Implementation of Blum Blum Shub Random Number Generator into RSA and Timing Attack

Runmin Wang, Zihao Song

Abstract—A reliable pseudo random number generator is deterministic for RSA. To implement RSA, firstly we mainly focus on how to fastly and efficiently implement a random number generator to provide two long enough prime numbers. Here we prefer the Blum Blum Shub Random Number Generator, because it can quickly generate long sequences from short seeds and it is unpredictable, even though it has quadratic residuacity problem. Depending on various types of BBS random number generators, in order to make it fast and efficient, we have the idea to choose the fast BBS sequence random number generator using montgomery multiplication, by taking the advance of the montgomery multiplication taken out of the squaring root to make it parallel. Also, it can be faster if the multiplicative offset is retained to the Blum Blum Shub sequence which make the modulus larger.

After implementing the Blum Blum Shub random number generator, we gain two long enough numbers, and then we can generate a group of numbers. By using the Miller-Rabin test, we can verify which number is a prime number and choose any two from this group. They are the two input prime numbers (p and q, where n = p*q) for RSA. Only if this pair of prime number is long enough, the probability of leaking information tends to be much lower, even considering a very little partial. By using the RSA algorithm, we can gain a pair of public/private key that can be matched with each other, which can be implemented to most of secure system. Montgomery will be implemented as the exponential function.

Our last step is to try to attack our RSA system contributed by the Blum Blum Shub random number generator by timing attack. It functions as a different type of evaluation for BBS based RSA security system. Since we know the detail implementation of our RSA algorithm, the branching of Montgomery will be point of attack. Based on whether the current bit is 1 or 0, an additional multiplication will be performed. This creates a difference of calculating time. By measuring this difference, we can explore one bit of the secret key. With one bit in hand, we can keep on doing this and crack the whole secret key.