Key Generation Algorithm Design Combination of RSA and ElGamal Algorithm

Ni Made Satvika Iswari
Faculty of Engineering and Informatics
Universitas Multimedia Nusantara
Tangerang, Indonesia
satvika@umn.ac.id

Abstract—**RSA is an algorithm for public-key cryptography and is considered as one of the great advances in the field of public key cryptography. RSA security lies in the difficulty of factoring large number into prime factors. The inventor of RSA Algorithm suggests prime number that is used to generate the keys have more than 100 digits' length for security reasons. Elgamal algorithm also is one of public key cryptography algorithm. The security of this algorithm lies in the difficulty of calculating discrete logarithm. In this paper, the author proposes key generation algorithm that is considered safe from the combination of the RSA and Elgamal algorithm. Based on the experiment that has been done, the computing time required for the proposed algorithm is relatively short, compared to the original RSA algorithm.**

Keywords—RSA; ElGamal; public key cryptography; combination; algorithm design

I. INTRODUCTION

RSA is a popular algorithm in public key cryptography. It was created by three researchers at the MIT (Massachusetts Institute of Technology) in 1976, namely Ron Rivest, Adi Shamir, and Leonard Adleman. RSA algorithm security lies in the difficulty of factoring large numbers into prime factors[1]. Factoring is done to obtain the private key.

While efficient algorithm for factoring large numbers into prime factors is undiscovered, RSA Algorithm security is still ensured. Inventors of this algorithm suggest the prime number used to generate the keys have more than 100-digit length. Thus, the multiplication result of that prime value is larger than 200-digit length. According to Rivest and friends, attempt to find a factor of 200-digit number takes computing time for 4 billion years[1]. Prime numbers which has more than 100 digits, that are considered safe for the RSA algorithm, will certainly take a longer computing time when compared to using prime numbers that have relatively fewer digits.

Elgamal Algorithm is also one of public key cryptography algorithm. This algorithm was originally used for digital signatures, but later was modified to be used for encryption and decryption. The security of this algorithm lies in the difficulty of calculating the discrete logarithm. Discrete logarithm problem is, if p is a prime number and g and y is any integer, then find x such that[2]:

$$g^x = y \ (mod\ p)$$

Speed is the main disadvantage of public-key cryptography, because there is always a tradeoff between efficiency and security[3]. Inventors of RSA suggest the prime number used to generate the keys have more than 100-digit length, but this may cause long computation time. But if we use small digit length for the key size, this will lead to many security holes.

An implementation and analysis of ElGamal and RSA algorithm has been done in [6]. Based on that implementation and analysis, ElGamal algorithm is more secure as compared to RSA algorithm because it generates more complex cipher text and it was also slow because when we encrypt and decrypt it, it generates more than one public keys.

In this paper, the author propose a key generation algorithm made from combination of RSA and Elgamal Algorithm. This algorithm uses 256 bit prime numbers, in order to make computing time for encryption and decryption processes faster than using 1024 bit of prime numbers on the original RSA algorithm. Although the prime numbers that are used is in small bit, but safety factors remain to be addressed by using a combination of RSA and Elgamal algorithm.

II. RSA ALGORITHM

RSA is a popular algorithm in public key crypthography. This algorithm was created by three researchers from MIT (Massachusetts Institute of Technology) in 1976, namely: Ron (R)ivest, Adi (S)Hamir, and Leonard (A)dleman. Its security lies in the difficulty of factoring large numbers into prime factors[1].

The following are properties used in RSA Algorithm:

1. p and q, prime number	(private)
2. $r = p\ .\ q$	(public)
3. $\phi(r) = (p-1)(q-1)$	(private)
4. PK (encryption key)	(public)
5. SK (decryption key)	(private)
6. X (plaintext)	(private)
7. Y(ciphertext)	(public)

A. Key Pair Generation

The following are procedures to generate pair of key for encryption:

1. Choose two random prime number, p and q.
2. Calculate $r = p \, . \, q$. It should be $p \neq q$, because if $p = q$, then $r = p^2$, and p can be obtained from the square root of r.
3. Calculate $\phi(r) = (p-1)(q-1)$
4. Choose public key, PK, which is relatively prime with $\phi(r)$.
5. Generate private key with this equation $SK.PK = 1 \; (mod \; \phi(r))$. Note that this equation is equivalent with $SK.PK = 1 + m\phi(r)$, so SK can be obtained with:

$$SK = \frac{1 + m\phi(r)}{PK}$$

There will be an integer m, so an integer SK can be obtained. Note that SK and PK generation order are interchangeable.

B. Encryption

The following are the encryption procedures:

1. Plaintext is structured into block $x_1, x_2, \ldots$ such that each block represents a value in the range from 0 to $r - 1$.
2. Each block x_i is encrypted to block y_i with this equation:

$$y_i = x_i^{PK} \; mod \; r$$

C. Decryption

The following are the decryption procedures:

1. Each cipher text block y_i is decrypted into block x_i with the following equation:

$$x_i = y_i^{SK} \; mod \; r$$

III. ELGAMAL ALGORITHM

Same with RSA Algorithm, ElGamal is also a public key cryptography algorithm. This algorithm was originally used for digital signature, but was later modified so it also can be used for encryption and decryption. The strength of this algorithm lies in the difficulty of calculating discrete logarithm[2]. To encrypt and decrypt a message, a discrete power is executed. This operation is efficient to compute. An attacker that seeks to decrypt an intercepted message may try to recover the private key. To this end, a logarithm needs to be computed. No actual method exists for this. Under these circumstances, the encryption is secure[4].

Today, ElGamal algorithm is used in many cryptographic products. The open-source software GnuPG uses ElGamal as standard for signatures. On behalf of this software and its problems with ElGamal discovered in late 2003[5].

The following are properties used in ElGamal Algorithm:

1.	Prime number, p	(public)
2.	Random number, $g, (g < p)$	(public)
3.	Random number, $x, (x < p)$	(private)
4.	Plaintext, M	(private)
5.	Ciphertext, a and b	(public)

A. Key Pair Generation

The following are procedures to generate pair of key for encryption:

1. Choose a random prime number, p
2. Choose two random number, g and x, where $(g < p)$ and $(x < p)$
3. Calculate $y = g^x \; mod \; p$
4. y is the public key and x is the private key. The value of g and p is public.

B. Encryption

The following are the encryption procedures:

1. Plaintext is structured into block $m_1, m_2, \ldots$ such that each block represents a value in the range from 0 to $p - 1$
2. Choose a random number, k, where $0 \leq k \leq p - 1$, such that k is relatively prime with $p - 1$
3. Each block of plaintext m is encrypted with the following equation:

$$a = g^k \; mod \; p$$
$$b = y^k m \; mod \; p$$

Pair of a and b is cipher text for message block m, so the size of cipher text is twice the size of its plaintext.

C. Decryption

Decryption of a and b is done by using secret key, x, and plaintext m is recovered by this equation:

$$m = b / a^x m \; mod \; p$$

Note that:

$$a^x \equiv g^{kx} \; (mod \; p)$$

then:

$$\frac{b}{a^x} \equiv \frac{y^k m}{a^x} \equiv \frac{g^{xk} m}{g^{xk}} \equiv m \; (mod \; p)$$

So that, plaintext can be recovered from pair of ciphertext a and b.

IV. Key Generation Algorithm Desgin

Based on the analysis that has been done above, the author obtains some conclusion about both algorithms', RSA and ElGamal as follows:

1. The security of RSA Algorithm lies in the difficulty in factoring non-prime numbers into its prime factors, which in this case: $r = p \, x \, q$.
2. Once r successfully factored into p and q, then $\phi(r) = (p-1)(q-1)$ can be calculated. Furthermore, because the encryption key, PK, is published (not secret), then the decryption key, SK, can be calculated from this equation $PK.SK \equiv 1(mod\ \phi(r))$.
3. The inventors of RSA Algorithm suggests the value of p and q are longer than 100 digits. Thus the result of $r = p \, x \, q$ will be more than 200 digits. According to Rivest and friends, efforts to seek prime factor of 200-digit number takes computing time for 4 billion years (assuming that factoring algorithm used is currently the fastest algorithm and by using computer that has a speed of 1 millisecond)
4. Fortunately, the most efficient algorithm for factoring large numbers have not been found. This is what makes RSA Algorithm still used today. While efficient algorithm for factoring integers into prime factors not yet found, RSA Algorithm is still recommended encrypt messages.
5. The security of ElGamal Algorithm lies in the difficulty of calculating the discrete logarithm.
6. The discrete logarithm in question is, if p is prime number, and g and y is random integer, then find x such that: $g^x \equiv y \ (mod\ p)$

Based on the conclusion above, in this section, the author proposes an algorithm design for RSA key generation using a combination of RSA and ElGamal key generation algorithm.

The following are properties used in original RSA Algorithm design:

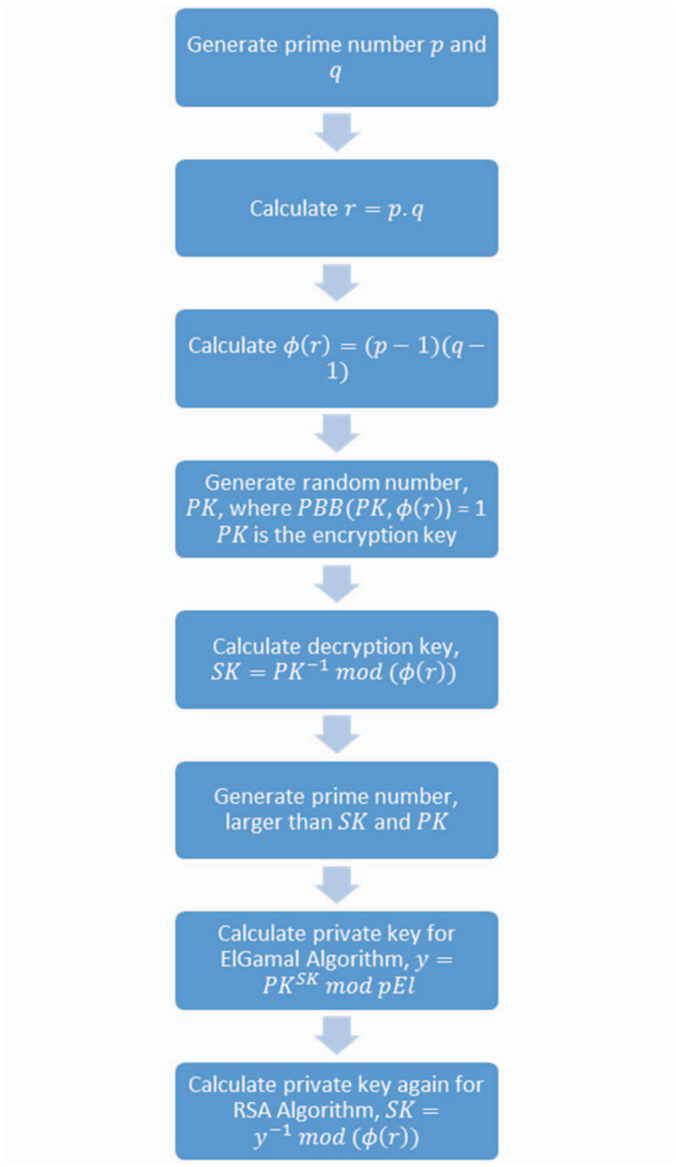
1.	p and q, prime number	(private)
2.	$r = p \, . q$	(public)
3.	$\phi(r) = (p-1)(q-1)$	(private)
4.	PK (encryption key)	(public)
5.	SK (decryption key)	(private)
6.	X (plaintext)	(private)
7.	Y(ciphertext)	(public)

The proposed algorithm design use additional properties obtained from ElGamal Algorithm, which is as follows:

1.	Prime number, pEl	(public)
2.	Random number, $g, (g < pEl)$	(public)
3.	Random number, $x, (x < pEl)$	(private)
4.	$y = g^x \ mod\ pEl$	(public)

The prime number, pEl, is randomly generated larger than random number g and x. Random number, g, which is not secret, is the encryption key, PK, obtained from the key generation for RSA Algorithm. While the random number, x, which is secret, is the decryption key, SK, obtained from the key generation for RSA Algorithm.

The following are the steps to perform key generation:



Generate prime number p and q
Calculate $r = p.q$
Calculate $\phi(r) = (p-1)(q-1)$
Generate random number, PK, where $PBB(PK, \phi(r)) = 1$ PK is the encryption key
Calculate decryption key, $SK = PK^{-1}\ mod\ (\phi(r))$
Generate prime number, larger than SK and PK
Calculate private key for ElGamal Algorithm, $y = PK^{SK}\ mod\ pEl$
Calculate private key again for RSA Algorithm, $SK = y^{-1}\ mod\ (\phi(r))$

Figure 1 Key Generation Steps for The Proposed Algorithm

Processes that was done in the early steps is the same as the key generation process by using RSA Algorithm. First, generate prime numbers p and q. After that, calculate $r = p.q$. Then calculate $\phi(r) = (p-1)(q-1)$. Once the generation is done, generates random number, PK, the encryption key, where the

Greatest Common Divisor, $GCD(PK, \phi(r)) = 1$. Having obtained the encryption key, then we can compute the decryption key with this formula : $SK = PK^{-1} \, mod \, (\phi(r))$.

After the stage of key generation using RSA Algorithm is done, process followed by key generation using ElGamal Algorithm. First, public key, PK, will be a random number, g, for ElGamal Algorithm which is not secret. Then, private key, SK, will be a random number, x, for ElGamal Algorithm which is secret. After that, a prime number, pEl, is generated which is greater than SK and PK. After that, we can calculate ElGamal public key algorithm using this formula:

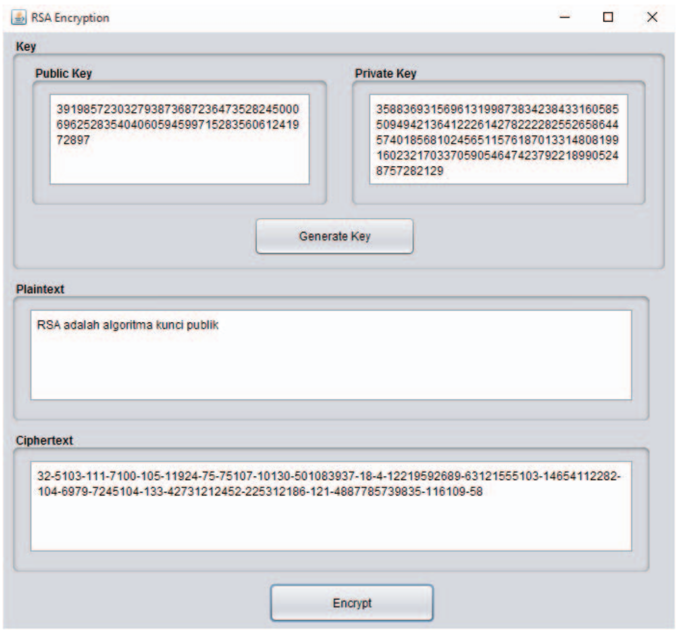
$$y = PK^{SK} \, mod \, pEl$$

The public key, y, which had been obtained is used to recalculate the private key for RSA Algorithm using this formula:

$$SK = y^{-1} \, mod \, (\phi(r))$$

However, for the calculation of this private key, the public key, y, must be based on the requirement to have $GCD(y, \phi(r)) = 1$. Therefore, if the public key, y, is not qualified, then prime number, pEl, is re-generate and the public key is re-calculated.

Figure 2 and Figure 3 is a screenshot of encryption and decryption program using the proposed algorithm for key generation process.



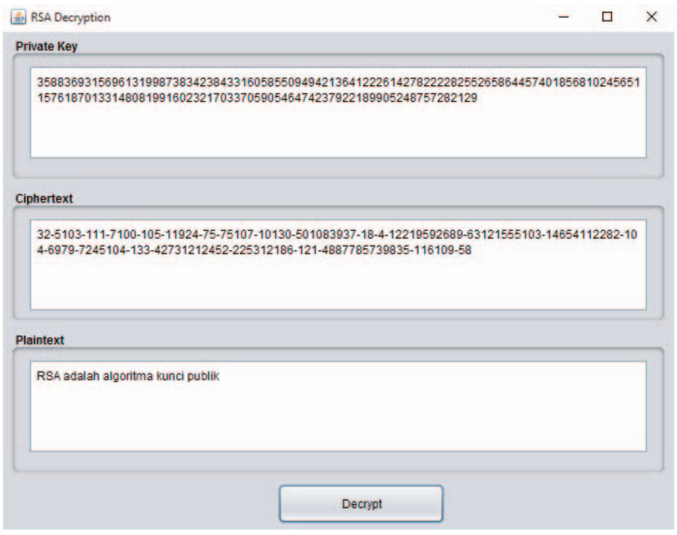
Public Key
3919857230327938736872364735282450006962528354040605945997152835606124197 2897

Private Key
35883693156961319987383423843316058550949421364122261427822228255265864457401856810245651157618701331480819916023217033705905464742379221899052 48757282129

Plaintext
RSA adalah algoritma kunci publik

Ciphertext
32-5103-111-7100-105-11924-75-75107-10130-501083937-18-4-12219592689-63121555103-14654112282-104-6979-7245104-133-42731212452-225312186-121-4887785739835-116109-58

Figure 2 Encryption Program Using Proposed Algorithm for Key Generation



Private Key
358836931569613199873834238433160585509494213641222614278222282552658644574018568102456511576187013314808199160232170337059054647423792218990524 8757282129

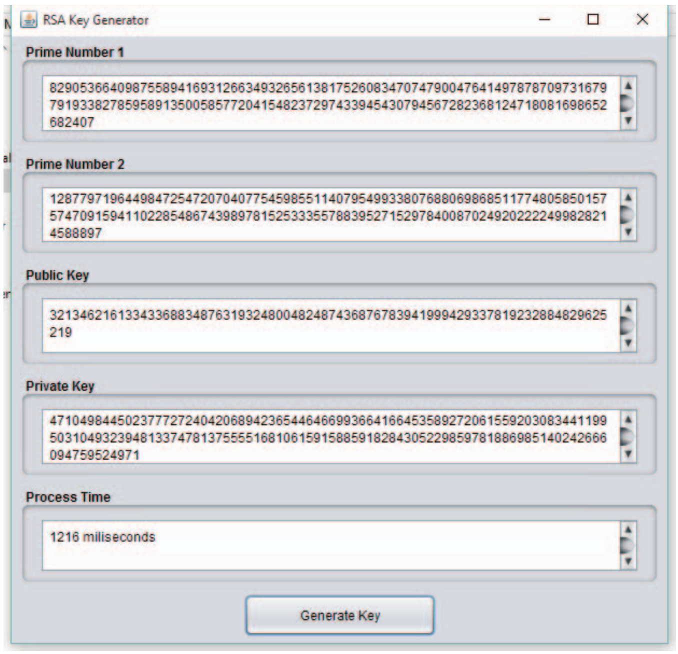
Ciphertext
32-5103-111-7100-105-11924-75-75107-10130-501083937-18-4-12219592689-63121555103-14654112282-104-6979-7245104-133-42731212452-225312186-121-4887785739835-116109-58

Plaintext
RSA adalah algoritma kunci publik

Figure 3 Decryption Program using Proposed Algorithm for Key Generation

V. Experiment and Analysis

In this section, the author tries to compare the key generation process time between the original RSA Algorithm using 1024-bit prime number and the proposed algorithm using 256-bit prime number. For this experiment, the author made a key generator prototype program as shown in Figure 4 for the original RSA, and Figure 5 for the proposed algorithm.



Prime Number 1
82905366409875589416931266349326561381752608347074790047641497878709731679791933827859589135005857720415482372974339454307945672823681247180816986526 82407

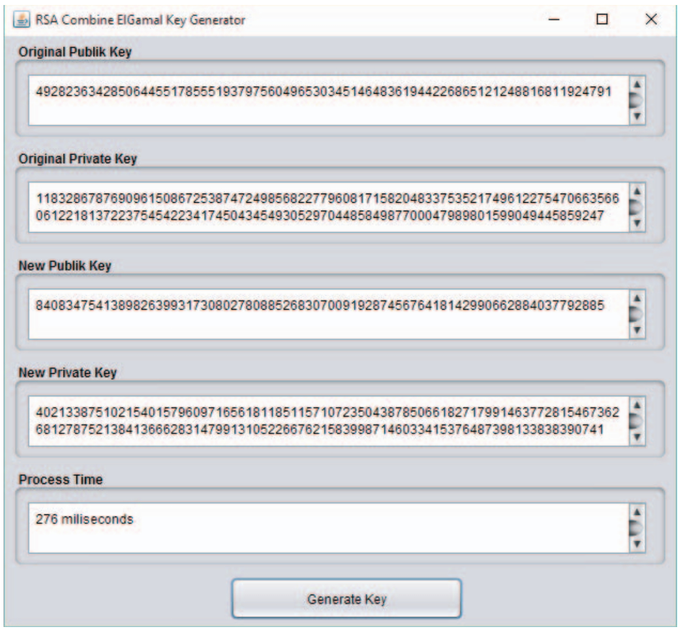
Prime Number 2
12877971964498472547207040775459855114079549933807688069868511774805850157574709159411022854867439897815253335578839527152978400870249202222499828214 588897

Public Key
32134621613343368834876319324800482487436876783941999429337819232884829625219

Private Key
47104984450237772724042068942365446466993664166453589272061559203083441199503104932394813374781375555168106159158859182843052298597818869851402426660 94759524971

Process Time
1216 miliseconds

Figure 4 Key Generation Process using Original RSA Algorithm (1024-bit prime number)



Original Publik Key

49282363428506445517855519379756049653034514648361944226865121248816811924791

Original Private Key

118328678769096150867253874724985682277960817158204833753521749612275470663566
0612218137223754542234174504345493052970448584987700047989801599049445859247

New Publik Key

8408347541389826399317308027808852683070091928745676418142990662884037792885

New Private Key

402133875102154015796097165618118511571072350438785066182717991463772815467362
6812787521384136662831479913105226676215839987146033415376487398133838390741

Process Time

276 miliseconds

Figure 5 Key Generation Process using a Combination of RSA and ElGamal Algorithm (256- bit prime number)

RSA Algorithm is only secure if r, i.e. multiplication between 2 prime numbers is quite long. If the length of r is only 256-bit or less, r can be factored in just a few hours with a single PC. If the length of r is 512-bit or less, n can be factored by using several hundred of computer. At this experiment, the author tries to use RSA key generation algorithm with 1024-bit prime numbers which assumed to be safe according to the RSA's inventor suggestion.

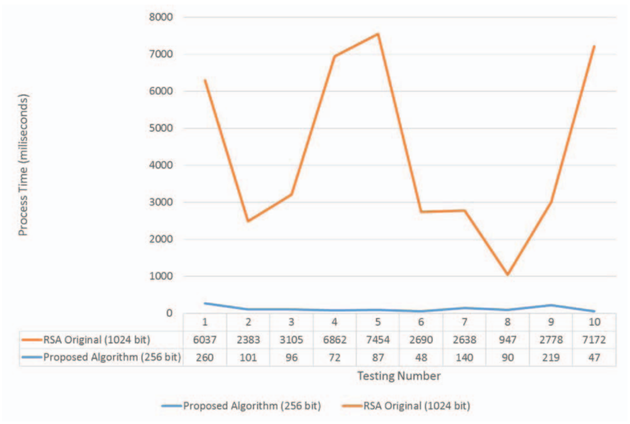
In this experiment, the key generation process is executed 10 times for both algorithms'. The results from this experiment were quite significant. The average computing time for the key generation using original RSA Algorithm with 1024-bit prime number is 4206 milliseconds. While for the proposed algorithm, the average computing time is 116 milliseconds. This comparison process time is shown in Figure 6.

Although only using 256-bit prime number, doesn't mean the proposed algorithm loses its security. Even if r can be factored in just a few hours, but public and private key are not directly known because the discrete logarithm must be calculated, which is the power of ElGamal Algorithm.

VI. Conclusion

From this research and experiment, here are the conclusions:

1. RSA is a famous public key algorithm and used by many applications
2. Security of RSA lies in the difficulty of factoring large numbers into prime factors.
3. ElGamal Algorithm is a public key algorithm that was originally used for digital signatures
4. Security of ElGamal Algorithm lies in the difficulty of calculating the discrete logarithm
5. The proposed algorithm, which is a combination of RSA and ElGamal Algorithm have double security, which is the difficulty of factoring large numbers into prime factors and also the difficulty of calculating the discrete logarithm



	1	2	3	4	5	6	7	8	9	10
RSA Original (1024 bit)	6037	2383	3105	6862	7454	2690	2638	947	2778	7172
Proposed Algorithm (256 bit)	260	101	96	72	87	48	140	90	219	47

Figure 6 Process Time Comparison Between The Original RSA and Proposed Algorithm

VII. Future Works

Based on the work that has been done, combination of RSA and ElGamal for key generation algorithm has decreased computing time required. But, security factor is not quitely proven. It can be proved by doing some piercing test, to factor r and calculate discrete logarithm.

References

[1] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," Commum. ACM, vol. 21, no. 2, pp. 120-126, Feb. 1978.

[2] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," IEEE Transactions on Information Theory, vol. 31, no. 4, pp. 469-472, Jul 1985

[3] S. Mahajan and M. Singh, "Analysis of RSA Algorithm using GPU Programming," International Journal of Network Security & Its Applications (IJNSA), vol. 6, no. 4, Jul 2014

[4] A. V. Meier, "The ElGamal Cryptosystem", June 2005

[5] P. Q. Nguyen, "Can We Trust Cryptographic Software? Cryptographic Flaws in GNU Privacy Guard v1.2.3," Advances in Cryptology-EUROCRYPT 2004, vol 3027 of the series Lecture Notes in Computer Science, pp. 555-570, May 2004.

[6] A. Sharma, J. Attri, A. Devi, and P. Sharma, "Implementation & Analysis of RSA and ElGamal Algorithm," Asian Journal of Advanced Basic Sciences, vol 2, no. 3, pp. 125-129