# A Flexible and Efficient Algorithm for Generating Prime Numbers using Biometric Identity (Bio-PNGA)

B. Indrani, PhD
Assistant Professor,
Department of Computer Science,
Directorate of Distance Education
Madurai Kamaraj University
Madurai-21

M. Karthigai Veni
Assistant Professor,
Department of Computer Applications,
Yadava College,
Govindarajan Campus,
Tiruppalai, Madurai-14

## ABSTRACT
A biometric based security system provides best on both authentication and confidentiality for public shared secret information. Enormous numbers of papers have been published by the researchers in this field. The generation of prime numbers plays the most important role in the public-key schemes, essentially as a major primitive needed for the creation of key pairs or as a computation stage appearing during various cryptographic setups. Most of the researchers have been made strong mathematical studies on primality testing and an observed progressive increase of cryptographic usages, prime number generation algorithms. Still not quite investigated and most of the real-life implementations are providing poor performance. Most of the common prime number generators typically output n-bit prime in heuristic average complexity$O(n^4)$ or $O(n^4/\log n)$.

In this paper, we have proposed A Flexible and Efficient *BioPNGA* algorithm for generating the prime number for Public Key Infrastructure (PKI) by using biometric identity (example finger print, iris, face, DNA,… etc). The proposed scheme capturers the biometric identity image from the corresponding user and this image will be used as a seed value for generating the prime number. The proposed scheme generates set of prime numbers and these prime numbers can be used in Public Key Authentication and Confidentiality for the corresponding user. The proposed scheme requires minimum computing cost for generating prime numbers and any size of prime numbers can be created by using this scheme. This scheme output n-bit prime number from biometric identity matrix $m \times n$ in a heuristic average complexity of $O(m \times n)$. This scheme is very suitable for power constrained biometric based security schemes.

## General Terms
Security, Algorithms

## Keywords
Biometric Identity, Prime numbers, Public Key Infrastructure (PKI).

## 1. INTRODUCTION
The prime numbers and the deterministic formulas used to find them, have garnered considerable attention from mathematicians, professionals and amateurs alike. A prime number is a positive integer, excluding 1, whose only divisors are 1 and itself. For example, 23 is a prime number as it can only be divided by 1 and 23. A number that is not prime is called a composite number. For over 150 years, mathematicians have attempted to expose a deterministic formula to identify prime numbers. If such a formula existed, all numbers could be factored efficiently by using computers. Paradoxically, much of electronic data today is encrypted by taking advantage of the fact that it is difficult and time consuming for a computer program to factor a large composite number.

Traditional prime number generation algorithms asymptotically require $O(n^4)$ or $O(n^4/\log n)$bit operations where n is the bit-length of the expected prime number. This complexity may even become of the order of $O(n^5/(\log n)^2)$ in the case of constrained primes, such as safe or quasi-safe primes for instance. These asymptotic behaviors, according to experience, seem impossible to improve significantly

"*Biometrics is the science of establishing the identity of an individual based on physical, chemical or behavioral attributes of the person*" [15]. Due to the distinctive nature of biometric traits [16] and the non-repudiation it offers [17], biometry is frequently used to enhance the overall security of the system in which it is implemented: the authentication system or the biometric cryptosystem. Biometric authentication is the process of validating the uniqueness of individuals according to their physiological or behavioral qualities [18]. Physiological qualities, such as a fingerprint, an iris or a face, refer to something that an individual is. Behavioral qualities, such as speech, signature and keystroke dynamics refer to something that an individual can do. Biggio [19] proposed a generic modular biometric authentication system and the steps as explained as follows. A user who wants to access some resources provides his identity. The sensor acquires the biometric sample of the user. Features are extracted from the sample and a similarity score is calculated between the provided biometric sample and the one stored in the biometric template database corresponding to the provided user identity. The similarity score is compared with the threshold and the user is identified as a genuine user or a fake. According to this decision, the user is allows to access the resources.

There are several advantages of biometric authentication compare to traditional authentication methods, such as difficulties in stealing, sharing and reproduction of biometric samples, tolerance to brute force attacks, and non-repudiation [20].

There are two types of biometric systems: *A unimodal*, which employs a single biometric sample acquired from the user, and *a multimodal*, which employs two or more biometric samples, e.g. an iris and a fingerprint. *Multimodal systems* overcome some drawbacks of unimodal systems, such as large false rejection rates (FRR) and unacceptable false acceptance rates (FAR). Additional information provided to the classifier increases the recognition accuracy and decreases error rates,

while the identity proof is strengthened as data is acquired from different sources [21]. When compared to unimodal, multimodal systems are less prone to spoof attacks [22] and carefully crafted attacks targeted towards modular biometric authentication systems (replaying old data, feature extractor overriding, stored template modification, communication channel interception and providing synthetic vectors to the matching module) [23, 24].

The basis of multimodal biometric authentication systems is the information fusion. The decision level fusion [25] is the initial approach to information fusion in multimodal biometric authentication systems. This approach is based on majority vote scheme that is used to combine classification results from different modalities and make the final decision [26]. At the matching score level [27], the system calculates similarity scores between the sample and the corresponding template for each modality and combines them to verify the identity of an individual. At the feature level, feature vectors extracted from different modalities are integrated into a new vector that represents the identity of the individual [28].

# 2. RELATED WORK

There are several ways in which we could assess the quality of a random prime generation algorithm, based on its speed (time complexity), its accuracy (the probability that it outputs numbers that are in fact composite), its statistical properties (the regularity of the output distribution), and the number of bits of randomness it consumes to produce a prime number (as good randomness is crucial to key generation and not easy to come by [8]).

Many cryptographers have proposed faster prime generation algorithms [6, 7, 10, 11] or algorithms providing a proof that the generated numbers are indeed prime numbers [12, 13, 14].

A number of these works also prove lower bounds on the entropy of the distribution of prime numbers they generate, usual based on very strong conjectures on the regularity of prime numbers, such as the prime *r*-tuple conjecture of Hardy-Littlewood [9]. However, such bounds on the entropy do not ensure that the resulting distribution is statistically close to the uniform distribution: for example, they do not preclude the existence of efficient distinguishers from the uniform distribution, which can indeed be shown to exist in most cases

*a) Primality and Compositeness Tests*

A lot of studies on primality testing have been carried out for years, and can be found in the literature devoted to the subject [30]. Computationally, we may distinguish true primes and probable primes: the difference being the way these are generated. A probable prime is usually obtained through a compositeness test. Such a test declares that a number is composite with probability 1 or prime with some probability < 1. Hence repeatedly running the test gives more and more confidence in the generated (probable) prime. Typical examples of compositeness tests include Fermat test, Solovay-Strassen test [32], and Miller-Rabin test [31]. There also exist (true) primality tests, which declare a number prime with probability. Typical examples of exist primality tests includes Pocklington's test [33] and its elliptic curve analogue [34], the Jacobi sum test [35]. However, these tests are generally more expensive or intricate

***1. Prime sieves***

A prime sieve or prime number sieve is a fast type of algorithm for finding primes. There are many prime sieves. The simple sieve of Eratosthenes (250s BCE), the sieve of Sundaram (1934), the still faster but more complicated sieve of Atkin (2004), and various wheel sieves are most common.

A prime sieve works by creating a list of all integers up to a desired limit and progressively removing composite numbers. This is the most efficient way to obtain a large range of primes. To find individual primes, direct primality tests are more efficient. Furthermore, based on the sieve formalisms, some integer sequences (sequence A240673 in OEIS) are constructed which they also could be used for generating primes in certain intervals

***2. Sieve of Eratosthenes***

To find all the prime numbers less than or equal to a given integer *n* by Eratosthenes' method:

1. Create a list of consecutive integers from 2 through *n*: (2, 3, 4, ..., *n*).
2. Initially, let *p* equal 2, the smallest prime number.
3. Enumerate the multiples of *p* by counting to *n* from *2p* in increments of *p*, and mark them in the list (these will be 2*p*, 3*p*, 4*p*, ... ; the *p* itself should not be marked).
4. Find the first number greater than *p* in the list that is not marked. If there was no such number, stop. Otherwise, let *p* now equal this new number (which is the next prime), and repeat from step 3.

When the algorithm terminates, the numbers remaining not marked in the list are all the primes below *n*.

The main idea here is that every value given to *p* will be prime, because we have already marked all the multiples of the numbers less than *p*. Note that some of the numbers being marked may have already been marked earlier (e.g., 15 will be marked both for 3 and 5)

***3. Sieve of Sundaram***

The sieve of Sundaram is a simple deterministic algorithm for finding all prime numbers up to a specified integer. It was discovered by Indian mathematician S. P. Sundaram in 1934

*a.* Algorithm

Start with a list of the integers from 1 to *n*. From this list, remove all numbers of the form $i + j + 2ij$ where:

**1.** $i, j \in N, 1 \leq i \leq j$

**2.** $i + j + 2ij \leq n$

The remaining numbers are doubled and incremented by one, giving a list of the odd prime numbers (i.e., all primes except 2) below $2n + 2$. The sieve of Sundaram sieves out the composite numbers just as sieve of Eratosthenes does, but even numbers are not considered; the work of "crossing out" the multiples of 2 is done by the final double-and-increment step. Whenever Eratosthenes' method would cross out $k$ different multiples of a prime$2i + 1$ , Sundaram's method crosses out$i + j(2i + 1)$ for$1 \leq j \leq \lfloor k/2 \rfloor$

***4. Sieve of Atkin***

*Algorithm*

In the algorithm:

- All remainders are modulo-sixty remainders (divide the number by 60 and return the remainder).
- All numbers, including *x* and *y*, are positive integers.

- Flipping an entry in the sieve list means to change the marking (prime or nonprime) to the opposite marking.
- This results in numbers with an odd number of solutions to the corresponding equation being potentially prime (prime if they are also square free), and numbers with an even number of solutions being composite.

*b) Primality test*

A primality test is an algorithm for determining whether an input number is prime. Amongst other fields of mathematics, it is used for cryptography. Unlike integer factorization, primality tests do not generally give prime factors, only stating whether the input number is a prime or not. Factorization is thought to be a computationally difficult problem, whereas primality testing is comparatively easy. The running time is polynomial in the size of the input numbers. Some primality tests *prove* that a number is prime, while others like Miller–Rabin prove that a number is composite. Therefore, the latter might be called *compositeness tests* instead of primality tests

*1)* Simple methods

The simplest primality test is *trial division*: Given an input number *n*, check whether any prime integer *m* from 2 to $\sqrt{n}$ evenly divides *n* (the division leaves no remainder). If *n* is divisible by any *m* then *n* is composite, otherwise it is prime.

For example, we can do a trial division to test the primality of 100. Let's look at all the divisors of 100:

2, 4, 5, 10, 20, 25, 50

Here we see that the largest factor is 100/2 = 50. This is true for all *n*: all divisors are less than or equal to *n*/2. If we take a closer look at the divisors, we will see that some of them are redundant. If we write the list differently:

$$100 = 2 \times 50 = 4 \times 25 = 5 \times 20 = 10 \times 10 = 20 \times 5 = 25 \times 4 = 50 \times 2$$

Once we reach 10, which is √100, the divisors just flip around and repeat. Therefore, we can further eliminate testing divisors greater than √n. We can also eliminate all the even numbers greater than 2, since if an even number can divide *n*, so can 2

*2)* Heuristic tests

The Fermat test and the Fibonacci test are simple examples, and they are *very* effective when combined. John Selfridge has conjectured that if p is an odd number, and p ≡ ±2 (mod 5), then p will be prime if both of the following hold:

- $2^{p-1} \equiv 1 \pmod p$
- $f_{p+1} \equiv 0 \pmod p$,

where $f_k$ is the $k^{th}$ Fibonacci number. The first condition is the Fermat primality test using base 2. The Baillie-PSW primality test is another excellent heuristic, using the Lucas sequence in place of the Fibonacci sequence

*3)* Probabilistic tests

Probabilistic tests are more exact than heuristics in that they provide provable bounds on the probability of being fooled by a composite number. Many popular primality tests are probabilistic tests. These tests use, apart from the tested number n, some other numbers a which are chosen at random from some sample space, the usual randomized primality tests never report a prime number as composite, but it is possible for a composite number to be reported as prime.

The probability of error can be reduced by repeating the test with several independently chosen values of *a*; for two commonly used tests, for *any* composite *n* at least half the *a*'s detect *n*'s compositeness. So *k* repetitions reduce the error probability to at most $2^{-k}$, which can be made arbitrarily small by increasing *k*.

The basic structure of randomized primality tests is as follows:

1. Randomly pick a number *a*.
2. Check some equality (corresponding to the chosen test) involving *a* and the given number *n*. If the equality fails to hold true, then *n* is a composite number, *a* is known as a *witness* for the compositeness, and the test stops.
3. Repeat from step 1 until the required accuracy is achieved.

After one or more iterations, if *n* is not found to be a composite number, then it can be declared probably prime.

4) Fermat primality test

The simplest probabilistic primality test is the Fermat primality test (a compositeness test). It works as follows:

Given an integer *n*, choose some integer *a* coprime to *n* and calculate $a^{n-1}$ modulo *n*. If the result is different from 1, then *n* is composite. If it is 1, then *n* may or may not be prime.

If $a^{n-1}$ (modulo *n*) is 1 but *n* is not prime, then *n* is called a pseudoprime to base *a*. In practice, we observe that, if $a^{n-1}$ (modulo *n*) is 1, then *n* is usually prime. But here is a counterexample: if *n* = 341 and *a* = 2, then

$$2^{340} \equiv 1 (mod\ 341)$$

even though 341 = 11·31 is composite. In fact, 341 is the smallest pseudoprime base 2 [36].

There are only 21853 pseudoprimes base 2 that are less than $2.5\times10^{10}$ [36]. This means that, for *n* up to $2.5\times10^{10}$, if $2^{n-1}$ (modulo *n*) equals 1, then *n* is prime, unless *n* is one of these 21853 pseudoprimes.

5) Miller-Rabin and Solovay-Strassen primality test

The Miller–Rabin primality test and Solovay–Strassen primality test are more sophisticated variants which detect all composites. These are also compositeness tests. The Miller–Rabin primality test works as follows: Given an integer *n*, choose some positive integer $a < n$. Let $2^s d = n - 1$ where *d* is odd. If

$$a^d \not\equiv 1 (mod\ n)$$

and

$$a^{2rd} \not\equiv -1 \ (mod\ n) \text{ for all } 0 \le r \le s-1$$

then *n* is composite and *a* is a witness for the compositeness. Otherwise, *n* may or may not be prime. The Miller-Rabin test is a strong pseudoprime test [36].

The Solovay–Strassen primality test uses another equality: Given an odd number *n*, choose some integer $a < n$, if

$a^{(n-1)/2} \not\equiv \left(\frac{a}{n}\right) \ (mod\ n)$, where $\left(\frac{a}{n}\right)$ is the Jacobi symbol,

then *n* is composite and *a* is a witness for the compositeness. Otherwise, *n* may or may not be prime. The Solovay-Strassen test is an Euler pseudoprime test [36].

For each individual value of *a*, the Solovay-Strassen test is weaker than the Miller-Rabin test. For example, if *n* = 1905 and *a* = 2, then the Miller-Rabin test shows that *n* is composite, but the Solovay-Strassen test does not. This is because 1905 is an Euler pseudoprime base 2 but not a strong pseudoprime base 2.

6) Frobenius primality test

The Miller-Rabin and the Solovay-Strassen primality tests are simple and are much faster than other general primality tests. One method of improving efficiency further in some cases is the Frobenius pseudoprimality test; a round of this test takes about three times as long as a round of Miller–Rabin, but achieves a probability bound comparable to seven rounds of Miller–Rabin.

The Frobenius test is a generalization of the Lucas pseudoprime test. One can also combine a Miller-Rabin type test with a Lucas pseudoprime test to get a primality test that has no known counterexamples. That is, this combined test has no known composite *n* for which the test reports that *n* is probably prime. One such test is the Baillie-PSW primality test, several variations of which are described in [37].

7) Other tests

Leonard Adleman and Ming-Deh Huang presented an errorless (but expected polynomial-time) variant of the elliptic curve primality test. Unlike the other probabilistic tests, this algorithm produces a primality certificate, and thus can be used to prove that a number is prime [39]. The algorithm is prohibitively slow in practice.

If quantum computers were available, primality could be tested asymptotically faster than by using classical computers. A combination of Shor's algorithm, an integer factorization method, with the Pocklington primality test could solve the problem in $O(log^3 n \log\log n \; log \; log \log n)$[38].

## 3. PROPOSED BIOPNGA SCHEME

Traditional prime number generation algorithms asymptotically require $O(n^4)$ or $O(n^4/\log n)$ bit operations where n is the bit-length of the expected prime number. This complexity may even become of the order of $O(n^4/(\log n)^2)$ in the case of constrained primes, such as safe or quasi-safe primes for instance. These asymptotic behaviors (assuming that multiplications modulo $q$ are in $O(|q|)^2$. Theoretically, one could decrease this complexity by using multiplication algorithms such as Karatsuba in $O(|q|^{\log_2 3})$ or Schonhage-Strassen in $O(q \log|q| \log\log|q|))$, according to researchers view, seem impossible to improve significantly. In this paper, we have proposed a technique to generate a set of prime numbers from a biometric identity of individual users, which substantially reduce the computational cost and improve the security level of prime numbers generation.

The proposed techniques will work very efficiently for implementations on cryptographic smart cards for on-board RSA [10] (or other schemes) key generation. This technique may apply for any kind of biometric based security systems. Our motivation here is to help transferring this task from terminals to smart-cards themselves in the near future for more confidence, security, and compliance with network-scaled distributed protocols that include smart-cards, such as electronic cash or mobile commerce. This concept is new to the research world and this will provide more security for selecting the prime numbers in PKI.

**Biometric Based Prime Number Generation Algorithm (BioPNGA)**

The proposed scheme has two phases:

1. In the first phase, user biometric identity is acquired from the user and the image is converted into gray scale image. The user can select a area from image based on the size of prime number. The selected area converted into binary matrix will be in 0's and 1's.
2. In second phase, the selected size of image value or size is taken from the binary image file (binary image matrix). The positive number is calculated by using this binary value from every row wise. This positive number is pass to the Primality Test phase for checking the Prime number. For example if we want to find a prime number with the size of 32 bits then we have to take a 32x32 bit matrix from the binary matrix.

The proposed architecture is given below,

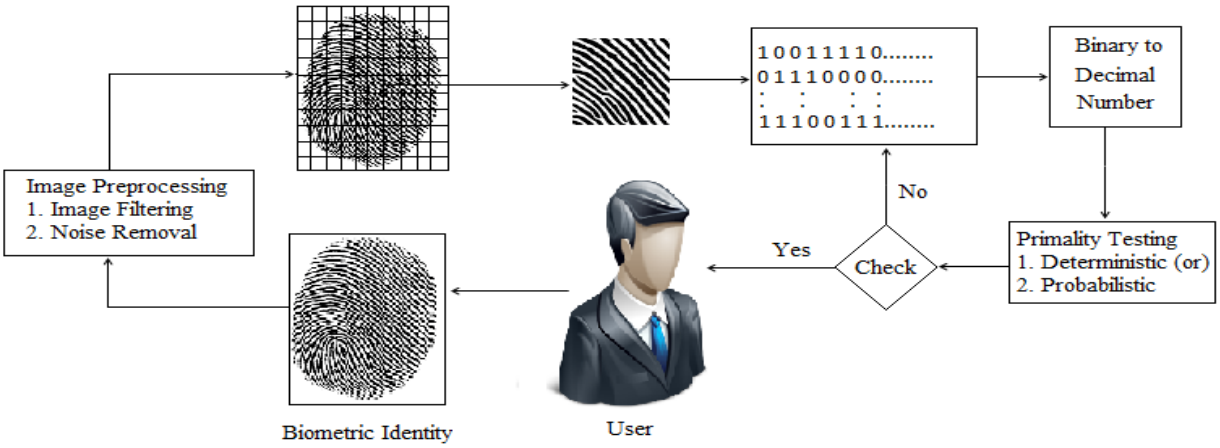



**Figure 1: Proposed BioNPGA Architecture**

***a. Biometric Identity Acquiring Phase***

In this phase, we have three sub-modules and module working principles are given below,

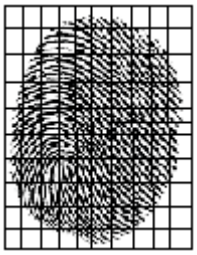
1) *Biometric Identity capturing module*

In this module, the new user or the register user $U_A$ can enroll their biometric identity $BI_A$. This $BI_A$ is used as a seed value for creating the Prime number

2) *Image Preprocessing and region selection module*

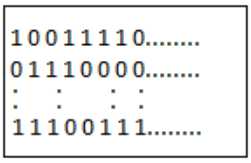
In this module, two main steps are carried out. First one is to apply the preprocessing on acquired biometric image from the user. In this step, first we apply the Image Filtering methods on acquired image and apply the Noise removal techniques. The preprocessed image is feed into the second step called region selection. In the region selection, a small portion of preprocessed image is taken as a seed value for generating Prime number. The selected region $BI_A[l,k]$ from $BI_A$.

**Figure 2: Biometric $BI_A$ for user $U_A$**



(a) Preprocessed Image $BI_A$

(b) Selected Portion of Image $BI_A[l,k]$ for Positive number calculation $PN_l$




(c) Binary matrix for the selected portion of Image $BinaryMax_{i,j}$

**Figure 3: Biometric Identity Acquiring Phase**

3) *Number Generation Module*

In this module, we have convert selected region image into binary value or binary matrix, for example 32x32, 64x64, 128x128, 256x256, likewise. Based on the selected region size the positive number will be generated. This number will be passing to the next module for applying the Primality test or Prime test

***b. Primality Testing Phase***

In this phase, apply the Primality Test for each positive number $PN_i$ from the row wise in the Binary Matrix $BinaryMax_{i,j}$. The Binary Matrix $BinaryMax_{i,j}$ is represented like in Figure 3 (c) and every row a positive integer value is calculated for the corresponding binary value from each row. The following algorithm illustrate the basic steps in the proposed method

**Algorithm for the Proposed Scheme**

*BioPNGAlgorithm(**BinaryMax**[ ], **Row**, **Column**)*
*Begin*
*For i=1 to Row*
*Begin*
$\boldsymbol{N_i = Calculate\ BinarytoDecimal(BinaryMax[i])}$ *//Calculate Decimal equivalent of Binary String* $\boldsymbol{BinaryMax[i]}$
*If(PrimalityTest(**$N_i$**)=TRUE)* *// Apply Primality Test on* $\boldsymbol{N_i}$
*Maintain it in a corresponding entry in Table*
*Else*
*Goto Next Entry*
*End*
*End*

# 4. IMPLEMENTATION AND RESULT DISCUSSION

The proposed scheme is implemented by using MATLAB version R2009a. The First phase of the proposed scheme is implemented by using basic standard functions in MATLAB for Image capturing, Image filtering, Noise removal, and other preprocessing functions. This phase produces good result in Image Quality. The preprocessed image is converted into small cell like given in Figure 3 (a) based on the required size of Prime number. For example, if a user needs 32 bits size prime number then the preprocessed image is converted into 32x32 block of small images. The one small block selected by the user and it is converted into binary matrix as shown in Figure 3 (c). A common table is maintained for group the binary number list, positive number list and prime number list as shown in figure 4.

The traditional prime number generation algorithms asymptotically require $O(n^4)$ or $O(n^4/\log n)$ bit operations where n is the bit-length of the expected prime number. The proposed Algorithm asymptotically require only $O(n)$ and only primality checking needs additional computational cost. The following table provides the time taken for compute the prime list for different size of blocks. For the comparison, we have considered the size of 32x32, 64x64, 128x128, and 256x256. The time taken for computing the prime numbers from the given binary matrix is very minimum and it is very easy way to generate the prime numbers for Public Key Infrastructure (PKI). The proposed scheme requires minimum time $\approx 10\ ms$ for 32 bit prime generation.

CASIA Fingerprint Image Database Version 5.0 (or CASIA-FingerprintV5) contains 20,000 fingerprint images of 500 subjects. The fingerprint images of CASIA-FingerprintV5 were captured using URU4000 fingerprint sensor in one session. The volunteers of CASIA-FingerprintV5 include graduate students, workers, waiters, etc. Each volunteer contributed 40 fingerprint images of his eight fingers (left and right thumb/second/third/fourth finger), i.e. 5 images per finger. The volunteers were asked to rotate their fingers with various levels of pressure to generate significant intra-class variations. All fingerprint images are 8 bit gray-level BMP files and the image resolution is 328x356

The Finger Print image subset used in our experiments consists of 250 image samples a resolution of 328x356 pixels. The optimal number of cells used in the non-invertible transform is selected as a compromise between the template security to brute force attacks and Equal Error Rate (EER). These parameters were found to provide high local entropy and optimum encoding on CASIA Finger Print database [40].

| i, j | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | +ve Number | Primality Test |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 5915587277 | Prime |
| 2 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 5915587275 | Not a Prime |
| 3 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1500450272 | Not a Prime |
| 4 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1500450271 | Prime |
| 5 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1500481942 | Not a Prime |
| 6 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 3267000056 | Not a Prime |
| 7 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 3267000040 | Not a Prime |
| 8 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 3267000024 | Not a Prime |
| 9 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 3267000013 | Prime |
| 10 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 5754853098 | Not a Prime |
| 11 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 5754853343 | Prime |
| 12 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 5754853208 | Not a Prime |
| 13 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 5754852058 | Not a Prime |
| 14 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 4093011218 | Not a Prime |
| 15 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 4093082899 | Prime |
| 16 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 4093075730 | Not a Prime |
| 17 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 4093083098 | Not a Prime |
| 18 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 3628273133 | Prime |
| 19 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 3628272808 | Not a Prime |
| 20 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 3628273568 | Not a Prime |
| 21 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 3628273642 | Not a Prime |
| 22 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 2860486313 | Prime |
| 23 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 2860490728 | Not a Prime |
| 24 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 2860494584 | Not a Prime |
| 25 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 2860502956 | Not a Prime |
| 26 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 5463458053 | Prime |
| 27 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 5463595328 | Not a Prime |
| 28 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5463467392 | Not a Prime |
| 29 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 3367900313 | Prime |
| 30 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 3367984280 | Not a Prime |
| 31 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 3367933592 | Not a Prime |
| 32 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 3368017048 | Not a Prime |
| 33 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 3367966360 | Not a Prime |

**Figure 4**: **Selected portion of 33x33 Binary Matrix $BinaryMax_{i,j}$ from Biometric Identity and the List of Prime Numbers**

**Table 1**: **Time taken for Prime Number generation and Success rate to find minimum one prime number per block**

| Block Size in Bits | Time taken for Number generation | Time taken for Primality Test | Success rate to find minimum one prime number per block |
|---|---|---|---|
| **32x32** | $\approx 10\ ms$ | $\approx 6\ ms$ | 67% |
| **64x64** | $\approx 15\ ms$ | $\approx 6\ ms$ | 86% |
| **128x128** | $\approx 27\ ms$ | $\approx 14\ ms$ | 92% |
| **256x256** | $\approx 32\ ms$ | $\approx 20\ ms$ | 93% |

The proposed scheme some time fails to find the prime number in a selected block for the biometric identity. The failure situation or case is very minimum when high bit size is used for prime number generation. The table 1 shows the performance evaluation for our proposed scheme

## 5. CONCLUSION

The generation of prime numbers plays the most important role in the public-key schemes, essentially as a major primitive needed for the creation of key pairs or as a computation stage appearing during various cryptographic setups. In this paper, we have proposed an algorithm for finding the prime number for Public Key Infrastructure (PKI) by using biometric identity (example finger print, iris, face, DNA,… etc). The proposed scheme requires minimum computing cost for generating prime numbers and any size of prime numbers can be created by using this scheme. This scheme output n-bit prime number from biometric identity matrix $m \times n$ in a heuristic average complexity of$O(m \times n)$. This scheme is very suitable for power constrained biometric based security schemes

## 6. REFERENCES

[1] A.O.L. Atkin, D.J. Bernstein, *Prime sieves using binary quadratic forms*, Math. Comp. 73 (2004), pp. 1023-1030.

[2] Carl Pomerance; John L. Selfridge; Samuel S. Wagstaff, Jr. (July 1980). "The pseudoprimes to $25 \cdot 10^9$". Mathematics of Computation, 35 (151), pp. 1003–1026.

[3] Robert Baillie; Samuel S. Wagstaff, Jr. (October 1980). "Lucas Pseudoprimes" (PDF). Mathematics of Computation **35** (152): 1391–1417

[4] Adleman, Leonard M.; Huang, Ming-Deh (1992). Primality testing and Abelian varieties over finite field. Lecture notes in mathematics 1512. Springer-Verlag. ISBN 3-540-55308-8

[5] Chau, H. F.; Lo, H.-K. (1995). "Primality Test Via Quantum Factorization". arXiv:quant-ph/9508005

[6] J. Brandt and I. Damgard. “On generation of probable primes by incremental search”, In E. F. Brickell, editor, *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 358–370. Springer, 1992.

[7] J. Brandt, I. Damgard, and P. Landrock. “Speeding up prime number generation”, In H. Imai, R. L. Rivest, and T. Matsumoto, editors, *ASIACRYPT*, volume 739 of *Lecture Notes in Computer Science*, pp. 440–449. Springer, 1991

[8] D. Eastlake 3rd, J. Schiller, and S. Crocker. “Randomness Requirements for Security”, RFC 4086 (Best Current Practice), June 2005

[9] G. H. Hardy and J. E. Littlewood. “Some problems of ‘partitio numerorum’: III. on the expression of a number as a sum of primes”, 44, pp. 1–70, 1922

[10] M. Joye and P. Paillier. “Fast generation of prime numbers on portable devices: An update”, In L. Goubin and M. Matsui, editors, *CHES*, volume 4249 of *Lecture Notes in Computer Science*, pp. 160–173. Springer, 2006.

[11] M. Joye, P. Paillier, and S. Vaudenay. “Efficient generation of prime numbers”, In Cetin Kaya Koc and C. Paar, editors, *CHES*, volume 1965 of *Lecture Notes in Computer Science*, pp. 340–354. Springer, 2000

[12] U. M. Maurer. “Fast generation of secure RSA-moduli with almost maximal diversity”, In *EUROCRYPT*, pp. 636–647, 1989.

[13] U. M. Maurer. “Fast generation of prime numbers and secure public-key cryptographic parameters”, *J. Cryptology*, 8(3), pp.123–155, 1995

[14] P. Mihăilescu. “Fast generation of provable primes using search in arithmetic progressions”, In Y. Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pp. 282–293. Springer, 1994.

[15] A. K. Jain, A. Ross: Introduction to Biometrics. In “Handbook of Biometrics”, A. Jain et al. (Eds), Springer, 2008

[16] Y. C. Feng, P. C. Yuen, A. K. Jain: “A Hybrid Approach for Face Template Protection”, In Proceedings of SPIE Conference of Biometric Technology for Human Identification, Orlando, USA, Vol. 6944, pp. 325, 2008

[17] P. Balakumar, R. Venkatesan: “A Survey on Biometrics-based Cryptographic Key Generation Schemes”, International Journal of Computer Science and Information Technology & Security, Vol. 2, No. 1, pp. 80-85, 2012

[18] A. K. Jain, A. Ross, S. Prabhakar: “An Introduction to Biometric Recognition”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 14, pp. 4-20, 2004

[19] B. Biggio: “Adversarial Pattern Classification. Doctoral dissertation”, University of Cagliari, Cagliari, Italy, 2010

[20] A. Jagadeesan, K. Duraiswamy: “Secured Cryptographic Key Generation From Multimodal Biometrics: Feature

Level Fusion of Fingerprint and Iris", International Journal of Computer Science and Information Security, Vol. 7, No. 2, pp. 28-37, 2010

[21] L. Hong, A. K. Jain, S. Pankanti: "Can Multibiometrics Improve Performance?", In Proceedings of IEEE Workshop on Automatic Identification Advanced Technologies, pp. 59-64, NJ, USA, 1999

[22] A. K. Jain, A. Ross: Multi-Biometric Systems: "Special Issue on Multimodal Interfaces that Flex, Adapt, and Persist", Communications of the ACM, Vol. 47, No. 1, pp. 34-40, 2004

[23] A. K. Jain, K. Nandakumar, A. Nagar: "Biometric Template Security", EURASIP J. Adv. Signal Process, 2008:1-17, 2008

[24] J. Galbally, C. McCool, J. Fierrez, S. Marcel, J. Ortega-Garcia. "On the Vulnerability of Face Verification Systems to Hill-Climbing Attacks", Pattern Recogn., 43(3) pp. 1027-1038, 2010

[25] S. Prabhakar, A. Jain: "Decision-Level Fusion in Fingerprint Verification", Pattern Recognition, Vol. 35, pp. 861-874, 2002

[26] Z. Wang, E. Wang, S. Wang, Q. Ding: "Multimodal Biometric System Using Face-Iris Fusion Feature. Journal of Computers", Vol. 6, No. 5, pp. 931-938, 2011

[27] K. Toh, J. Kim, S. Lee: "Biometric Scores Fusion Based on Total Error Rate Minimization",. Pattern Recognition, Vol. 41, pp. 1066-1082, 2008

[28] A. Ross, R. Govindarajan: Feature Level Fusion in Biometric Systems. In proceedings of Biometric Consortium Conference, September 2004

[29] S. Adamović, M. Milosavljević: Information Analysis of Iris Biometrics for the Needs of Cryptology Key Extraction. Serbian Journal of Electrical Engineering, Vol. 10, No. 1, pp. 1-12, 2003

[30] C. Couvreur and J.-J. Quisquater. An introduction to fast generation of large prime numbers. Philips Journal of Research, vol. 37, pp. 231-264, 1982

[31] D.E. Knuth. The Art of Computer Programming - Seminumerical Algorithms, vol. 2, Addison-Wesley, 2nd ed., 1981

[32] R. Solovay and V. Strassen. A fast Monte-Carlo test for primality. SIAM Journal on Computing, vol. 6, pp. 84-85, 1977

[33] H.C. Pocklington. The determination of the prime or composite nature of large numbers by Fermat's theorem. Proc. of the Cambridge Philosophical Society, vol. 18, pp. 29-30, 1914

[34] A.O.L. Atkin and F. Morain. Elliptic curves and primality proving. Mathematics of Computation, vol. 61, pp. 29-68, 1993

[35] W. Bosma and M.-P. van der Hulst. Faster primality testing. In Advances in Cryptology-CRYPTO'89, vol. 435 of Lecture Notes in Computer Science, pp. 652-656, Springer-Verlag, 1990

[36] Carl Pomerance; John L. Selfridge; Samuel S. Wagstaff, Jr. (July 1980). "The pseudoprimes to $25 \cdot 10^9$". Mathematics of Computation 35 (151): 1003–1026. doi:10.1090/S0025-5718-1980-0572872-7

[37] Robert Baillie; Samuel S. Wagstaff, Jr. (October 1980). "Lucas Pseudoprimes", Mathematics of Computation 35 (152): 1391–1417

[38] Chau, H. F.; Lo, H.-K. (1995). "Primality Test Via Quantum Factorization". arXiv:quant-ph/9508005

[39] Adleman, Leonard M.; Huang, Ming-Deh (1992). Primality testing and Abelian varieties over finite field. Lecture notes in mathematics 1512. Springer-Verlag.

[40] http://biometrics.idealtest.org/dbDetailForUser.do?id=7