

# Aspect of Prime Numbers in Public Key Cryptosystem

Md.Mehedi Masud, Huma Galzie, Kazi Arif Hossain and Md.Minhaj Ul Islam

Computer Science and Engineering Discipline  
Khulna University, Khulna-9208, Bangladesh  
Email: mmehedi@bttb.net.bd

## Abstract

*The secrecy of Public Key cryptosystem relies heavily on the intractability of various mathematical problems. As a result numbers and their complexities play a very important role in keeping the system secure. Prime numbers are of the greatest importance here due to their widespread use in encryption algorithms such as the RSA algorithm. For example, if there were efficient algorithms for factoring large numbers, the vast majority of encrypted electronic communications worldwide would be easily crackable. In most of the public key algorithms large prime numbers (100 decimal digits) have been used as a means of securing the system. In this paper we deal with the subject of handling prime numbers efficiently in cryptosystems and give an overview of certain efficient primality tests. We hope it's a new look to the prime numbers.*

## 1. Introduction

Based on their factorability, numbers are broadly classified as prime numbers and composite numbers. A **prime number** [6] is a positive integer, which has no divisors other than 1 and itself. Positive integers other than 1 that are not prime are called **composite** [6]. Although the number 1 used to be considered a prime, it requires special

treatment in so many definitions and applications involving primes greater than or equal to 2 that it is usually placed into a class of its own. Since 2 is the only **Even prime**, it is also somewhat special, so the set of all primes excluding 2 is called the **Odd Primes**. The first few primes are 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37,.....[6]

Let  $\{E_e : e \in K\}$  be a set of encryption transformations, and correspondingly let  $\{D_d : d \in K\}$  be the set of decryption transformations, where  $K$  is the key space. The cryptosystem is said to be a **public key cryptosystem** [1] if for each associated key pair  $(e,d)$  the public key  $e$ , is made available to all while the private key  $d$ , is kept secret. Consider any pair of associated encryption/decryption transformations  $(E_e; D_d)$ ; each pair has the property that knowing  $E_e$  it is computationally infeasible, given a random ciphertext  $c \in C$ , to find the message  $m \in M$  such that  $E_e(m) = c$ . This property implies that given  $e$  it is infeasible to determine the corresponding decryption key  $d$  [1].

Public key cryptosystems exploit the idea of using two different keys ---- one for encryption and one for decryption ---- hence being asymmetric. The most natural question that arises here is --- Where do prime numbers feature in public key cryptosystems? All public key cryptosystems are based on one or the other one-way

trapdoor functions. These functions mostly involve mathematical operations on primes. And this is where primes come into the picture. Prime numbers play a twofold role here. Firstly, they are used in generating the keys and secondly, their properties are used in keeping the decryption key secret.

## 2. Preliminaries

### 2.1 Properties of primes

1. The function which gives the number of primes less than a number  $n$  is denoted by  $\varnothing(n)$  and is called the **Euler Totient Function** or the **Euler Phi Function** [5]. The theorem giving an asymptotic form for  $\varnothing(n)$  is called the **Prime Number Theorem**. If  $n$  is prime then  $\varnothing(n) = n-1$ .

2. The **Fundamental Theorem of Arithmetic** [6] states that any positive integer can be represented in exactly one way as a product of primes. Every integer  $n \geq 2$  has a factorization as a product of prime powers as shown :  $n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$ , where the  $p_i$  are pair wise distinct primes and each  $e_i \geq 1$ .

3. **Euclid's Second Theorem** demonstrated that there are an infinite number of primes. This theorem, also called the **infinitude of primes theorem** [7], was proved by Euclid.

4. Given a finite sequence of consecutive primes  $2, 3, 5, \dots, p$ , the number  $N = 2*3*5*\dots*(p+1)$  known as the  **$i$ th Euclid number** [7], when  $p = p_i$  is the  $i$ th prime, is either a new prime or the product of primes. If  $N$  is a prime, then it must be greater than the previous primes, since one plus the product of primes must be greater than each prime composing the product.

5. The probability that the largest prime factor of a random number  $x$  is less than  $\sqrt{x}$  is  $\ln 2$  [7].

6. **Mersenne primes** are primes of the form  $2^p-1$ . These are the easiest types of number to check for primality on a binary computer so they usually are also the largest primes known [8].

7. **Twin primes** are primes of the form  $p$  and  $p+2$ , i.e., they differ by two [8].

8. A prime number  $p$  is said to be a **strong prime** [8] if integers  $r$ ,  $s$ , and  $t$  exist such that the following three conditions are satisfied:

- (i)  $p-1$  has a large prime factor, denoted  $r$ ;
- (ii)  $p + 1$  has a large prime factor, denoted  $s$ ; and
- (iii)  $r-1$  has a large prime factor, denoted  $t$ .

Here  $p-1$  and  $p+1$  are twin primes.

### 2.2 Scope of primes in Public Key Cryptosystem

Prime numbers are a very important tool in any public key cryptosystem. They are used in almost every public key system to generate the public as well as the private key. The efficient generation of public-key parameters is a prerequisite in public-key systems.

A specific example is the requirement of a prime number  $p$  to define a finite field  $Z_p$  for use in the Diffie-Hellman key agreement protocol. In this case, an element of high order in  $Z_p^*$  is also required. Another example is the requirement of primes  $p$  and  $q$  for an RSA modulus  $n=p*q$  [1]. In this case, the prime must be of sufficient size, and be random in the sense that the probability of any particular prime being selected must be sufficiently small to

preclude an adversary from gaining advantage through optimizing a search strategy based on such probability. Prime numbers may be required to have certain additional properties, in order that they do not make the associated cryptosystems susceptible to specialized attacks.

## 2.3 Facts and Findings

### 2.3.1 Fact 1

**A source of large random prime integers is an essential part of recent cryptosystems.** [3]

**Finding:** The generation of primes is fundamental to the concept of cryptography. Most current cryptosystems like RSA, Diffie-Hellman, El-Gamal etc. rely on primes for the generation of their public and private keys.

### 2.3.2 Fact 2

**The primes used in current cryptosystem need to be very large.** [3]

**Finding:** The primes are very large integers having of the order of 100 decimal digits. This use of large primes can be justified as follows:

To construct the cipher keys for say RSA, we need an appropriate value for the modulus  $n$ . So, we start by randomly picking two prime numbers,  $p$  and  $q$ , and multiplying them together. There are two good reasons [5] for selecting a value for  $n$  that has exactly two prime factors.

1. Firstly, we have an easy formula for calculating  $\phi(n)$ . Since  $n = p * q$ , then

$$\phi(n) = (p - 1) * (q - 1)$$

2. Secondly, we want  $n$  to be hard to factor. The fewer factors a number has, the longer it takes to find them [5].

### 2.3.3 Fact 3

**The security of most current public key cryptosystems are based on one-way trapdoor functions involving primes.** [1]

**Finding:** Most of the current cryptosystems are based on one-way trapdoor functions, which in turn are based on some intractable property of primes, eg. RSA is based on the Integer Factorization Problem [2] involving the factorization of a number into its prime components. Similarly, the ElGamal Cryptosystem is secure due to the intractability of the Discrete Logarithm Problem which requires us to find an element in the multiplicative group of another prime number.

## 3. Prime Number generation

Prime number generation has remained a taboo for the cryptographer. This is mainly due to the huge size of the primes involved. It has been found that primality checking is the means of generating primes in any system.

### 3.1 General method

The most natural method is to generate a random number  $n$  of appropriate size, and check if it is prime. This can be done by checking whether  $n$  is divisible by any of the prime numbers  $n$  [7].

Consider the following approach:

1. Generate as candidate a random odd number  $n$  of appropriate size.
2. Test  $n$  for primality.
3. If  $n$  is composite, return to the first step.

A slight modification is to consider candidates restricted to some *search sequence* starting from  $n$ ; a trivial search sequence which may be used is  $n, n + 2, n + 4, n + 6, \dots$  Using specific search sequences

may allow one to increase the expectation that a candidate is prime.

### 3.2 Primality tests

The test for primality [2] might be either:---

**True primality test** which proves that the candidate is prime called a *provable prime*, or **Probabilistic primality test** which establishes a weaker result, such as that  $n$  is “probably prime” called a *probable prime*.

#### 3.2.1 Probabilistic primality tests

Probabilistic primality tests have the following framework [2]. For each odd positive integer  $n$ , a set  $W(n)$  which is a subset of  $Z_n$  is defined such that the following properties hold:

- (i) given  $a \in Z_n$ , it can be checked in deterministic polynomial time whether  $a \in W(n)$ ;
- (ii) if  $n$  is prime, then  $W(n) = \emptyset$ , (the empty set); and
- (iii) if  $n$  is composite, then  $\#W(n) \geq n/2$

If  $n$  is composite, the elements of  $W(n)$  are called *witnesses* [2] to the compositeness of  $n$ , and the elements of the complementary set  $L(n) = Z_n - W(n)$  are called *liars*.

##### 3.2.1.1 Method

A probabilistic primality test utilizes these properties of the sets  $W(n)$  in the following manner. Suppose that  $n$  is an integer whose primality is to be determined. An integer  $a \in Z_n$  is chosen at random, and it is checked if  $a \in W(n)$ . The test outputs “composite” if  $a \in W(n)$  and outputs “prime” if  $a \notin W(n)$ . If indeed  $a \in W(n)$ , then  $n$  is said to *fail the primality test for the base a*; in this case,  $n$  is surely composite. Otherwise,  $n$  is said to *pass the primality test for the base a*; in this case, no conclusion with absolute certainty can be

drawn about the primality of  $n$ , and the declaration “prime” may be incorrect.

#### 3.2.1.2 Facts and Findings

1. Any single execution of this test which declares “composite” establishes this with certainty.

2. Successive independent runs of the test all of which return the answer “prime”, allow the confidence that the input is indeed prime, to be increased to whatever level is desired — the cumulative probability of error is multiplicative over independent trials. If the test is run  $t$  times independently on the composite number  $n$ , the probability that  $n$  is declared “prime” all  $t$  times (i.e., the probability of error) is at most  $(1/2)^t$ .

#### 3.2.1.3 Examples

Here we give two of these probabilistic primality tests.

**1. Fermat’s test: Fermat’s Theorem** [6] states that for any prime number  $n$  and any integer  $a$  such that  $1 \leq a \leq n-1$  and  $\gcd(a, n) = 1$ ,

$$a^n \equiv a \pmod{n} \Rightarrow a^{n-1} \equiv 1 \pmod{n}$$

$$\Rightarrow a^{n-1} - 1 \equiv 0 \pmod{n}$$

Therefore, given an integer  $n$  whose primality is under question, finding any integer  $a$  in this interval such that this equivalence is not true is called the Fermat’s primality test which is a necessary but not sufficient test for primality. Fermat’s theorem shows that, if  $n$  is prime, there *does not* exist a base  $a < n$  with  $\gcd(a, n) = 1$  such that  $a^{n-1} - 1$  possesses a nonzero residue modulo  $n$ . If such base  $a$  exists,  $n$  is therefore guaranteed to be composite. However, the lack of a nonzero residue in Fermat’s theorem does *not* guarantee that  $p$  is prime.

Let  $n$  be an odd composite integer. An integer  $a$ ,  $1 \leq a \leq n-1$ , such that  $a^{n-1} \not\equiv 1 \pmod{n}$  is called a **Fermat witness** (to compositeness) for  $n$ .

Let  $n$  be an odd composite integer and let  $a$  be an integer,  $1 \leq a \leq n-1$ . Then  $n$  is said to be a **pseudoprime to the base a** if  $a^{n-1} \equiv 1 \pmod{n}$ . The integer  $a$  is called a **Fermat liar** (to primality) for  $n$ .

The test : A random large integer  $n$  is generated and then a large numbers (say  $t$ ) of the number  $a$  is taken, and for each  $(a,n)$  pair we check whether  $a^{n-1} - 1$  possesses a nonzero residue modulo  $n$ . We start with  $a = 2$  and test  $n$  for primality as mentioned above. If anything besides 0 results, we know with certainty that the number is composite. If the residue modulo  $n$  is 0, however, we try the test again with 3, and then 4, and so on. If we keep getting back 0 as the result, it soon becomes rather unlikely that the number is anything but prime. Here  $t$  is the security parameter which binds the error.

A **Carmichael number**  $n$  [2] is a composite integer such that  $a^{n-1} \equiv 1 \pmod{n}$  for all integers  $a$  which satisfy  $\gcd(a, n) = 1$ . Carmichael numbers (the smallest of which is 561) give zero residue for **any** choice of the base  $a$  relatively prime to  $n$ .

Besides these, there are composite numbers known as **Fermat pseudoprimes** or **pseudoprimes** [7] which have zero residue for some  $a$ 's and so are not identified as composite.

**2. Miller-Rabin test** : The probabilistic primality test used most in practice is the Miller-Rabin test [2], or the **strong pseudoprime test**. The test is based on the properties of **strong pseudoprimes**. A

strong pseudoprime to a base  $a$  is an odd composite number  $n$  with  $n-1 = 2^s * r$ , for  $r$  odd, for which either

$$a^r \equiv 1 \pmod{n} \quad \text{or} \quad a^{2^j r} \equiv -1 \pmod{n}.$$

Let  $n$  be an odd integer, and let  $n-1 = 2^s * r$  where  $r$  is odd. Let  $a$  be any integer,  $1 \leq a \leq n-1$  such that  $\gcd(a,n) = 1$ . Then if  $a^r \equiv 1 \pmod{n}$  or  $a^{2^j r} \equiv -1 \pmod{n}$  for some  $j$ ,  $0 \leq j \leq s-1$  then  $n$  is prime.

Let  $n$  be an odd composite integer and let  $n-1 = 2^s * r$  where  $r$  is odd. Let  $a$  be an integer in the interval  $[1, n-1]$ , then

- (i) If  $a^r \not\equiv 1 \pmod{n}$  and if  $a^{2^j r} \not\equiv -1 \pmod{n}$  for all  $j$ ,  $0 \leq j \leq s-1$ , then  $a$  is called a **strong witness** (to compositeness) for  $n$ .
- (ii) Otherwise, if either  $a^r \equiv 1 \pmod{n}$  or  $a^{2^j r} \equiv -1 \pmod{n}$  for some  $j$ ,  $0 \leq j \leq s-1$ , then  $n$  is said to be a **strong pseudoprime to the base a** [2]. The integer  $a$  is called a **strong liar** (to primality) for  $n$ .

Note : --- If  $n$  is an odd composite integer, then at most 1/4 of all the numbers  $a$ ,  $1 \leq a \leq n-1$  are strong liars for  $n$ . In fact, if  $n \leq 9$ , the number of strong liars for  $n$  is at most  $\phi(n)/4$ .

The test : First find an  $r$  and  $s$  such that  $n-1 = 2^s * r$ . Choose a random integer  $a$ ,  $2 \leq a \leq n-2$ . Compute  $y = a^r \pmod{n}$ . Then for each  $j$ ,  $1 \leq j \leq s-1$  we check whether  $y \neq 1$  and  $y \neq n-1$ , updating the value of  $y$  as  $y = y^2 \pmod{n}$ . If any of these two inequalities holds then  $n$  is composite. Otherwise, as in the previous case, for a security parameter  $t$ , continue checking until the probability of error is considerably reduced. If it so happens we can reasonably conclude that  $n$  is prime.

Note :---For any odd composite integer  $n$ , the probability that this test declares  $n$  to be “prime” is less than  $(1/4)^t$ .

### 3.2.1.4 Comparison of the two techniques

If  $a$  is a *strong liar* for  $n$ , then it is also an *Euler liar* for  $n$ . Thus, from a correctness point of view, the Miller-Rabin test is never worse than the Fermat test [2]. What remains is a comparison of the computational costs. While the Miller-Rabin test may appear more complex, it actually requires, at worst, the same amount of computation as Fermat’s test in terms of modular multiplications; thus the Miller-Rabin test is better than Fermat’s test in all regards. At worst, the sequence of computations in Miller-Rabin test requires the equivalent of computing  $a^{(n-1)/2} \bmod n$ .

### 3.2.2 True Primality tests

The true primality tests are methods by which positive integers can be *proven* to be prime, and are often referred to as *primality proving algorithms*. These primality tests are generally more computationally intensive than the probabilistic primality tests. Here we define one of the more common true primality tests:

#### 3.2.2.1 Jacobi sum test

The basic idea is to test a set of congruences which are analogues of Fermat’s theorem. The running time of the Jacobi sum test for determining the primality of an integer  $n$  is  $O((\ln n)^{e \ln \ln \ln n})$  bit operations for some constant  $e$ . This is “almost” a polynomial-time algorithm since the exponent  $\ln \ln \ln n$  acts like a constant for the range of values for  $n$  of interest. For example, if  $n \leq 2^{512}$ , then  $\ln \ln \ln n < 1.78$ .

The version of the Jacobi sum primality test used in practice is a randomized

algorithm which terminates within  $O(k(\ln n)^{e \ln \ln \ln n})$  steps with probability at least  $1 - (1/2)^k$  for every  $k \geq 1$ , and always gives a correct answer [2]. The test is, indeed, practical in the sense that the primality of numbers that are several hundred decimal digits long can be handled in just a few minutes on a computer. However, the test is not as easy to program as the probabilistic Miller-Rabin test and the resulting code is not as compact.

### 3.2.3 Comparison between probabilistic and true primality tests

We have found from the analysis of primality tests that the two types of tests although related in the sense of being based on the properties of primes have the following discrepancies ----

1. Most so-called probabilistic primality tests are absolutely correct when they declare candidates to be composite, but do not provide a mathematical proof that  $n$  is prime in the case when such a number is declared to be “probably” so. For this reason, such tests are more properly called **compositeness tests** than probabilistic primality tests. The probability of error in the probabilistic primality tests is at most  $(1/2)^t$ . True primality tests, allow one to conclude with mathematical certainty that a number is prime. The Jacobi Sum test is always correct.
2. Probabilistic tests generally require less computational resources eg. Miller Rabin test requires  $a^{(n-1)/2} \bmod n$  computations as compared to true tests eg. The Jacobi sum test requires  $O(k(\ln n)^{e \ln \ln \ln n})$  steps .
3. True tests when implemented are highly structured and require more time and space as compared to probabilistic tests.

#### 4. Some practical aspects

Consider as an example the RSA cryptosystem. Here a modulus of the form  $n=p*q$  is used, where  $p$  and  $q$  are distinct odd primes. Some points to remember while choosing the primes to be used here are as follows: ----

1. The primes  $p$  and  $q$  must be of sufficient size that factorization of their product is beyond computational reach.
2. They should be random primes in the sense that they should be chosen as a function of a random input through a process defining a pool of candidates of sufficient cardinality that an exhaustive attack is infeasible.
3. The resulting primes must also be of a pre-determined bitlength, to meet system specifications.

The discovery of the RSA cryptosystem led to the consideration of several additional constraints on the choice of  $p$  and  $q$  which are necessary to ensure the resulting RSA system safe from cryptanalytic attack, and the notion of strong primes was defined. However, it is now believed that strong primes offer little protection beyond that offered by random primes, since randomly selected primes of the sizes typically used in RSA moduli today will satisfy the constraints with high probability. On the other hand, they are no less secure, and require only minimal additional running time to compute; thus, there is little real additional cost in using them.

#### 5. Conclusion

The prime numbers play very important role in cryptography. Still none could find out its alternative in this regard and hopefully those days are not very far ahead

that scientists or mathematicians will be able to find its alternative. So, it is still a matter of research that how the prime number can be used more effectively which is making the prime numbers more permanent in the world of cryptography. In our paper we tried to find out the feature and fact that positively help out the cryptographer to have a better use of prime numbers. Our analysis of the primality tests has revealed that although true primality tests are more accurate they are less efficient in terms of computation resources. On the other hand the probabilistic primality tests are computationally more feasible though lacking little in accuracy. The accuracy requirements and resources available to a cryptographer will determine the primality tests used.

#### References

- [1] Sanna Jaalinoja, Aspects of Cryptography and Internet Security.
- [2] A. Mennezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, , CRC Press, 1996.
- [3] Neal R Wagner, *The Laws of Cryptography with Java Code*
- [4] R.L.Rivest, A.Shamir and L.Adleman, *A Method for obtaining Digital Signatures and Public Key Cryptosystems*
- [5] Brian Raiter, *Prime Number Hide-and-Seek : How the RSA Cipher Works*
- [6] William Stallings, *Cryptography and Network Security*
- [7] H. Tietze, *Famous Problems of Mathematic* ,
- [8] P.Ribenboim, *The new book of prime number records*