratic residuosity for the family N if, for all but a finite number of indices $k \in I$, the following property holds $\forall n \in N_k$:

$$\frac{1}{2}(P[B(n, x) = 1 | x \in QR_n] + P[B(n, x) = 0 | x \notin QR_n]) \geq \frac{1}{2} + \frac{1}{P(k)}$$

where for each $n \in N_k$, x ranges over $\mathbb{Z}_n^*(+1)$.

The first step of the proof of security is based on the following theorem, which describes the reduction from an advantage over the parity of x_{-1} to an advantage over the quadratic residuosity :

Theorem 11 ([5], original version in [1])
Let n be the product of two Blum primes. An $1/P$-advantage for determining parity of $x_{-1} := \sqrt{x_0}$ given the quadratic residue $x_0 \in QR_n$ can be converted efficiently and uniformly to an $1/P$-advantage for determining quadratic residuosity of $x \in \mathbb{Z}_n^*(+1)$.

The proof of Theorem 11 is based on the following lemma :

Lemma 4
Let $n = p \cdot q$ be the product of two Blum primes. $\forall x \in \mathbb{Z}_n^*(+1)$,

$$x \in QR_n \iff parity(x) = parity(\sqrt{x^2} \pmod n)$$

Proof :

"$\Longrightarrow$" : By assumption, $x \in QR_n$. Then x is the unique square root of $x^2 \pmod n$ (see Definition 8).
"$\Longleftarrow$" : Suppose first that $x \notin QR_n$; let $x_0 := \sqrt{x^2} \pmod n$ be the unique square roots of x^2 which is a quadratic residue. We have $n = p \cdot q$. By assumption, $\left(\frac{x}{n}\right) = 1$. Using Theorem 8,we have:

$$\left(\frac{-1}{p}\right) = \left(\frac{-1}{q}\right) = -1$$

Hence

$$\left(\frac{-1}{n}\right) = \left(\frac{-1}{p}\right) \cdot \left(\frac{-1}{q}\right) = 1$$

and

$$\forall x \in \mathbb{Z}_n^* \quad \left(\frac{-x}{n}\right) = \left(\frac{x}{n}\right)$$

which is a consequence of the multiplicative property of the Jacobi symbol (see Theorem 5). We conclude that $x = -x_0$ and from Lemma 1 we know that $x \neq x_0$, so they must have different parities, n being odd, which is a contradiction.

□

By assumption we have an algorithm $A(n, x_0)$ with a polynomial advantage in computing the parity of $x_{-1} := \sqrt{x_0}$. The goal is now to find an algorithm $B(n, x)$ with a polynomial advantage in solving the problem of quadratic residuosity which uses $A(n, x_0)$ as subroutine.

Proof of Theorem 11 :

By assumption we have an algorithm with a polynomial advantage in computing the parity of $x_{-1} := \sqrt{x_0}$. Let $B(n, x)$ be the following algorithm:

$$B(n, x) = A(n, x^2 \pmod n) \oplus parity(x) \oplus 1$$

Here, $\oplus$ is a notation for the XOR binary operation. The construction of $B(n, x)$ is illustrated in Figure 3. It is clear from this definition that $B(n, x)$ outputs a 1 if and only if $A(n, x)$ predicts the good parity of $\sqrt{x^2}$. We define now the following sets :

$$A_n := \{x \in QR_n : A(n, x) = parity(\sqrt{x} \pmod n)\}$$

which is the set of quadratic residues whose square root's parity is predicted right by $A(n, x)$. The next set defines the *squared* quadratic residues for which $A(n, x)$ works fine:

$$X_n := \{x \in QR_n : x^2 \pmod n \in A_n\}$$

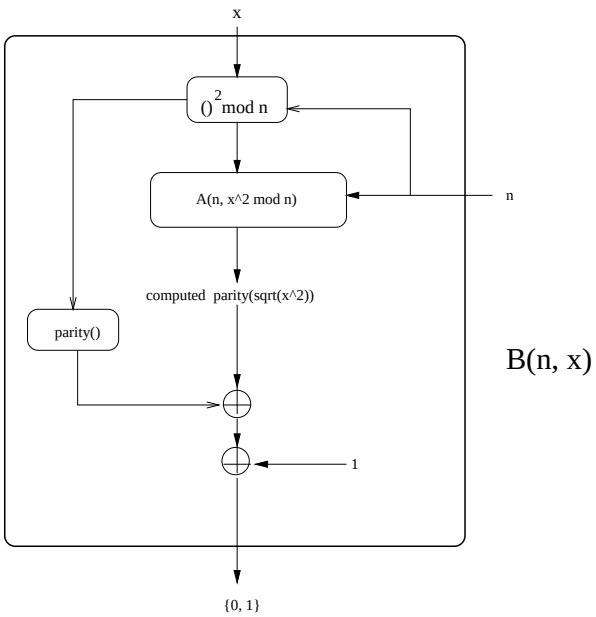

Figure 3: Description of the algorithm $B(n, x)$

For each quadratic residue x^2 whose square root's parity is good predicted by $A(n, x)$, there is bijectively an element $-x \in \mathbb{Z}_n^*(+1) \setminus QR_n$ with $x^2 = (-x)^2$ for which $A(n, x)$ produces a false output; Y_n is the set of these elements:

$$Y_n := \{x \in \mathbb{Z}_n^*(+1) \setminus QR_n : x^2 \pmod{n} \in A_n\}$$

The set W_n is the union of the two last disjunct sets :

$$W_n := \{x \in \mathbb{Z}_n^*(+1) : x^2 \pmod{n} \in A_n\}$$

Furthermore, $|X_n| = |Y_n| = |A_n|$. This equality holds because we can define bijectively the following relation: for each $a^2 \in A_n$, we have exactly one $x \in X_n$ with $x := a = \sqrt{a^2}$ and exactly one $y \in Y_n$ with $y := -a$. Following these considerations, we have clearly:

$$P[x \in \mathbb{Z}_n^*(+1) : x \in W_n] = \frac{|W_n|}{|\mathbb{Z}_n^*(+1)|}$$

$$= \frac{|X_n| + |Y_n|}{2|QR_n|} = \frac{|A_N|}{|QR_n|} = P[x \in QR_n : x \in A_n]$$

Taking Definition 12, we must now show that

$$\frac{1}{2}(P[B(n, x) = 1 | x \in QR_n] + P[B(n, x) = 0 | x \notin QR_n]) \geq \frac{1}{2} + \frac{1}{P(k)}$$

Using Theorem 3 and Lemma 4, we have

$$P[B(n,x)=1|x \in QR_n] = \frac{P[B(n,x)=1 \wedge x \in QR_n]}{P[x \in QR_n]} = \frac{P[x \in \mathbb{Z}_n^*(+1) : x \in X_n]}{\frac{1}{2}}$$

while

$$P[B(n,x)=0|x \in QNR_n] = \frac{P[B(n,x)=0 \wedge x \in QNR_n]}{P[x \in QNR_n]} = \frac{P[x \in \mathbb{Z}_n^*(+1) : x \in Y_n]}{\frac{1}{2}}$$

which gives us

$$\begin{aligned}
& \frac{1}{2}\left(P[B(n,x)=1|x \in QR_n] + P[B(n,x)=0|x \notin QR_n]\right) \\
= \; & \frac{1}{2}\left(\frac{P[x \in \mathbb{Z}_n^*(+1) \quad : \quad x \in X_n]}{\frac{1}{2}} + \frac{P[x \in \mathbb{Z}_n^*(+1) \quad : \quad x \in Y_n]}{\frac{1}{2}}\right) \\
= \; & P[x \in \mathbb{Z}_n^*(+1) : x \in X_n] + P[x \in \mathbb{Z}_n^*(+1) : x \in Y_n] \\
= \; & P[x \in QR_n : x \in A_n] \\
\geq \; & \frac{1}{2} + \frac{1}{P(k)}
\end{aligned}$$

which concludes the proof.

□

It is now possible to strengthen Definition 12 as follows :

Definition 13 ($1 - 1/P$-advantage in the quadratic residuosity, [3])
A poly-time algorithm $A(n,x)$ has a $1/P$-advantage for determining quadratic residuosity for the family N if, for all but a finite number of indices $k \in I$, the following property holds $\forall n \in N_k$:

$$\frac{1}{2}(P[A(n,x)=1|x \in QR_n] + P[A(n,x)=0|x \notin QR_n]) \geq 1 - \frac{1}{P(k)}$$

where for each $n \in N_k$, x ranges over $\mathbb{Z}_n^*(+1)$.

The following theorem is the second part in the proof of security :

Theorem 12 (Goldwasser & Micali, [3])
An $1/P$-advantage for determining quadratic residuosity can be amplified uniformly and efficiently in an $1 - 1/P$-advantage.

Sketch of the proof ([1]):

Let $x \in \mathbb{Z}_n^*(+1)$ be an element whose quadratic residuosity is to be determined. For this goal, select r at random with an uniform probability over $\mathbb{Z}_n^*$. Compute $x \cdot r^2 \pmod{n}$. We have now the following properties : for $x \in QR_n$, $x \cdot r^2 \pmod{n}$ is uniformly distributed over QR_n and for $x \in QNR_n$, $x \cdot r^2 \pmod{n}$ is uniformly distributed over $\mathbb{Z}_n^*(+1) \setminus QR_n$. Test each of the resulting numbers, $x \cdot r^2 \pmod{n}$, for quadratic residuosity. Taking the majority vote amplifies the advantage as much as one likes.

□

The main theorem is the following :

Theorem 13 (Blum, Blum, Shub, [1])
The Blum-Blum-Shub PRBG is an unpredictable (cryptographic secure) generator, i.e., for each probabilistic poly-time predicting algorithm $A(n, x)$, and positive integer t, A has at most an $1/s^t$-advantage for n in predicting sequences to the left, s being the length of n, for sufficiently large n and for all but $1/s^t$ prescribed numbers n of length s.

Proof :

Suppose we have a predicting algorithm $A(n, x)$ with an $1/P$-advantage for n. This can be converted efficiently and uniformly into an algorithm with an $1/P$-advantage in guessing the parity of x_{-1} given an arbitrary $x_0 \in QR_n$. From seed x_0 generate the sequence $b_0 b_1 b_2 \ldots$. Then the parity of x_{-1} is b_{-1}. We can now convert Theorem 11 to a procedure guessing quadratic residuosity with an amplified advantage (see Theorem 12) to get a contradiction to the Assumption 1.

□

References

[1] Lenore Blum, Manuel Blum, and Michael Shub. Comparison of two pseudo-random number generators. In R. L. Rivest, A. Sherman, and D. Chaum, editors, *Proc. CRYPTO 82*, pages 61–78, New York, 1983. Plenum Press.

[2] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Computing*, 13(4):850–863, November 1984.

[3] S. Goldwasser and S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *Proc. 14th ACM Symp. on Theory of Computing*, pages 365–377, San Francisco, 1982. ACM.

[4] Donald E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, Reading, MA, USA, third edition, 1997.

[5] E. Kranakis. *Primality and Cryptography*. Wiley-Teubner Series in Computer Science, 1986.

[6] A. J. (Alfred J.) Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of applied cryptography*. The CRC Press series on discrete mathematics and its applications. CRC Press, 2000 N.W. Corporate Blvd., Boca Raton, FL 33431-9868, USA, 1997.

[7] A. C. Yao. Theory and application of trapdoor functions. In *Proc. 23rd IEEE Symp. on Foundations of Comp. Science*, pages 80–91, Chicago, 1982. IEEE.