International Journal of Computer Science Engineering
and Information Technology Research (IJCSEITR)
ISSN(P): 2249-6831; ISSN(E): 2249-7943
Vol. 5, Issue 1, Feb 2015, 117-126
TJPRC Pvt. Ltd.

A FAST FACTORIZATION ALGORITHM FOR COMPOSITE INTEGERS OF THE FORM N = P X Q, WHERE P AND Q ARE PRIMES AND N MOD 6= + 1

NOURELDIEN A. NOURELDIEN & EBTISAM ABAKER

Department of Computer Science, University of Science and Technology, Omdurman, Sudan

ABSTRACT

The factorization of integers of the form N = p x q, where p and q are primes is of special interest in cryptography and algebra. It is well known facts that a prime number P is in one o two series; P mod 6 = ±1, and each prime P generate a series of composite numbers of the form N mod 6 = ±1.

Based on the concept of Integer Absolute Position, which define the position of an integer N, where N mod 6 = ±1, within its series as AP (N) = N div 6, we develop a fast factorization algorithm for composite integers of the form N = p x q, where p and q are primes and N mod 6= + 1.

In comparing the developed algorithm with the well known factorization algorithm Pollard Rho, the developed algorithm out performs Pollard Rho under certain conditions, namely, when the value of the smallest factor p/q is small or when the values of p and q are relatively close.

KEYWORDS: Form N = P X Q, Where P and Q are Primes and N MOD 6= + 1, Cryptography and Algebra

1. INTRODUCTION

Fast factoring algorithms are important for ensuring that sensitive data is protected in electronic transmissions [1]. Also factoring is an important process in algebra which is used to simplify expressions, simplify fractions, and solve equations.

Today, factoring algorithms are tending to be of great interest in cryptography due to the fact that some encryption algorithms get their cryptographic strength from the difficulty of factoring large Integers, such as the RSA encryption algorithm.

If it were possible to factor products of large prime numbers quickly, RSA algorithm would be insecure, and consequently the protocols that relies on the security of the RSA algorithm such as SSL [2], which is used to secure TCP/IP connections over the web.

There are many proposed factorization algorithm that attempt to improve factorization performance. In fact, most of the algorithms that exist today run on the order of en, where e is Euler's number [3]. Generally speaking, the run time of factoring algorithms is either depends on the size of N, the number being factored or on the size of factor found, F [4].

Examples of algorithms that their run time depends on the size of N includes; Lehman's algorithm [5] which has a rigorous worst-case run time bound O(N 1/3), Shanks's SQUFOF algorithm [6], which has expected run time O(N1/4). Shanks's Class Group algorithm [7,8] which has run time O(N 1/5+€) on the assumption of the Generalized Riemann Hypothesis, the Continued Fraction algorithm [9] and the Multiple Polynomial Quadratic Sieve algorithm [4,7,14], which under plausible assumptions have expected run time O(exp(c(log N log log N) 1/2)), where c is a constant, and the fastest known general-purpose factoring algorithm, the General Number Field Sieve (GNFS), which in asymptotic notation takes S = O (exp((64/9 n) 1/3 (log n) 2/3)) steps to factor an integer with n decimal digits. The running time of the algorithm is bounded below by functions polynomial in n and bounded above by functions exponential in n .

On the other hand, algorithms that their run time depends mainly on the size of the factor found f, includes; the trial division algorithm, which has run time O(f · (log N)2). The Pollard rho algorithm [10], which has expected run time O(f1/2·(logN)2). Lenstra's "Elliptic Curve Method" (ECM) [11, 12] which has expected run time O(exp(c(log f log log f) ½. log(N)2), where c is a constant.

This paper contributes to the efforts of developing fast factoring algorithms. The paper proposes a deterministic factoring algorithms for integers that takes the RSA modulus form, that is, integers of the form N=p X q, where p and q are primes, and N mod 6 = +1.

The proposed algorithm is developed using the concept of integer absolute position defined in [13], and its Java implementation is compared to some known fast factorization algorithms. The experimental tests show that the developed algorithm is competent when p and q are relatively closed.

The rest of this paper is organized as follows; in section 2 a preliminary theory about prime series and the nature of the composite numbers in the form N mod 6 =+1 is stated and the composite absolute position matrices of composites of the form N mod 6=+1 are developed. In section 3 the proposed algorithm is presented. Section 4 is dedicated for experimental testing and results. Conclusions are given in section 5.

2. PRELIMINARY THEORY

It is a known fact that a prime number P satisfies either P mod 6 = 1 or P mod 6 = -1, we call the sets {N: N mod 6 = - 1, N ϵ Z+} and {N: N mod 6 = 1, N ϵ Z+}, R5 and R1 prime series respectively. Thus R5 = {5, 11, 17, 23, 29, 35, 41, 47, 53, 59, 65, 71, 77, 83, 89, 95, 101, 107,….} and R1 = {1, 7, 13, 19, 25, 31, 37, 43, 49, 55, 61, 67, 73, 79, 85, 91, 97,……}.

A composite number N is in R1 if it is a result of multiplying two integers that both belongs either to R1 or R5, i.e. N = p x q, ϵ R, when either p and q ϵ R1 or ϵ R5. We express this fact in the following theorem.

Theorem (1)

For any integer N ϵ R1, then N is either a prime or composite number.

- If N is a composite number that has exactly two prime factors, i.e. N= p x q, then both p, q are ϵ R1 or both p and q ϵ R5.
- If N is a composite number that has more than two prime factors, i.e. N= p1 x p2 x …pi, i≥3, then zero or an even number of pi's are ϵ R5, and the others ϵ R1.

Proof

(a) Suppose N ϵ R1 is a composite number that has two factors, then

N = pq → pq (mod 6) =1 → (((p mod 6) (q mod 6)) mod 6) =1

Since (1) holds only when both (p mod 6) = 1 and (q mod 6) = 1, therefore the two factor p and q ϵ R1, or when both (p mod 6) =5 and (q mod 6) = 5. Therefore the two factor p and q ϵ R5.

(b) Proof is directly follows from (a).

Based on theorem 1 and prime series definition, N=p x q, where p and q are primes is an entry in one of two labeled matrices A and B, shown below.

Matrix A: Space of Composite Numbers in R1 that results from R1*R1

R1	7	13	19	25	31	37	43	49	55	61	67	73	79	85	R1
	49	91	133	175	217	259	301		…						7
	91	169	247	325	403	481	559		…						13
	133	247	361	475	589	703	817		…						19
	175	325	475	625	775	925	1075		…						25
	217	403	589	775	961	1147	1333		…						31
	:	:	:	:	:	:	:		:						:

Matrix B: Space of Composite Numbers in R1 that Results from R5*R5

R5	5	11	17	23	29	35	41	47	53	59	65	71	77	83	R5
	25	55	85	115	145	175	205	…							5
	55	121	187	253	319	385	451	…							11
	85	187	289	391	493	595	697	…							17
	115	253	391	529	667	805	943	…							23
	145	319	493	667	841	1015	1189	…							29
	:	:	:	:	:	:	:	:							:

The position of each number matrix A or B is called Absolute Position (AP), and for a given integer N, this position is given by: AP (N) = N div 6 [13]. Using absolute position concept, matrix C and D, generated from matrices A and B respectively, defines the Absolute Positions for composite numbers in R1.

Matrix C: Composite Number's Absolute Positions in R1 that result from R1 x R1

P	7	13	19	25	31	37	43	49	Q's AP	Q
P's AP	1	2	3	4	5	6	7	.		
	8	15	22	29	36	43	50	.	1	7
	15	28	41	54	67	80	93	.	2	13
	22	41	60	79	98	117	136	.	3	19
	29	54	79	104	129	154	179	.	4	25
	36	67	98	129	160	191	222	.	5	31
	43	80	117	154	191	228	265	.	6	37
	50	93	136	179	222	265	308	.	7	43
		:	:	:	:	:	:	:	AP	:

Matrix D: Composite Number's Absolute Positions in R1 that Result from R5 x R5

P	11	17	23	29	35	41	.	Q's AP	Q
P's AP	1	2	3	4	5	6	.		
	20	31	42	53	64	75	.	1	11
	31	48	65	82	99	116	.	2	17
	42	65	88	111	134	157	.	3	23
	53	82	111	140	169	198	.	4	29
	:	:	:	:	:	:	:	:	:

If we eliminate in C and D the columns and rows that are generated by composite labels such as 25, 35, 49, 55, 85, 121,…etc, (shaded in matrixes C and D), then the remaining space contains absolute positions of composites that are generated by exactly two prime factors, P and Q.

For matrix C, the position of any entry in the matrix space can be denoted by Cij , where i and j denotes absolute positions of the prime factors P and Q that compose Cij; For example C63 is the absolute position of a composite whose factors have absolute positions 6 and 3.(since Cij = Cji there is no meaning to differentiate between row label i and column label j).

Now, if Cij is an AP in matrix space which is generated column wise by the prime P and row wise by the prime Q, then Cij satisfies the following two equations:

$$Cij = P \times AP(Q) + AP(P) = P \times i + j \tag{1}$$

$$Cij = Q \times AP(P) + AP(Q) = Q \times j + i \tag{2}$$

To express Cij in terms of i and j only, and since P = 6j + 1 and Q = 6i + 1 then (1) and (2) becomes respectively:

$$Cij = i\,(6j+1) + j \tag{3}$$

$$Cij = j(6i+1) + i \tag{4}$$

From (3) and (4) we get respectively:

$$(Cij - j) \bmod (6j + 1) = 0 \tag{5}$$

$$(Cij - i) \bmod (6i+1) = 0 \tag{6}$$

Similarly for matrix D,

$$Dij = P \times AP(Q) + 5 \times AP(P) + 4 \tag{7}$$

$$Dij = Q \times AP(P) + 5 \times AP(Q) + 4 \tag{8}$$

To express Dij in terms of i and j only, and since P = 6j + 1 and Q = 6j + 1 then (7) and (8) becomes respectively

$$Dij = i\,(6j+5) + 5j + 4 \tag{9}$$

$$Dij = j(6i+5) + 5i + 4 \tag{10}$$

From (9) and (10) we get respectively:

$$(Dij - 5j - 4) \bmod (6j + 5) = 0 \tag{11}$$

$$(Dij - 5i - 4) \bmod (6i + 5) = 0 \tag{12}$$

Now, based on equations (5), (6), (11) and (12) the prime factors for absolute position is Cij / Dij are the primes with absolute positions i and j.

Although each pair of equations is identical, they behave differently when used to search the matrix space of C/D. Equations (5) and (11) on variable value of j will search the space vertically, while equation (6) and (12) on variable value of i will search the space horizontally.

3. A FACTORIZATION ALGORITHM FOR COMPOSITE NUMBERS IN R1

Given N = p x q in R1, we can use (5), (6), (11) and (12) to scan matrices C and D vertically or/and horizontally to verify whether AP (N) = Cij/Dij is within the matrixes space or not.

To make the search finite, we have to determine the search space limits for each matrix, that is to say, we have to specify an initial value for j and/or i to represent the upper limit, and a lower limit. To determine a suitable initial value for j and/or i, we state the following theorem.

Theorem (2)

Let N ϵ R1 be a large composite number with AP (N) =Cij /Dij. The suitable initial value for j or i, is sqrt(Cij/Dij /6).

Proof

From (5) or (6) we have Cij = i (6j+1) + j. This implies that Cij > 6ij. If we set i=j, then this means Cij > 6 j2 → Cij /6 > j2 → j < sqrt (Cij /6). Thus if we select j= sqrt (Cij /6), then one of the two factors of N will be generated by a j value greater than the selected initial value while the second factor is generated by a j value less than the initial value. The same result follows if we use equation (11) or (12) for Dij.

Now, since the square root of any composite integer number leans towards the smallest factor, it is always more efficient to locate the smallest factor if searching is started from the square root of that number. Therefore the proposed algorithm scans C and D matrix space backwards rather than forwards.

Based on the above stated theory, figure 1 shows the developed algorithm which searches vertically and backwards the matrices C and D simultaneously.

```
Choose a number, N= p x q in R1, you wish to factor
Let n = N div 6 // n represent Cij /Dij
Is (n mod 5 = 0)
YES: N is composite, the two factors are; f1= 5 and f2= N/f1, End.
NO: continue to next step
Let j= sqrt(n/6)
Let PrimalityFlag = 1.
While (j >0)
Is ((n - j) mod (6*j + 1)) = 0
YES: N is composite, the two factors are; f1=j * 6 +1 and f2=N/f1. PrimalityFlag=0, End.
Is ((n- 5j - 4) mod (6j + 5)) = 0
YES: N is composite, the two factors are; f1=j * 6 +5 and f2=N/f1. PrimalityFlag=0, End.
Decrement j
is Prime.
```

Figure 1: Deterministic Factorization Algorithm

If the value of the smallest factor of N is close to the initial value of j, the above algorithm will have a high performance. But if the value of the smallest factor is close to the value 1 the algorithms will achieve bad performance since searching is start from j down to 1.

To solve this problem we propose that the searching space [1, j] has to be divided into two halves [1, j/2] and [j/2, j] and to search the two halves forewords and backwards simultaneously starting from j/2. Figure 2 shows how this will work, and Figure 3 shows the refined algorithm.

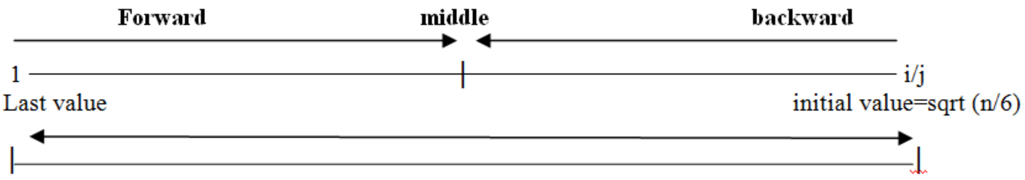

Figure 2: Parallel Searching in the Search Space

```
Choose a number, N in R1, you wish to factor
Let n  = N div  6 // n represent Cij /Dij
Is (n mod 5 = 0)
YES: N is composite, the two factors are; f1= 5 and f2= N/f1.
NO: continue to next step
Let j=  sqrt(n/6)
Let j1 = 1
Let PrimalityFlag = 1.
While (j >j1)
Is ((n - j) mod (6*j + 1)) = 0
YES: N is composite, the two factors are; f1=j * 6 +1 and f2= N/f1. PrimalityFlag=0, End.
Is ((n – j1) mod (6*j1 + 1)) = 0
YES: N is composite, the two factors are; f1=j1 * 6 +1 and f2=N/f1. PrimalityFlag=0, End.
Is ((n- 5*j - 4) mod (6*j + 5)) = 0
YES: N is composite, the two factors are; f1=j * 6 +5 and f2=N/f1. PrimalityFlag=0, End.
Is ((n- 5*j1 - 4) mod (6j*1 + 5)) = 0
YES: N is composite, the two factors are; f1=j1 * 6 +5 and f2=N/f1. PrimalityFlag=0, End.
Decrement j
increase j 1
N is Prime.
```

Figure 3: The Refined Developed Algorithm

4. TESTING AND RESULTS

The developed algorithm is tested against one of the best known factoring algorithm Pollard-Roh. The developed algorithm, its improved version and Pollard-Roh algorithm are tested using three different scenarios, first when the two factors p and q are equal, second when the two factors p and q are too far from each other and third time when the composite number N is prime.

All tests were run on an Intel(R) Core(TM) i3 2.40 GHz processor with 4GB of RAM. Windows 7 Ultimate Service Pack 1 was the host operating system. Version 6.2.2 of the Java Runtime Environment was used. Programs were executed using a command line invocation of the Java client virtual machine. Java Development Kit version 7.0.9 was used for compiling codes. Tables (1 - 5) show the testing results, where time is measured in seconds.

Table 1: Comparison When Factors are Equal and belong to R1

Length in Bit	N	Factors: p and q	Developed Alg	Improved Alg	Pollard rho
16	10609	103 103	.0008	.0006	.033
32	7420616449	86143 86143	.004	.001	.050
64	1067328908494200016 9	3267000013 3267000013	.003	.003	.105
128	1132075226011584461 0878782010589180172 1	10639902377426140939 10639902377426140939	.007	.009	More than 24 hours
256	4976075420923246673 7987782475534088654 4246326438805934466 4532933217600008280 9	223071186416427316086838918075 788680053 223071186416427316086838918075 788680053	.008	.013	More than 48 hours
512	1325957649039914677 4870159396916742868 4177647352555233330 1659829760535904903 6707085434613286019 5581280236717947475 6284835378233478473 4655216327746749444 569	115150234434842323402409730122 262930087264958429512455299234 48468378141034941 3115150234434842323402409730122 262930087264958429512455299234 484683781410349413	.010	.016	More than 96 hours
1024	6391062478810970306 6987778668792026297 9790491303508763876 6301770715555616770 5656104580315445980 5924216098484705814 4052128371956197131 8077490632492996936 7173249927435971422 0762387490729208079 5373780495324200183 5038423612642040481 0034953613795712250 6140796659727273294 5487502390549410187 8233432228168704501 8361	799441209771611054812721173333 160052293377675704670764996367 396268620083843295023910398107 072836959981631464648272070682 601836018119684315422474838221 1019 799441209771611054812721173333 160052293377675704670764996367 396268620083843295023910398107 072836959981631464648272070682 601836018119684315422474838221 1019	.046	.049	More than 384 hours

Table 2: Comparison When Factors are Too Far and belong to R1

Length in Bit	N	Factors	Developed Alg	Improved Alg	Pollard rho
16	19933	31 643	.002	.001	.027
32	3672285319	109 33690691	.016	.003	.025
64	6405689219373700062 1	7 915098459910528580 3	1351	.002	.030
128	5535992206679418465 4286697174480454858 9	7 790856029525631209346952816778 29221227	More than 24 hours	.019	.042
256	4227087677152783041 8228558317733439153 7970015566310661722 3950681293217708680 841	67 630908608530266125645202362951 245360504432859054195017496112 04198406234457923	More than 48 hours	.026	.057
512	7994412097716110548 1272117333316005229 3377675704670764996 3673962686200838432 9502391039810707283 6959981631464648272 0706826018360181196 8431542247483822110 13	13 614954776747393119086708594871 661578687213596695900588458744 150975861602956380787623383159 286797661524331895883286208217 386027706245911011863442183247 001	More than 96 hours	.014	.106
1024	7244745153529326664 8075334003517752694 8150324241419014532 4170149403724031970 5318901834653360553 7790117919865810668 9091299815727422469 4383550187240714909 1245391486769744339 8184239476965785783 3792517357759285011 8811678525949996522 9428419776968442410 3922632894651110383 7583786275932284599 0154909133353333326 3439	7 103496359336133238068679048576 453932421164332034488430647488 144991481771885293312716906647 657933970016845695115809844161 425938963178134833643124629592 727320770212395677762831203421 099511225476846453368227550016 973096932278570931848977425385 263463005603761278073014833940 540896561754942716498701905047 61894777	More than 384 hours	.046	.248

Table 3: Comparison When the Factors are the Same and belong to R5

Length in Bit	N	Factors	Developed Alg	Improved Alg	Pollard rho
16	10201	101 101	.0006	.0006	.026
32	7420271881	86141 86141	.002	.005	.035
64	106732897187400121	3267000011 3267000011	.004	.003	.069
128	1132075226011584460 6622821059618723796 9	10639902377426140937 10639902377426140937	.009	.010	More than 24 hours
256	4976075420923246673 7987782475534088653 5323478982148841822 9797365987284536260 1	2230711864164273160868389180 7 5788680051 2230711864164273160868389180 7 5788680051	.008	.008	More than 48 hours
512	1325957649039914677 4870159396916742868 4177647352555233330 1659829760535904903 6246484496873916725 9484891031827430272 1378852006428496353 7717277592621108046 921	115150234434842323402409730 12 226293008726495842951245529 92 34484683781410349411 115150234434842323402409730 12 226293008726495842951245529 92 34484683781410349411	.015	.016	More than 96 hours
1024	6391062478810970306 6987778668792026297 9790491303508763876 6301770715555616770 5656104580315445980 5924216098484705814 4052128371956197131 8077490632492996936 6853473443527327000 1511502797396567870 3638673467137369583 6491465105194006943 6854857972203283959 2662397394468687363 6659229349815002939 9496170538269351617 4289	79944120977161105481272117 33331600522933776757046707 64996367396268620083843295 02391039810707283695998163 14646482720706826018360181 196843154224748382211017 79944120977161105481272117 33331600522933776757046707 64996367396268620083843295 02391039810707283695998163 14646482720706826018360181 196843154224748382211017	.046	.048	More than 384 hours

Table 4: Comparison When Factors are Too Far and belong to R5

Length in Bit	N	Factors: p and q	Developed Alg	Improved Alg	Pollard rho
16	19951	71 281	.0009	.001	.026
32	3588471073	11 326224643	.046	.001	.027
64	291814300586537338 51	39749 734142495626399	854	.065	.237
128	320525006761579726 662809720356346071 063	461 695282010328806348509348634 17 8624883	More than 24 hours	.012	.050
256	386428403330381178 704023366067828750 814321374536459376 210187389544110528 57353	11 351298548482164707912748514 60 711704619483761321496306928 19 885359491913896123	More than 48 hours	.016	.057
512	727150860362242384 951687694729780067 352468087459065195 698440473108158059 581532385978350856 313316794089835705 607038516875395490 419148650358167494 2402246947	149 488020711652511667752810533 37 569132037078395131480885617 34 499819517839326050552926029 20 037794172345026735372890629 87 709758086605311989956923992 90 2028503	More than 96 hours	.016	.098
1024	586430859437460846 406355903726327304 453556703098198875 941677985205138553 538602445538062393 538675495353962706 604173932864451992 199051716074441280 371596769393000729 037491786699047488 396011903796041885 519180343885877141 658667862876468520 304028869548943164 378538438027905098 275020830773751354 525301061334717788 849	857 684283383240911139330636993 84 635624790380012030128223563 78 972989558209492865839504528 14 928283168606013958817507787 55 293401632695338589973680429 44 826099992917161365665015013 99 121923449255430333771161632 64 917606286714312563344559681 27 424784467555997003084930436 84 980052623424761762863187111 46 4762031895820057	More than 384 hours	.052	.256

Table 5: Comparison When Number is Prime

Length in Bit	Number (Z) Prime	Developed Alg	Improved Alg	Pollard rho
16	46663	.002	.002	.024
32	22121887	.003	.003	.024
64	10639902377426140939	467	463	.029
128	223071186416427316086838918075788680053	7956	7955	.039
256	11515023443484232340240973012226293008726495842951 2455299234484683781410349413	More than 98 hours	More than 98 hours	.049
512	79944120977161105481272117333160052293377675704670 764996367396268620083843295023910398107072836959981631464648272070682601836018119684315422474838221 1019	More than 192 hours	More than 192 hours	.083
1024	14812672559172489972249103890864740981063211050267 18280536738104320663868363315567751535414107598130 31826382848509744213883064825096771209214268697416 02198110584411236874108609993237871487304231419709 01448782468211130281038117941326068287653826533159 84134114355125510400361867987499256156303506019194 867888583	More than 384 hours	More than 384 hours	.173

Form tables (1-5) it is clear that the improved version of the algorithm is highly efficient and out performs Pollard-Rho when the one of the factors is small or when the two factors are equal, but the algorithm is inefficient for primality testing.

For the complexity of the developed algorithms; the worst case of the algorithms is when N is prime, in such case the while loop is executed $(((N/6)^{½}) /6)$ times. The best case is when N is composite with factors p and q are same, in such case the while loop executed only once O (1). The average case is when N is composite and p and q are not same, in this case the while loop executed in average $O(((N/6)^{½}) /24)$ in improved algorithm or $O((N/6)^{½}) /12)$ in deterministic backward algorithm

5. CONCLUSIONS

Factorization problem is an open problem that motivates scientists to develop fast algorithms. The developed algorithm utilizes a new approach for developing primality testing and factorization algorithms that gives new sights to the factorization and primality testing problems.

The positive features of the developed theory and algorithms are simplicity and ease of implementation. The parallel characteristic of the developed algorithms and its dependency on a matrix search algorithm makes it competent to achieve a better performance on refinements.

REFERENCES

1. Mollin, “On Factoring”, *Int. J. Contemp. Math. Sciences,* Vol. 3, 2008, no. 33, 1635- 1642,.
2. Housley et al. *"RFC 2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile."*.
3. Diffie, W. and M. E. Hellman. New directions in cryptography. IEEE Trans. Inform. Theory, 22 (6): 644-654, 1976.
4. Richard P. Brent, *“Some parallel algorithms for integer factorization”,* Springer Link, Volume 19, Issue 2-3, pp 129-145, March 2000.
5. Lehman, “Factoring large integers*”,* Mathematics of Computation 28 (1974), 637-646.

6. M. Voorhoeve, *"Factorization*", in Studieweek Getaltheorie en Computers, Math. Centrum, Amsterdam, 1980, 61-68.
7. R. J. Schoof, *"Quadratic fields and factorization*", Computational methods in number theory, Part II, Math. Centre Tracts, vol. 155, 1982, pp. 165-206.
8. D. Shanks, *"Class number, a theory of factorization, and genera",* Proc. Symp. Pure Math. 20, American Math. Soc, 1971, 415-440.
9. M. A. Morrison and J. Brillhart, *"A method of factorization and the factorization of F7",* Mathematics of Computation 29 (1975), 183-205.
10. Pollard, *"A Monte Carlo method for factorization",* BIT 15 (1975), 331-334.
11. Junod, Pascal. *"Cryptographic Secure Pseudo-Random Bits Generation: The Blum-Blum-Shub Generator",* August 1999.
12. H. W. Lenstra, Jr, *"Factoring integers with elliptic curves",* Ann. of Math. (2) 126 (1987), 649-673.
13. Noureldien A. Noureldien, Mahmud A, and DeiaEldien M. Ahmed, "*A deterministic Factorization and Primality Testing Algorithm for Integers of the form Z mod 6 = -1",* Information Security Assurance, Communications in Computer and Information Science, Volume 200, 2011, PP 156-165. Springer. Verlag Berlin Heidelberg 2011.
14. R. P. Brent, *"Some integer factorization algorithms using elliptic curves",* Australian Computer Science Communications 8 (1986), 149-163.