Security Enhancement and Speed Monitoring of RSA Algorithm

Sarthak R Patel[1], Prof. Khushbu Shah[2]

[1]PG Scholar, [2]Assistant Professor
Computer Engineering Department,
LJIET, Gujarat Technological University, Ahmedabad

***Abstract* - Cryptography involves the science and technique to secure the information. Public key cryptography with the idea of key pair-up is a great break-through for the traditional cryptography. RSA is one of most used asymmetric key encryption algorithm. RSA uses two different keys for encryption and decryption leading to secure transmission of messages. This Research Work mainly focuses on increasing the security of RSA algorithm by using two additional random values for the computation of N, and hence to retain the similar decryption speed the value of N produced by only two prime number is used. That helps to retain the same decryption speed. Proposed algorithm will increase the factoring complexity of the standard algorithm up to minimum 6 times.**

***Key Terms* - RSA, Modular Arithmetic, Cryptography, Cryptosystem, private-key, public-key**

I. INTRODUCTION

Cryptography is a technique to hide the data over communication channel. The developed tangible and hardware tools are not Sufficient to protect the data from unauthenticated parties. Therefore, the experts, researchers and developers have to build and develop security systems, protect the information and prevent the attackers from playing with the very important source (information). For this reason, the term "Encryption" was brought out, and it is the main factor that should be available in protection system and take for a real process to manipulate and generate the security system. It is an art to hide the data to strangers. As the technology grows day by day the need of data security over communication channel is increased to high extent. For securing the knowledge cryptography is used, and this cryptosystem can be distinguished in two major types: Secrete-Key Cryptography and Public Key Cryptography.

II. STANDARD RSA ALGORITHM

RSA is one of most used asymmetric key encryption algorithm [9]. RSA uses multiple keys for encryption and decryption leading to secure transmission of messages. RSA works better if value of the key is long, as it becomes difficult to figure out the factors of n. RSA algorithm involves three different phases [6]:

Phase 1: Key Generation
Phase 2: Encryption
Phase 3: Decryption

Phase 1: Key Generation

RSA involves two keys public key and private key. Public key is used for encryption and private key is used for decryption of message. The key generation takes places as follows:

STEP 1: Take any two large prime numbers P and Q.
STEP 2: Compute N by using the given formula
N= P * Q
STEP 3: Compute Eular's totient function $\emptyset(N)$
$\emptyset(N)$= (P-1) * (Q-1)
STEP 4: Choose the public key exponent E such that
1 < E < $\emptyset(N)$ and, E and $\emptyset(N)$ are co-prime
Which means that GCD (E, $\emptyset(N)$) = 1
STEP 5: Determine private key exponent D through the given formula:
D *E= 1*mod ($\emptyset(N)$)
This means that D is the multiplicative inverse of
E*mod (($\emptyset(N)$).

Now, the public key consists of public key exponent E and N. And private key consists of private key exponent D & N.

Public Key: (N, E) **Private Key: (N, D)**

Phase 2: Encryption
For encrypting any message, the algorithm converts the given message into an integer number by using a suitable padding scheme. Then following formula is used to generate encrypted message **C**:
C = M ^ E mod (N)
Phase 3: Decryption
Following formula is used to decrypt the encrypted message:
M = C ^ D mod (N)

Example of RSA Cryptosystem:
Phase 1: Key Generation
STEP 1: Take any two large prime numbers P and Q.
Choose P = 7 and Q = 17
STEP 2: Compute N by using the given formula N= P * Q
N=3*11 = 119
STEP 3: Compute Euler's totient function ∅(N)
∅(N)= (P-1) * (Q-1)
∅(N)= 6 * 16
∅(N)= 96
STEP 4: Choose the public key exponent E such that 1 < E < ∅(N) and, E and ∅(N) are co-prime Which means that GCD (E, ∅(N)) = 1
Let E = 5
STEP 5: Determine private key exponent D through the given formula: D *E≡ 1*mod (∅(N)). This means that D is the (multiplicative inverse of E)*mod ((∅(N)).
D = 77 [As ED = 1 mod 96, $D = E^{-1}$ mod 96 = 77]
Public Key is (E, N) = (5, 119)
Private Key is (D, N) = (77, 119)

Phase 2: Encryption (Suppose Message is M=20, Public Key (7, 33))
Cipher Text will be calculated through equation C = M ^ E mod (N)
C= M ^ E mod (N)
$C=20^5$ mod 119 = 3200000%119 = 90

Phase 3: Decryption (Cipher Text=90, Private Key (77, 119))
M = C ^ D mod (N)
$M=90^{77}$ *mod 119 = 20*

III. Secure Implementation of RSA Algorithm

Secure Implementation of RSA algorithm using two random numbers for the computation of N value. The same value is used to generate E and D value. :

Phase 1: Key Generation
STEP 1: Take any two large prime numbers P and Q., Also take two random numbers R1 and R2.
STEP 2: Compute N by using the given formula
N=P*Q*R1*R2
STEP 3: Compute Eular's totient function ∅(N)
∅(N)= (P-1) * (Q-1)*(R1-1)*(R2-1)
STEP 4: Choose the public key exponent E such that
1 < E < ∅(N) and, E and ∅(N) are co-prime
Which means that GCD (E, ∅(N)) = 1
STEP 5: Determine private key exponent D through the given formula:
D *E= 1*mod (∅(N))
This means that D is the multiplicative inverse of
E*mod ((∅(N)).

Now, the public key consists of public key exponent E and N. And private key consists of private key exponent D & N.

Public Key: (N, E) Private Key: (N, D)

Phase 2: Encryption
For encrypting any message, the algorithm converts the given message into an integer number by using a suitable padding scheme. Then following formula is used to generate encrypted message **C**:
C = M ^ E mod (N)

Phase 3: Decryption
Following formula is used to decrypt the encrypted message:
M = C ^ D mod (N)

Example of Secure RSA Algorithm with two random numbers:
Let us consider that, we have to send a message whose value is 10 i.e. m=10.
Phase 1: Key Generation
STEP 1: Take two prime numbers x and y. P=5 and Q=3 Take two random numbers X and Y.
X=4 and Y=6
STEP 2: N = (5*3*4*6) => N = 360
STEP 3: ϕ(N)= (5-1)*(3-1)*(4-1)*(6-1) =4*2*3*5 => ϕ(N)= 120
STEP 4: GCD (E, 120) = 1 Thus, E= 7 as its co-prime to 120
STEP 5: 7 * D = 1 mod (120) => D = 103
Public Key = (7,360)
Private Key = (103,360)

Phase 2: Encryption
Encryption of plain text using public key components (7,360).
$C=M^E \bmod (N)$
$C=10^7 \bmod (360) = 280$

Phase 3: Decryption
Decryption of message using Private key components (103, 360)
$M=C^D \bmod (N) = C = 280 \text{ ^ } 103 \bmod (360) = 10$

IV. High Speed and Secure Implementation of RSA Algorithm

RSA involves two keys public key and private key. Public key is used for encryption and private key is used for decryption of message. The key generation takes places as follows:

STEP 1: Take any two large prime numbers P and Q., Also take two random numbers R1 and R2.
STEP 2: Compute N by using the given formula
N1=P*Q*R1*R2, N2=P*Q
STEP 3: Compute Eular's totient function $\emptyset(N1)$
$\emptyset(N1)$= (P-1) * (Q-1)*(R1-1)*(R2-1)
STEP 4: Choose the public key exponent E such that
$1 < E < \emptyset(N1)$ and, E and $\emptyset(N1)$ are co-prime
Which means that GCD (E, $\emptyset(N1)$) = 1
STEP 5: Determine private key exponent D through the givenformula:
D *E= 1*mod ($\emptyset(N1)$)
This means that D is the multiplicative inverse of
E*mod (($\emptyset(N)$).

Now, the public key consists of public key exponent E and N. And private key consists of private key exponent D & N.

Public Key: (E, N1) Private Key: (D, N2)

Phase 2: Encryption
For encrypting any message, the algorithm converts the given message into an integer number by using a suitable padding scheme. Then following formula is used to generate encrypted message **C**:
C = M ^ E mod (N1)
Phase 3: Decryption
Following formula is used to decrypt the encrypted message:
M = C ^ D mod (N2)

Example RSA Algorithm with two random numbers with two different N Values:
Let us consider that, we have to send a message whose value is 10 i.e. m=10. Same as previous example.
Phase 1: Key Generation
STEP 1: Take two prime numbers x and y. P=5 and Q=3 Take two random numbers X and Y.
X=4 and Y=6
STEP 2: N1 = (5*3*4*6) => N1 = 360 | N2 = (5*3) => N2 = 15
STEP 3: ϕ(N)= (5-1)*(3-1)*(4-1)*(6-1) =4*2*3*5 => ϕ(N)= 120
STEP 4: GCD (E, 120) = 1 Thus, E= 7 as its co-prime to 120
STEP 5: 7 * D = 1 mod (120) => D = 103

Public Key = (7,360)
Private Key = (103, 15)

Phase 2: Encryption

Encryption of plain text using public key components (7,360).
$C=M^E \bmod (N1)$
$C=10^7 \bmod (360) = 280$

Phase 3: Decryption

Decryption of message using Private key components (103, 15)
$M=C^D \bmod (N2) = C = 280 \wedge 103 \bmod (15) = 10$

V. IMPLEMENTATION

The proposed algorithm is implemented using PHP coding language. The obtained results are as follows.

Table 1 Encryption and Decryption Time using Standard RSA Algorithm

P	Q	N	Phi(N)	E	D	E.T. (µs)	D.T. (µs)
1359833.000	1359833	1849145787889	1849143068224	3649134810816461	952143487749	0.112	0.094
2190971.000	2275067	4984605820057	4984601354020	3649134810816461	2230804116541	0.115	0.113
3306151.000	3399131	11238040354781	11238033649500	3649134810816461	1150182813641	0.137	0.104
4276829.000	4999307	21381181157503	21381171881368	3649134810816461	4746947729389	0.134	0.081
5788127.000	5792651	33528599654677	33528588073900	3649134810816461	14600829753941	0.112	0.094
6613231.000	6998177	46280561079887	46280547468480	3649134810816461	26767682055941	0.122	0.123
7002257.000	7414711	51919712002727	51919697585760	3649134810816461	20218262294981	0.100	0.094
8462089.000	8964113	75855122012057	75855104585856	3649134810816461	6051306844805	0.111	0.093
9850081.000	9865267	97173679036627	97173659321280	3649134810816461	24676700472581	0.102	0.107
10757543.000	11129113	119721911649359	119721889762704	3649134810816461	38674667625845	0.121	0.135

Table 2 Encryption and Decryption Time using Secure RSA Algorithm

P	Q	R1	R2	N	Phi(N)	E	D	E.T. (µs)	D.T. (µs)
1359833	1350403	45	90	3718553189465470	7191028542249020	3649134810816460	1116482470298110	0.108	0.165
2190971	2275067	88	36	38600787470521400	15178111122990900	3649134810816460	15080649900027000	0.128	0.124
3306151	3399131	74	92	1123804035478100	11238040354781	3649134810816461	2788182527889641	0.133	0.273
4276829	4999307	83	23	123497702365737328	21381181157503	3649134810816461	2976729839239541	0.134	0.271
5788127	5792651	10	84	45900652927252813	33528599654677	3649134810816461	51916855168151141	0.116	0.216
6613231	6998177	61	63	303646761245138607	46280561079887	3649134810816461	127761078695060741	0.103	0.319
7002257	7414711	95	95	2843123429269330 52	51919712002727	3649134810816461	95500542122507621	0.116	0.205
8462089	8964113	10	84	522565935541060673	75855122012057	3649134810816461	49539434601408773	0.121	0.211
9850081	9865267	76	44	242934197591567000	97173679036627	3649134810816460	166385981458503000	0.110	0.209
10757543	11129113	37	48	52797363037367319	119721911649359	3649134810816461	53793803171079941	0.115	0.211

Table 3 Encryption and Decryption Time using High Speed and Secure RSA Algorithm

P	Q	R1	R2	N1	N2	Phi(N)	E	D	E.T. (µs)	D.T. (µs)
1359833	1350403	61	63	6832956255802979	6831109851166080	1836322562699	3649134810816461	2027297117120261	0.106	0.0857
2190971	2275067	95	95	44986067526014425	44043937564120720	4984605820057	3649134810816461	21266540180365861	0.118	0.1023
3306151	3399131	10	84	1123804035478100	8394811136176500	11238040354781	3649134810816461	2788182527889641	0.121	0.0973
4276829	4999307	76	44	123497702365737328	68954279317411800	21381181157503	3649134810816461	2976729839239541	0.118	0.0971
5788127	5792651	37	48	45900652927252813	56730371021038800	33528599654677	3649134810816461	51916855168151141	0.108	0.1159
6613231	6998177	81	49	303646761245138607	177717302278963200	46280561079887	3649134810816461	127761078695060741	0.116	0.119
7002257	7414711	74	92	284312342926933052	344902551062203680	51919712002727	3649134810816461	95500542122507621	0.108	0.105
8462089	8964113	83	23	522565935541060673	136842608672884224	75855122012057	3649134810816461	49539434601408773	0.116	0.111
9850081	9865267	50	49	242934197591567000	228552446723650000	97173679036627	3649134810816460	166385981458503000	0.103	0.109
10757543	11129113	21	27	52797363037367319	62255382676606080	119721911649359	3649134810816461	53793803171079941	0.116	0.131

Table 4 Average Encryption and Decryption Time

Average Time	Encryption Time	Decryption Time
Standard RSA	0.117(µs)	0.104
Secure RSA	0.118(µs)	0.220
High Speed and Secure RSA	0.113(µs)	0.107

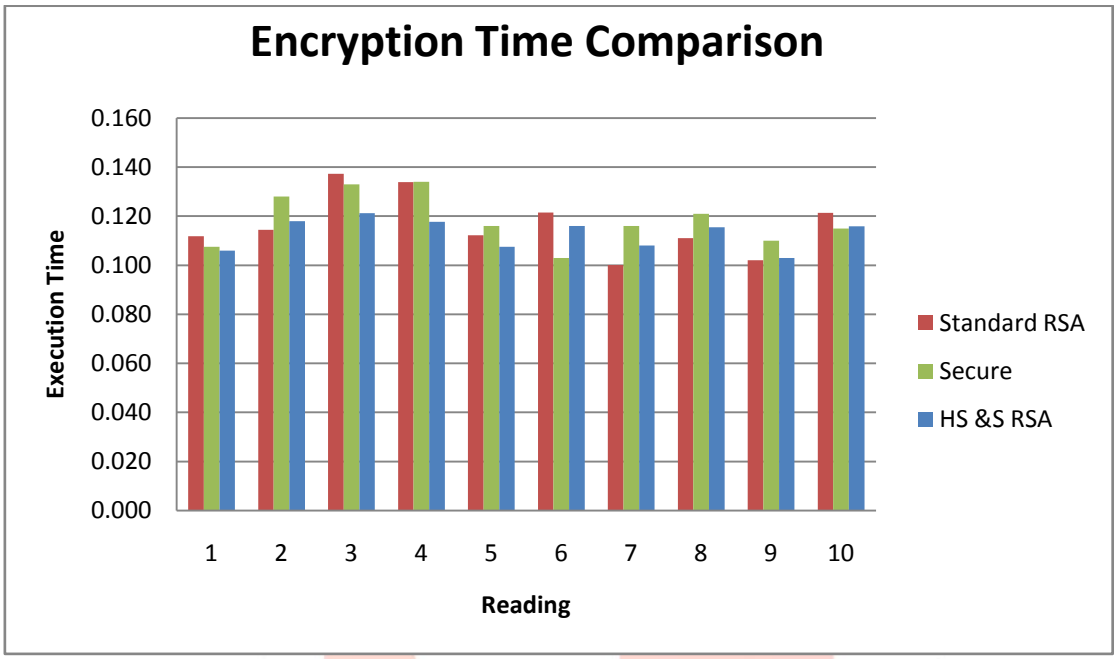

Figure 1: Graph: Encryption Time Comparisons

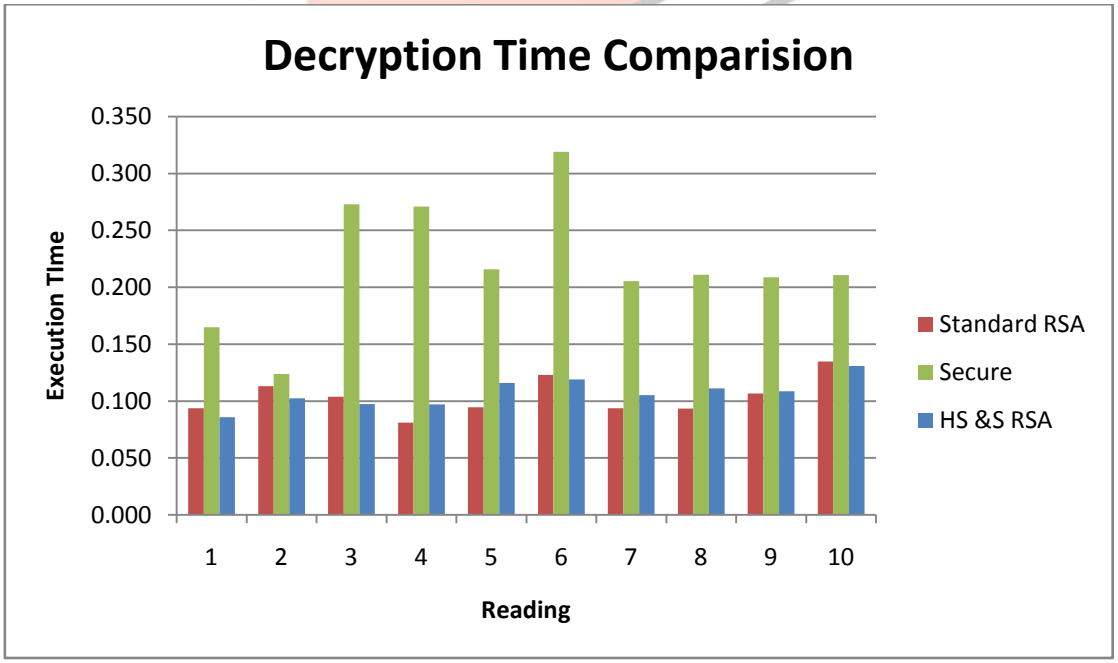

Figure 2: Graph: Decryption Time Comparisons

VI. Analysis

How the security is increased?

A Standard RSA Algorithm is secure enough, but for the factorizing process of N, it gives P, Q as a output. While in the case of modified RSA algorithm, the N (in public key component) is a combination of P, Q, R1, R2, taking the best case, considering R1 and R2 prime numbers, we can have N as a composition of four prime number. Now if we are applying factoring attack and still able to find all the factor of N, we would have 4 factors, which will result in six unique pair of them, which will increase the complexity of RSA algorithm six times. If N=P*Q then we will get single pair of prime number which will make attacking task easy. But in case of N=P*Q*R1*R2, we will have all possible combination of RSA prime pairs as: P*Q, P*R1, P*R2, Q*R1, Q*R2, R1*R2. This is just the case if we have R1 and R2 as prime, but as we have chosen R1 and R2 as Random numbers, it will increase the security of RSA algorithm at better level.

How the speed is maintained?

As we can see that the secure RSA algorithm increases the speed to minimum six times than the original RSA algorithm, but it also causes decrease in decryption speed. Hence it can be increased by using the N value generated by multiplication of P and Q, which also gives correct output and correct decrypted text.

VII. Conclusion

In this paper, we surveyed different methods modified by various researchers and scholars for faster implementation of RSA algorithm. They used various techniques and methodologies in order to achieve high speed implementation of RSA algorithm. As we can see that the secure RSA algorithm increases the speed to minimum six times than the original RSA algorithm, but it also causes decrease in decryption speed. Hence it can be increased by using the N value generated by multiplication of P and Q, which also gives correct output and correct decrypted text.

References

[1] William Stallings, “Cryptography and Network Security”, ISBN 81-7758-011-6, Pearson Education, Third Edition

[2] M. Bahadori, M. R. Mali, O. Sarbishei, M. Atarodi and M. Sharifkhani “A novel approach for secure and fast generation of RSA public and private keys on Smartcard” NEWCAS Conference (NEWCAS), 2010 8th IEEE International, 2010, pp. 265-268.

[3] H. Ren-Junn, S. Feng-Fu, Y. Yi-Shiung and C. Chia-Yao “An efficient decryption method for RSA cryptosystem” Advanced Information Networking and Applications, 2005 (AINA 2005). 19th International Conference on, 2005, pp. 585-590 vol.1.

[4] Nagar, S.A.; Alshamma, S., "High speed implementation of RSA algorithm with modified keys exchange," Sciences of Electronics, Technologies of Information and Telecommunications (SETIT), 2012 6th International Conference on , vol., no., pp.639,642, 21-24 March 2012

[5] Selby, A.; Mitchell, C., "Algorithms for software implementations of RSA," Computers and Digital Techniques, IEE Proceedings E , vol.136, no.3, pp.166,170, May 1989

[6] Dhananjay Pugila, Harsh Chitrala, Salpesh Lunawat, P.M.Durai Raj Vincent “An efficeient encrpytion algorithm based on public key cryptography”,IJET ,Vol 5 No 3 Jun-Jul 2013, pp. 3064-3067

[7] Atul Kahate, Cryptography and Network Security, ISBN-10:0-07-064823-9, Tata McGraw Hill Publishing Company Limited, India, Second Edition, pages 38-62,152-165,205-240.

[8] Chung-Hsien Wu; Jin-Hua Hong; Cheng-Wen Wu, "RSA cryptosystem design based on the Chinese remainder theorem," Design Automation Conference, 2001. Proceedings of the ASP-DAC 2001. Asia and South Pacific , vol., no., pp.391,395, 2001

[9] R. Rivest, A. Shamir, L. Adleman, “A Method for Obtaining Digital Signature and Public-Key Cryptosystems, Communication of the ACM, Vol.21, No.2, 1978, pp. 120-126.

[10] Wang Rui; Chen Ju; Duan Guangwen, "A k-RSA algorithm," Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on , vol., no., pp.21,24, 27-29 May 2011

[11] Hung-Min Sun and Mu-En Wu, "Design of Rebalanced RSA-CRT for Fast Encryption," In Proceedings of Information Security Conference 2005 (ISC 2005), pages 16-27, June 2005, (Best Paper Award, the only one out of 58 papers)

[12] Yadav, Prasant Singh, Pankaj Sharma, and Dr KP Yadav. "Implementation of RSA algorithm using Elliptic curve algorithm for security and performance enhancement" International Journal of Scientific & Technology Research Vol 1.

[13] Sharma, Sonal, Jitendra Singh Yadav, and Prashant Sharma. "Modified RSA Public Key Cryptosystem Using Short Range Natural Number Algorithm." International Journal 2.8 (2012).

[14] J. J. Quisquater and C. Couvreur, “Fast decipherment algorithm for RSA public key cryptosystem,” Electronic Letters, vol. 18, pp.905-907, 1982.

[15] M. J. Wiener, “Cryptanalysis of RSA with short secret exponents,” IEEE Transactions on information Theory, IT-36, pp.553-558, 1990.

[16] P. Kornerup, “A systolic, linear-array multiplier for a class of right-shift algorithms”, IEEE Transactions on Computers, vol. 43, no. 8, pp. 892–898, Aug. 1994.

[17] C. D. Walter, “Systolic modular multiplication”, IEEE Transactions on Computers, vol. 42, no. 3, pp. 376–378, Mar. 1993.

[18] M. Shand and J. Vuillemin, “Fast implementation of RSA cryptography”, in Proc. 11th IEEE Symp. Computer Arithmetic, Windsor, Ontario, June 1993, pp. 252–259.

[19] P.-S. Chen, S.-A. Hwang, and C.-W. Wu, “A systolic RSA public key cryptosystem”, in Proc. IEEE Int. Symp. Circuits and Systems (ISCAS), Atlanta, May 1996, pp. 408–411.

[20] C.-C. Yang, T.-S. Chang, and C.-W. Jen, "A new RSA cryptosystem hardware design based on Montgomery's algorithm", IEEE Trans. Circuits and Systems II: Analog and Digital Signal Processing, vol. 45, no. 7, pp. 908–913, July 1998.
[21] C.-Y. Su, S.-A. Hwang, P.-S. Chen, and C.-W. Wu, "An improved Montgomery algorithm for high-speed RSA public-key cryptosystem", IEEE Trans. VLSI Systems, vol. 7, no. 2, pp. 280–284, June 1999.

[22] J.-H. Hong, P.-Y. Tsai, and C.-W.Wu, "Interleaving schemes for a systolic RSA public-key cryptosystem based on an improved Montgomery's algorithm", in *Proc. 11th VLSI Design/CAD Symp.*, Pingtung, Aug. 2000, pp. 163–166.
[23] P. L. Montgomery, "Modular multiplication without trial division",Math. Computation, vol. 44, pp. 519–521, 1985.