EE 720 - Assignment 2
Blum Blum Shub - PRNG

Prajval Nakrani, 17D070014
Siddharth Maniar, 170020006

November 15, 2020

1 Introduction

1.1 Psuedo Random Number Generators

A pseudorandom number generator (PRNG), also known as a deterministic random bit generator (DRBG), is an algorithm for generating a sequence of numbers whose properties approximate the properties of sequences of random numbers. A PRNG-generated sequence is not random in true sense, because it is completely determined by an initial value, called as seed (which may include truly random values). Although sequences that are closer to truly random can be generated using hardware random number generators, pseudorandom number generators are important in practice for their speed and their reproducibility. PRNGs are widely applicable in the fields of simulation and analysis (for Monte Carlo simulations), vidoe games (e.g. for procedural generation) and mainly in the field of Cryptography. Although, cryptographic applications require the PRNGs to satisfy some more stringent properties than the simpler PRNGs available. Specifically, for a cryptographic application, a PRNG must satisfy the property of "**Unpredictability**", i.e. the state or the output of PRNG should not be predictable from the knowledge of previous states or outputs.

1.2 Blum Blum Shub

The Blum Blum Shub PRNG is also known as the x^2 mod N generator. The generator takes inputs N, x_0 where $N = P \cdot Q$, where P and Q are distinct primes and both congruent to 3mod4 and x_0 is a quadratic residue mod N. And it outputs a random sequence $b_0 b_1 b_2 \dots$ where b_i =parity(x_i) and $x_{i+1} = x_i^2 \bmod N$.
The above generator displays certain interesting properties as follows:

- From the knowledge of x_0 and N but not P and Q, one can generate the sequence forwards but one cannot generate the sequence backwards.
- With the additional knowledge of P and Q, one can generate the sequence backwards and one can even jump from any point in the sequence to any other point in the sequence.

As a result, the generator shows good promises in application in public-key cryptography. In the rest of the report, the algorithm is explained in detail followed by a rough explanation for its security/unpredictability. A scaled down version of the algorithm is then implemented in Python3 followed by extensive experimental testing of the robustness of the generator based on certain common metrics like entropy, auto-correlation etc.

2 Algorithm

First we define certain notations that will be useful in compact representation of the algorithm. Then we state the assumptions that are involved in the theory that follows next. Then we introduce the algorithm following which we provide, modulo the assumptions, theoretical reasoning for its security or unpredictability.

2.1 Notations

- The base, b, is always an integer > 1
- Let N be any integer > 0
- $|N|_b = \lfloor 1 + \log_b N \rfloor$ represents the *length of* N when expanded in base b.
- $n = |N| = |N|_2$, hence it follows; $N = O\left(2^n\right)$
- $\Sigma = \{0, 1, \cdots, b-1\}$
- Σ^* denotes the set of finite sequences of elements of Σ
- Σ^∞ denotes the set of infinite sequences of elements of Σ
- For $x \in \Sigma^*$, let $|x|$ be the length of x and for $k \geq 0$, let $\Sigma^k = \{x \in \Sigma^* || x \mid = k\}$
- For $x \in \Sigma^\infty$ and $k \geq 0$, x^k denotes the *initial segment of* x of length k and x_k denotes the kth element of x

2.2 Definitions

- $\mathbf{N} = \left\{ \text{ integers } N \mid N = P \cdot Q, \text{ such that } P, Q \text{ are equal length } (|P| = |Q|) \text{ distinct primes} \equiv 3 \bmod 4 \right\}$
- $\mathbf{X}_N = \left\{x^2 \bmod N \mid x \in Z_N^*\right\}$ denotes the set of quadratic residues mod N
- $\mathbf{X} = \text{disjoint} \bigcup_{N \in \mathbf{N}} \mathbf{X}_N$ denotes the ***Seed Domain***.
- $\mathbf{X}^n = \{(N, x) \mid N \in \mathbf{N}, |N| = n, \text{ and } x \in \mathbf{X}_N\}$ denotes the set of seeds of length n in seed domain $\mathbf{X}$
- A pair (**X**,U), where **X** is a seed domain and U is an accessible distribution on **X**, is called a seed space.
- For $(N, x) \in \mathbf{X}^n$, let $\mu_n(N, x) = u_n(N) \cdot v_N(x)$, where u_n is uniform distribution on $\{N \in \mathbf{N} \mid |N| = n\}$ and v_N is a uniform distribution on X_N
- Let the *transformation* $T : \mathbf{X} \to \mathbf{X}$ be defined by $T(x) = x^2 \bmod N$ for $x \in \mathbf{X}_N$
- Let the *partition* $B : \mathbf{X} \to \{0, 1\}$ be defined by $B(x) = \text{parity}(x)$

2.3 The Generator

The following claim can be proved easily; The distribution $\mu_n(N, x)$ defined earlier is an *accessible probability distribution* on **X**. i.e. there exists a probabilistic polynomial time procedure that samples on $\mathbf{X}^n$ with distribution μ'_n, where for all t and sufficiently large n, we have

$$\sum_{(N,x) \in X^n} |\mu_n(N, x) - \mu'_n(N, x)| < 1/n^t$$

Given the above claim, the system $\langle X, T, B \rangle$, with **X** a seedspace, T a transformation on X and B a partition all as defined in the previous section, naturally defines a pseudo-random sequence generator G on **X** where the kth coordinate, $[G(N, x)]_k = B\left(T^k x\right)$. This is called the x^2 mod N generator or more informally known as the "**Blum Blum Shub**" (BBS) generator. Thus with inputs (N, x_0), the BBS generator outputs the pseudo-random sequence of bits $b_0 b_1 b_2 \ldots$ obtained by setting $x_{i+1} = x_i^2 \bmod N$ and extracting the bit b_i =parity(x_i)

3 Cryptographic Security of the BBS Algorithm

3.1 Assumptions

The **Quadratic Residuacity Assumption** (QRA) asserts that any efficient procedure for guessing quadratic residuacity will be incorrect for a fraction of the inputs. Let $\mathbf{P}[N, x]$ be any polynomial time probabilistic procedure which outputs a number between 0 and 1 denoting the probability of x being a quadratic residue. Then for some positive integer t, the following inequality holds true

$$\left(\frac{\sum_{x \in Z_N^*(+1)} \text{Prob}(\mathbf{P}[N, x] \text{ is incorrect })}{\varphi(N)/2} \right) > 1/n^t$$

3.2 Theoretical Reasoning for Security

Based on the formulation mention so far, the following property obviously holds true

- Knowledge of N is sufficient to efficiently generate the sequence of bits $b_0 b_1 b_2 \ldots$ in the forward direction given the seed or the initial state x_0.

Lemma 3.1. *If $N = P \cdot Q$ where P and Q are distinct primes such that $P \equiv Q \equiv 3 \bmod 4$, then each quadratic residue mod N has exactly one square root that is a quadratic residue.*

Theorem 3.2. *There exists an efficient deterministic algorithm A which when given N and its prime factors and any quadratic residue x_0 in Z_N^* efficiently computes the unique quadratic residue $x_{-1} \bmod N$ such that $(x_{-1})^2 \bmod N = x_0$*

Proof. In order to prove theorem 3.2, we provide an algorithm that computes the sequence in reverse order. Let A be the required algorithm, then we want,

$$A\left(P, Q, x_0\right) = x_{-1}$$

- Given: N, P, Q, x_0
- Compute $x_P = \sqrt{x_0} \bmod P$, $x_Q = \sqrt{x_0} \bmod Q$
- Using Euclidean algorithm, find out integers u, v such that, $P \cdot u + Q \cdot v = 1$
- Obtain the number, $x_N = \pm x_P \cdot Q \cdot v \pm x_Q \cdot P \cdot u$
- Observe, $x_N = \sqrt{x_0} \bmod N$ is a quadratic residue w.r.t both P and Q and hence w.r.t N
- From lemma 3.1, since each quadratic residue has exactly one square root, we get, $x_{-1} = x_N$

□

Definition 3.1. Given a polynomial poly and $0 < \varepsilon \leq 1/2$, a $0-1$ valued probabilistic polynomial time procedure has an ε -advantage for N in guessing parity, if and only if

$$\frac{\sum_{x_0 \in QR_N} \text{Prob}\left[\mathbf{P}\left[N, x_0\right] = Parity(x_{-1})\right]}{\varphi(N)/4} \geq (1/2) + \varepsilon$$

Lemma 3.3. *An $\varepsilon-$advantage for guessing parity (of x_{-1} given x_0) can be converted efficiently and uniformly to an $\varepsilon-$advantage for guessing quadratic residuacity (of x in $Z_N^*(+1)$)*

Lemma 3.4. *An $\varepsilon-$advantage for guessing quadratic residuacity can be amplified to a $1/2 - \varepsilon$ advantage, efficiently and uniformly.*

Theorem 3.5. *Modulo the QRA, the BBS generator is a cryptographically secure (unpredictable) pseudo-random sequence generator. That is to say, for each probabilistic polynomial time predictor $\boldsymbol{P}$, each constant $0 < \delta < 1$, and positive integer t, $\boldsymbol{P}$ has at most a $1/n^t$ advantage for N in predicting sequences to the left (for sufficiently large n and for all but a fraction of prescribed numbers N of length n).*

Proof. We prove the theorem using contradiction. QRA applies as mentioned in the assumptions section.

- Assume there is a predictor for the BBS generator with an $\varepsilon-$advantage for N.
- This can be converted efficiently and uniformly into a procedure with an $\varepsilon-$advantage in guessing the parity of x_{-1} given arbitrary x_0 in QR_N. (This is straight forward from the algorithm itself)
- Now this can be converted efficiently and uniformly into a procedure for guessing quadratic residuacity with $\varepsilon-$advantage. (lemma 3.3)
- This in turn can be converted efficiently and uniformly into a procedure for guessing quadratic residuacity with and amplified advantage.
- However, this statement is a contradiction to the Quadratic Residuacity Assumption.
- Hence, our asusmption of the predictor with an $\varepsilon-$advantage is false.
- Hence, there cannot be a predictor for the BBS generator with $\varepsilon-$advantage for N.

□

4 Experimental Analysis of the BBS PRNG

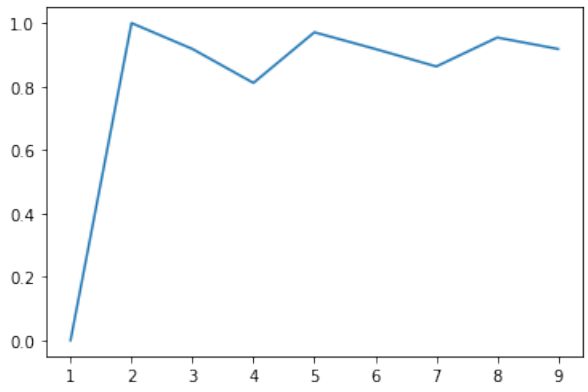
We evaluate the randomness of the sequence generated based on two major properties of the sequence viz. 1) **Entropy** and 2) **Auto correlation**.

4.1 Entropy Test

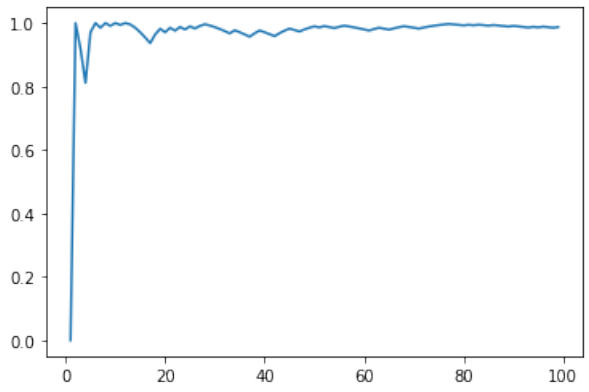
For a long truly random bit sequence generated by repeated flipping of an unbiased coin, the entropy of the sequence tends to 1 as number of coin tosses tends to infinity. We test the pseudo-random sequence generated by BBS algorithm to see if it follows this pattern of a truly random sequence. The observations and results are as follows:

Length of Sequence	Entropy
10	0.918
100	0.987
1000	0.999
10000	0.999

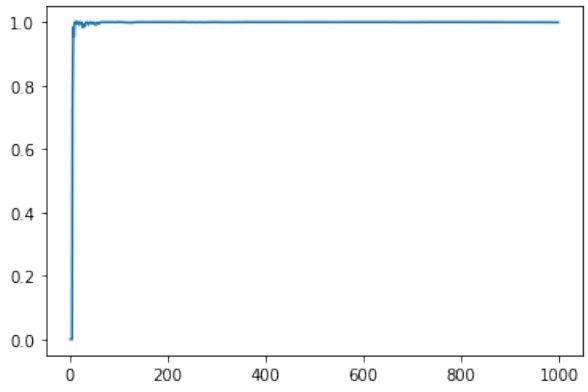
Table 1: Entropy of Pseudo-Random Sequence of Varying lengths



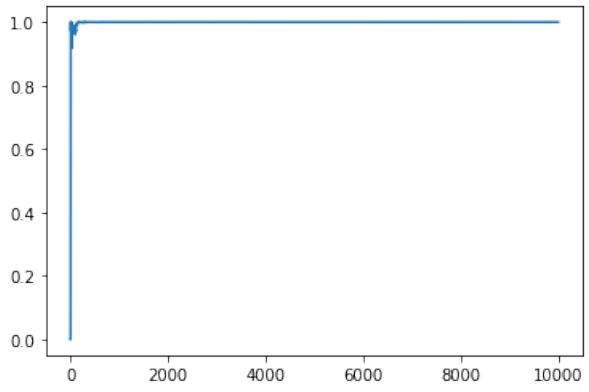
(a) Length = 10



(b) Length = 100



(c) Length = 1000



(d) Length = 10000

Figure 1: (Entropy of first K elements) v/s K for Pseudo-Random Sequences with varying length

Observations:

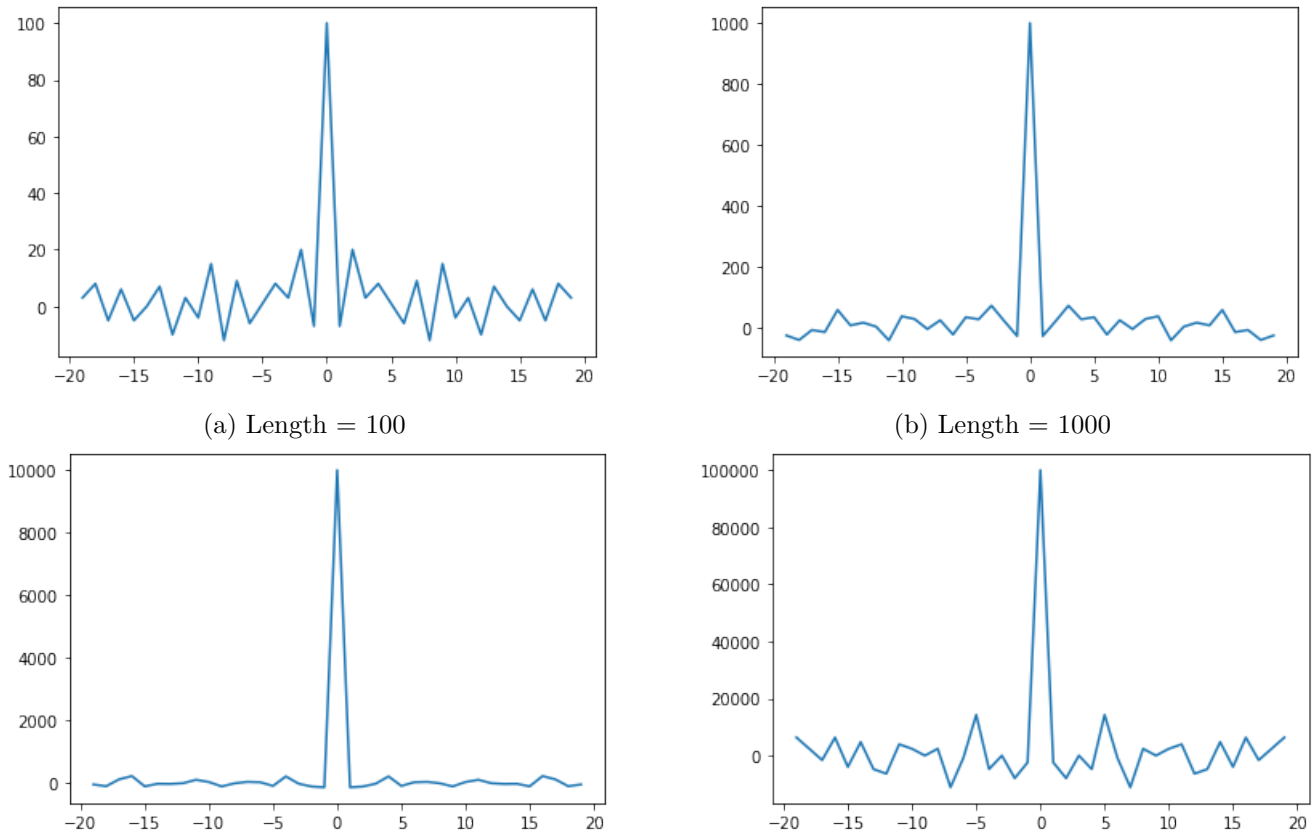
- We observe that for all length sequence generated, the entropy value of the sequence saturates at value 1 for long sequences.
- This is the exact behaviour expected out of a truly random sequence.
- This is a good indicator of the fact that there is enough mixing of 1's and 0's i.e. there are equal number of 1's and 0's in a long sequence generated by the BBS algorithm

4.2 Auto Correlation Test

A truly random bit sequence generated by repeated flipping of an unbiased coin is essentially a Stationary signal and an Ergodic signal. This is an intuitive fact and proof lies in the domain of signal processing which we won't discuss here. Hence, we can use the ergodic definition of auto correlation function as follows:

$$\mathbf{R}_{xx}(k) = \mathrm{E}[X_t X_{t+k}] = \lim_{M\to\infty} \frac{1}{M} \sum_{i=1}^{M} X[t]X[t+k]$$

For a truly random bit sequence, the auto correlation function is indeed a delta function $\delta(x)$. Now we test the pseudo-random sequence generated by the BBS algorithm.



(a) Length = 100

(b) Length = 1000

(c) Length = 10000

(d) Length = 100000

Figure 2: Auto Correlation function upto index 20 for Pseudo-Random Sequences of varying length

Observations:

- We observe that the auto correlation functions are very close to delta function for all sufficiently long sequence lengths.
- The auto correlation function will better and better approximate a delta function as the sequence becomes longer and longer.
- We had changed the {0,1} sequence to {-1,1} sequence for ease of function evaluation and hence we observe that R[0] = length of sequence for all pseudo-random sequences generated by the BBS algorithm.
- This delta function nature of the auto correlation functions is an indicator of the fact that sequence values exhibit almost negligible correlation for any two elements in the sequence separated by distance $k > 0$.
- Hence, it is a positive indication of the "good" noisy-ness or randomness of the generated pseudo-random bit sequence.