Blum-Blum-Shub cryptosystem and generator
A prime $p$ is called a **Blum prime** if $p \mod 4 = 3$.

**ALGORITHM**

- Alice, the recipient, makes her BBS key as follows:
A prime $p$ is called a **Blum prime** if $p \mod 4 = 3$.

ALGORITHM

- Alice, the recipient, makes her BBS key as follows:
  1. She chooses two distinct Blum primes $p$ and $q$ and computes their product, $n = pq$. The number $n$ will be her public key, while its factorization is her private key.
Bob, the sender, encrypts as follows:

1. He chooses a random number $1 < x_0 < n$ which is a quadratic residue modulo $n$, and computes the sequence $x_0, x_1, x_2, \ldots, x_n, x_{n+1}$ where for each $i \in [0, n]$, 
   \[ x_{i+1} = x_i^2 \mod n. \]
2. For each $i$, he computes $e_i = b_i + x_i$ mod 2 where $b_i$ are the message bits.
3. He sends the encrypted message $e_0, e_1, e_2, \ldots, e_n$, as well as $x_n + 1$ to Alice.
Bob, the sender, encrypts as follows:

1. He chooses a random number $1 < x_0 < n$ which is a quadratic residue modulo $n$, and computes the sequence $x_0, x_1, x_2, \ldots, x_n, x_{n+1}$ where for each $i \in [0, n]$, $x_{i+1} = x_i^2 \mod n$. 

Blum-Blum-Shub cryptosystem and generator
Bob, the sender, encrypts as follows:

1. He chooses a random number $1 < x_0 < n$ which is a quadratic residue modulo $n$, and computes the sequence
   $x_0, x_1, x_2, ..., x_n, x_{n+1}$ where for each $i \in [0, n]$,
   $x_{i+1} = x_i^2 \mod n$.

2. For each $i$, he computes $e_i = b_i + x_i \mod 2$ where $b_i$ are the message bits.
Bob, the sender, encrypts as follows:

1. He chooses a random number $1 < x_0 < n$ which is a quadratic residue modulo $n$, and computes the sequence $x_0, x_1, x_2, ..., x_n, x_{n+1}$ where for each $i \in [0, n]$, $x_{i+1} = x_i^2 \mod n$.

2. For each $i$, he computes $e_i = b_i + x_i \mod 2$ where $b_i$ are the message bits.

3. He sends the encrypted message $e_0, e_1, e_2, ..., e_n$, as well as $x_{n+1}$ to Alice.
Alice decrypts as follows:

1. She recovers the $x_i$'s in the order: $x_n, x_{n-1}, \ldots, x_2, x_1, x_0$.
2. She recovers the message by performing the computations $m_i = e_i + x_i \mod 2$ for $i = 0, 1, 2, \ldots, n$. 
Alice decrypts as follows:

1. She recovers the $x_i$'s in the order: $x_n, x_{n-1}, \ldots, x_2, x_1, x_0$. 
Alice decrypts as follows:

1. She recovers the $x_i$'s in the order: $x_n, x_{n-1}, \ldots, x_2, x_1, x_0$.
2. She recovers the message by performing the computations $m_i = e_i + x_i \ mod \ 2$ for $i = 0, 1, 2, \ldots, n$. 
Alice decrypts as follows:

1. She recovers the $x_i$'s in the order: $x_n, x_{n-1}, \ldots, x_2, x_1, x_0$.
2. She recovers the message by performing the computations 
   \[ m_i = e_i + x_i \mod 2 \text{ for } i = 0, 1, 2, \ldots, n. \]
 Exhaustive search attack on the random number $y$. \\
Security of the BBS encryption scheme

- Exhaustive search attack on the random number \( y \).
- Repeated use of the random number \( y \) can be dangerous.
Security of the BBS encryption scheme

- Exhaustive search attack on the random number $y$.
- Repeated use of the random number $y$ can be dangerous.
What you need to know

A pseudorandom generator is a deterministic algorithm that, given a truly random binary sequence of length $n$, outputs a binary sequence of length $m > n$ that “looks random.”

The input to the generator is called the seed.

The output is called the pseudorandom bit sequence.

Security of a pseudorandom generator is a characteristic that shows how hard it is to tell the difference between the pseudorandom sequences and truly random sequences.

For the Blum-Blum-Shub pseudorandom generator distinguishing these two sequences is as hard as factoring a large composite integer.
What you need to know

- A pseudorandom generator is a deterministic algorithm that, given a truly random binary sequence of length $n$, outputs a binary sequence of length $m > n$ that “looks random”.

Blum-Blum-Shub cryptosystem and generator
What you need to know

▶ A pseudorandom generator is a deterministic algorithm that, given a truly random binary sequence of length \( n \), outputs a binary sequence of length \( m > n \) that “looks random”.

▶ The input to the generator is called *the seed*. 
Blum-Blum-Shub pseudo random number generator

What you need to know

- A pseudorandom generator is a deterministic algorithm that, given a truly random binary sequence of length \( n \), outputs a binary sequence of length \( m > n \) that “looks random”.
- The input to the generator is called the seed.
- The output is called the pseudorandom bit sequence.
What you need to know

- A pseudorandom generator is a deterministic algorithm that, given a truly random binary sequence of length $n$, outputs a binary sequence of length $m > n$ that “looks random”.
- The input to the generator is called the seed.
- The output is called the pseudorandom bit sequence.
- Security of a pseudorandom generator is a characteristic that shows how hard it is to tell the difference between the pseudorandom sequences and truly random sequences.
What you need to know

- A pseudorandom generator is a deterministic algorithm that, given a truly random binary sequence of length $n$, outputs a binary sequence of length $m > n$ that “looks random”.
- The input to the generator is called *the seed*.
- The output is called *the pseudorandom bit sequence*.
- Security of a pseudorandom generator is a characteristic that shows how hard it is to tell the difference between the pseudorandom sequences and truly random sequences.
- For the Blum-Blum-Shub pseudorandom generator distinguishing these two sequences is as hard as factoring a large composite integer.
ALGORITHM

- Generate $p$ and $q$, two big Blum prime numbers.
- $n := p \cdot q$.
- Choose $s \in [1, n - 1]$, the random seed.
- $x_0 := s^2 \mod n$.
- The sequence is defined as $x_i := x_{i-1}^2 \mod n$ and $z_i := \text{parity}(x_i)$.
- The output is $z_1, z_2, z_3, ...$ where $\text{parity}(x_i)$ is 0 when $x_i$ is even and 1 when $x_i$ is odd.