Isabela State University Linker:
Journal of Engineering, Computing, and Technology
Volume 1, Issue 2
ISSN (Print): 3082-3676 ISSN (Online): 3082-3684
DOI: https://doi.org/10.65141/ject.v1i2.n7

CERS Algorithm: An Enhanced Security System Using Hybrid Cryptosystem

Justine Jhorryeth P. Burgos[1], Paula Bianca S. Dimapilis[2], Edwin R. Arboleda[3], Jericho M. Bojorcelo[4], John Benedict A. de Vila[5]
Department of Computer, Electronics, and Electrical Engineering, College of Engineering and Information Technology, Cavite State University, Indang, Cavite, 4122, Philippines[1,2,3,4,5]
edwin.r.arboleda@cvsu.edu.ph

RESEARCH ARTICLE INFORMATION

Received: May 26, 2023
Reviewed: November 20, 2024
Accepted: December 27, 2024
Published: December 31, 2024

ABSTRACT

Communication and information exchange through the internet have a significant risk of information leakage owing to security issues. In this paper, a new hybrid cryptosystem design is proposed in order to accomplish high-security communication. The Schnorr authentication algorithm is used in this research to ensure a high degree of data security and authentication by integrating RSA, El Gamal, and Chaos-based cryptosystems. The process begins with El Gamal's key generation scheme. El Gamal is well-known for its ability to compute complex discrete logarithms, and when paired with RSA, its abilities are dependent on the difficult process of factorization of large prime integers. Multiple key schemes were generated using the chaotic system. Finally, the Schnorr was utilized to verify the original sender of the message.

Keywords: *Chaos-based system, El Gamal cryptosystem, RSA algorithm, and Schnorr digital signature, hybrid cryptosystem*

Introduction

Multimedia technology is quickly evolving these days, and it is mostly used for communication. Internet communication carries a high risk of unprotected data transfer due to security attacks and insecure services (Padmavathi & Kumari, 2013). Cryptography is a method of protecting data from unauthorized hackers (Vekariya, 2015). It is a method of converting plain text into cipher text through a process known as encryption, and its opposite is known as decryption. Cryptography consists of two

parts: the key and the algorithm (Vekariya, 2015). The term key refers to the numerical and alphanumeric values used by an algorithm to manipulate data, making it safe and accessible only to people who have the necessary key for access (Mo et al., 2017).

The key choice in cryptography is critical since it directly affects the reliability of the encryption technique (Padmavathi & Kumari, 2013). Asymmetric algorithms and symmetric algorithms are the two types of encryption algorithms (Chinnasamy et al., 2021; Singh & Supriya, 2013). A symmetric algorithm is a type of cryptosystem in which just one key is used for both encryption and decryption. It is often referred to as conventional encryption (Al Shabi, 2019; Bansal & Singh, 2016). An asymmetric algorithm is a type of cryptosystem that handles encryption and decryption using a private key and a public key pair (Maxwell et al., 2019). The public key serves as an authentication and encryption key, whereas the private key serves as a decryption key (Alegro et al., 2019). The encryption algorithm's dependability is determined by the security of the key, the length of the key, the initialization vector, and how they all work together.

El Gamal Algorithm

Taher Elgamal initially described the Elgamal Cryptosystem in 1984. It is a public key cryptography method. This algorithm's encryption and decryption are based on the discrete logarithm problem (Rivera et al., 2019). Understanding the message simplifies decryption, which is a critical element of this procedure. The encryption strategy is employed when the RSA algorithm is separated into three distinct processes such as key creation, encryption, and decryption (Magsino et al., 2019; Ukwuoma et al., 2015). The three steps are characterized as follows:

Key Generation:

- Choose a large prime p
- Choose random g and x number such that gcd (x, q) = 1.
- Compute y = gx mod p

The encryption (p, g, y) is the public key and x is kept as the private key.

Encryption Algorithm:

- Prepare a message m ∈ Zp
- Choose any number k that is relatively prime to p − 1
- Calculate p = gk and s = hk = gxk.
- Multiply it with M (p, M*s) = (gk, M*s).

Decryption Algorithm:

- Calculate s′ = px = gxk.
- Divide all M*s by s′ to obtain M as s = s′

Chaos-Based Cryptosystem

Chaotic encryption was introduced by Silakari, Shukla, and Khare and was established in the year 1990. It is a fast and secure cryptosystem wherein chaotic map properties are used. It is an algorithm that utilizes a symmetric key algorithm that uses at least one key to encrypt and decrypt data. This algorithm is used in systems where the speed of encryption and decryption is required (Arboleda, 2019). The three processes: key generation, encryption, and decryption, are defined as follows:

Key Generation:

- Choose the parameter value (M, A, X)
- *Xn+1 = { A* Xn (Xn-1) }MOD 256* using these equation generate keys

where: *n = 1 to j* (number of keys)
A = any integer (1, 2, 3…n)
Xn = chaotic function initial value (2, 3, … n)

- Apply the gray code to the calculated values.
- Convert generated keys to binary form.

Encryption Algorithm:

- Convert the plaintext into its ASCII equivalent form.
- After converting to ASCII characters, then convert it to its 8-bit binary form and convert it into decimal numbers.
- To encrypt the plaintext, use the digital logic XOR and denote it as ciphertext C.
- Get the 1's complement of ciphertext C.

Decryption Algorithm:

- Get the 1's complement of ciphertext C.
- To decrypt the ciphertext C, get the XOR value using the digital logic XOR.
- Convert the generated XOR values into decimal numbers.
- Convert it to 8-bit binary form.
- Find the equivalent ASCII characters.

RSA Algorithm

RSA algorithm is an asymmetric cryptography algorithm that was first described by Ron Rivest, Adi Shamir, and Leonard Adleman of the Massachusetts Institute of Technology. It was first publicly established in the year 1990 and became the most widely used and the most popular cryptographic algorithm. It enables public-key encryption and is widely used to secure sensitive data, particularly when it is being sent over an insecure network such as the internet. The idea of RSA was based on the published works of Diffie and Hellman a few years ago, who introduced the concept of such an algorithm but never really implemented it (Enriquez et al., 2019). The three processes are key generation, encryption, and decryption (Espalmado & Arboleda, 2017), and are defined as follows:

Key Generation:

- Choose two random large prime numbers p and q for defining N = p x q.
- Compute the value of n using the formula n = p*q and ϕ (n) using the formula ϕ (n) = (p-1)*(q-1).
- Choose another number e for this equation gcd (ϕ(N), e) = 1.
- Choose again another number to find the natural number satisfying this equation (d x e) (mod ϕ(N)) = 1.

Encryption Algorithm:

- RSA encryption is done with the help of public key (n, e) to generate the ciphertext.
- Convert the plaintext into its ASCII equivalent.
- Use c = mod n formula to calculate the ciphertext.

Decryption Algorithm:

- In decrypting, use this formula c = mod n to calculate the ciphertext.
  Convert cipher text to ASCII equivalent.

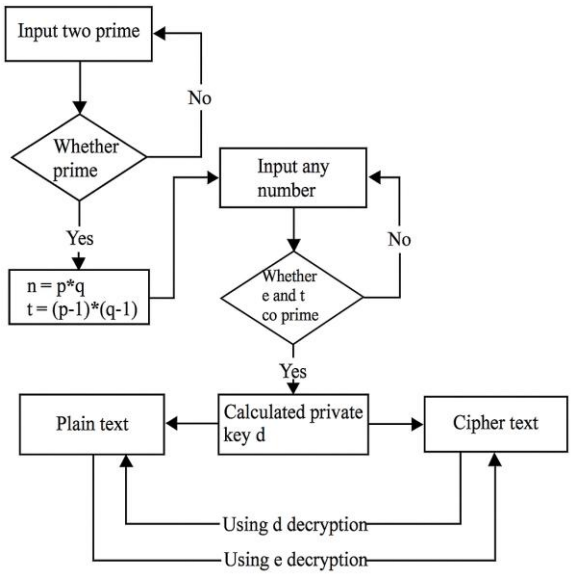

Figure 1. *Flowchart of Rivest-Shamir-Adleman Algorithm (RSA)*

Schnorr Signature Algorithm

Schnorr is a digital signature algorithm that was created by Claus-Peter Schnorr in 1989. The security is based on discrete logarithm problems that are difficult to solve. This algorithm is known as the simplest digital signature scheme considered to be secure in a random oracle model. It is efficient and generates short signatures (Kocarev, 2013). The three processes, key generation, encryption, and decryption are defined as follows:

Key Generation:

- First choose two primes, p, and q, such that q (1 < q <p − 1) is a prime factor of p − 1.
- To generate a public key, choose a is not equal to 1 such that a ≡ h^(p−1)/q (mod p), that is, a^q ≡ h^(p−1) (mod p).
- To generate a key pair, choose a random number s < q which is used as the private key.
- To generate a key pair, choose a random number s < q which is used as the private key.
- Select a value of r, such that r < q, and compute x ≡ ar (mod p).
- Select a value of t and send it to the receiver.
- Calculate y ≡ r +st (mod q) and send it to the receiver.
  The receiver verifies and computes x' ≡ ay · λt (mod p).

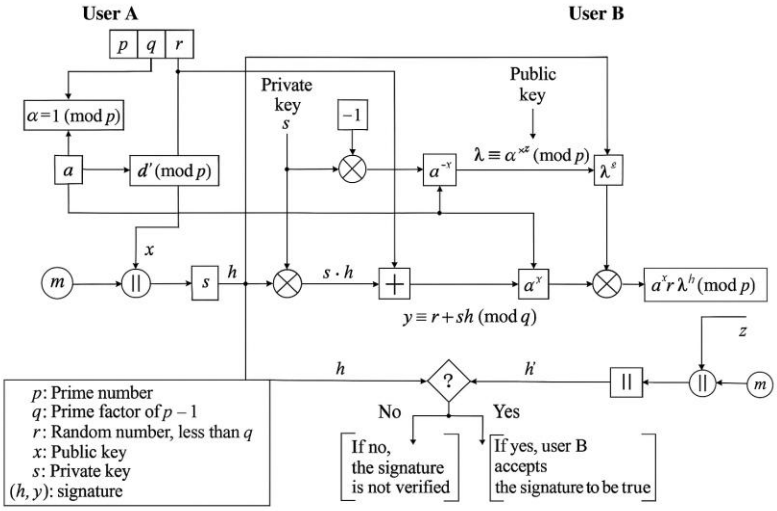

Figure 2. *Schnorr's Authentication Scheme*

Methods

The proposed algorithm is the combination of El Gamal cryptosystem, chaos-based cryptography, Rivest-Shamir-Adleman Algorithm (RSA), and secured with the Schnorr digital signature algorithm using MD4 as its hashing function.

Scheme of the Key Generation

El-Gamal Key Generation:

- Choose two large random different values of p and q satisfying this equation 255 < q < p.
- Choose (i) for the number of keys.
- Choose random multiplicative g_i wherein $g_i < q$.
- Compute $y_i = g_{iq} \pmod p$.

Chaos-Based Key Generation:

- Convert y_i 's to binary.
- Get the equivalent gray code (k_i) of y_i.

RSA Key Generation:

- Compute ϕ(n) = (p-1) (q-1).
- Compute n =(p) (q).
- Choose the value of the public key (e); (e) and ϕ (n) should be coprime and e < p.
- Using the Extended Euclidean Algorithm computes the value of the private key (d).

Schnorr Algorithm Key Generation:

- Choose two prime numbers (x) and (y); (y) must be a prime factor of *x -1.*
- Set the value of (a) and (s) satisfying these two equations $ak \equiv 1 \pmod j$ *and* $s < k$.
- Compute $\lambda \equiv a^{-s} \pmod j$.
- Choose a random number (r) wherein $r < j$.
- Compute $x \equiv a^r \pmod j$.

Scheme of Encryption

Chaos-Based Encryption:

- Convert plaintext into its ASCII binary equivalent.
- XOR the binary equivalent of the plaintext (mi) and the keys generated (ki).
- Get the 1's complement (wi).
- Convert generated values into decimals.

RSA Encryption:

- Compute ci = s_i^e(mod n) using the public keys (e, n).

Schnorr Algorithm:

- Get the summation of the encrypted values.
- Compute h ≡ H (tci.||x) (mod y). Use MD4 as a hashing function algorithm.
- Convert values into decimals.
- Compute u ≡ r + sh (mod y).
- Send values (h, u) to receiver B to verify the identification.

Scheme of Decryption

Schnorr Algorithm:

- Compute v ≡ $a^u \cdot \lambda^h$ (mod j).
- Compute h' ≡ H (tci.||x) (mod k). Use MD4 as a hashing function algorithm.
- Convert values into decimals.
- If it satisfies this equation h = h' accepts the signature.

RSA Decryption:

Compute si = c_i^d(mod n) using the public keys (e, n).

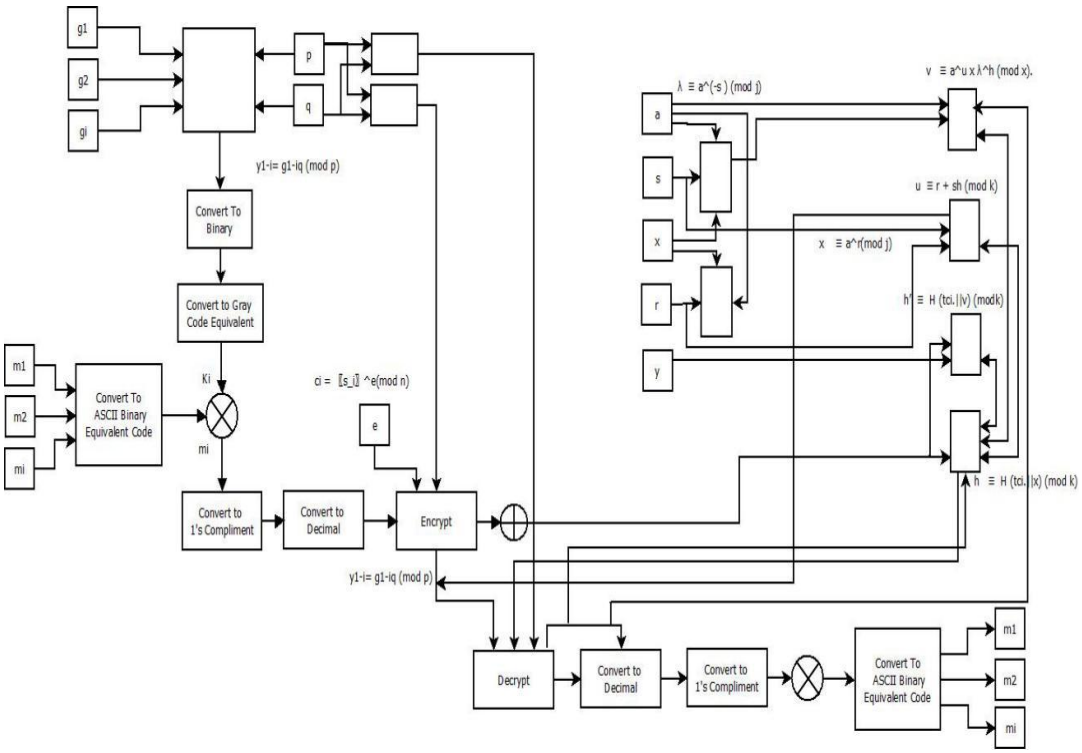

Figure 3. *Block Diagram of the Proposed Algorithm*

Results and Discussion

The proposed algorithm is the combination of El Gamal cryptosystem, chaos-based cryptography, Rivest-Shamir-Adleman Algorithm (RSA), and secured with the Schnorr digital signature algorithm using MD4 as its hashing function.

Key Generation

El Gamal Key Generation:

1. Choose two large random different values of p and q satisfying this equation 255 < q < p.
   q = 260 and p = 270
2. Choose (i) for the number of keys.
   i = 5
3. Choose random multiplicative gi such that gi< q.
   g1 = 100, g2 = 110, g3 = 120, g4 = 130, g5 =140
4. Compute $y_i = g_{iq} \pmod p$.
   $y_1 = g_1{}^q \pmod p = 100^{260} \pmod{270} = 10$
   $y_2 = g_2{}^q \pmod p = 110^{260} \pmod{270} = 40$
   $y_3 = g_3{}^q \pmod p = 120^{260} \pmod{270} = 0$

Chaos Key Generation:

1. Convert yi 's to binary

y_1 = 10	*00001010*
y_2 = 40	00101000
y_3 = 0	00000000
y_4 = 135	10000111
y_5 = 70	01000110

2. Get yi gray code equivalent (ki).

ki	**Binary**		**Gray Code**
k_1 =	00001010	=	00001111
k_2 =	00101000	=	00111100
k_3 =	00000000	=	00000000
k_4 =	10000111	=	11000100
k_5 =	01000110	=	01100101

RSA Key Generation:

1. Compute *ϕ(n) = (p-1)(q-1).*
   Φ (n) = (p-1) (q-1)
   = (270-1) (260-1)
   = (269) (259)
   = 69 671
2. Compute *n =(p) (q).*
   n = pq
   = (270) (260)
   = 70 200
3. Choose the value of the public key (e); (e) and ϕ (n) should be coprime and e < p.
   e = 26
4. Using the Extended Euclidean Algorithm, computes the value of private key d.
   d = 8039

Schnorr Key Generation:

1. Choose two prime numbers (x) and (y); (y) must be a prime factor of x -1.
   x = 29 and y = 7
2. Set the value of (a) and (s) satisfying these two equations $ak \equiv 1(modj)$ *and* $s < k$.
   a = 7 and s = 4
3. Compute $\lambda \equiv a^{-s} \ (mod\ j)$.
   λ ≡ a^(-s) (mod x)
   ≡ 7^(-4)(mod 29)
   ≡ 24
4. Choose a random number (r) wherein $r < j$.
   r = 5
5. Compute $x \equiv a^{r} (mod\ j)$.
   j ≡ a^r(mod x)
   ≡ 7^5(mod 29)
   ≡ 16

Data Encryption

MESSAGE: The quick brown fox jumps over the lazy dog.

1. Convert plaintext into its ASCII binary equivalent as shown in the table. The message must have no space.

Table 1. ASCII Binary Equivalent

Message	ASCII	Binary
T	84	01010100
h	104	01101000
e	101	01100101
q	113	01110001
u	117	01110101
i	105	01101001
c	99	01100011
k	107	01101011
b	98	01100010
r	114	01110010
o	111	01101111
w	119	01110111
n	110	01101110
f	102	01100110
o	111	01101111
x	120	01111000
j	106	01101010
u	117	01110101
m	109	01101101
p	112	01110000
s	115	01110011
o	111	01101111
v	118	01110110
e	101	01100101

r	114	01110010
t	116	01110100
h	104	01101000
e	101	01100101
l	108	01101100
a	97	01100001
z	122	01111010
y	121	01111001
d	100	01100100
o	111	01101111
g	103	01100111

2. XOR the binary equivalent of the plaintext (mi) and the keys generated (ki) as shown in Table 2.

Table 2. XOR-ed Values (mi)

	Message	**Key used**	**m_i XOR k_i**
m_1	01010100	00001111	01011011
m_2	01101000	00111100	01010100
m_3	01100101	00000000	01100101
m_4	01110001	11000100	10110101
m_5	01110101	01100101	00010000
m_6	01101001	00001111	01100110
m_7	01100011	00111100	01011111
m_8	01101011	00000000	01101011
m_9	01100010	11000100	10100110
m_{10}	01110010	01100101	00010111
m_{11}	01101111	00001111	01100000
m_{12}	01110111	00111100	01001011
m_{13}	01101110	00000000	01101110
m_{14}	01100110	11000100	10100010
m_{15}	01101111	01100101	00001010
m_{16}	01111000	00001111	01110111
m_{17}	01101010	00111100	01010110
m_{18}	01110101	00000000	01110101
m_{19}	01101101	11000100	10101001
m_{20}	01110000	01100101	00010101
m_{21}	01110011	00001111	01111100
m_{22}	01101111	00111100	01010011
m_{23}	01110110	00000000	01110110
m_{24}	01100101	11000100	10100001
m_{25}	01110010	01100101	00010111
m_{26}	01110100	00001111	01111011
m_{27}	01101000	00111100	01010100
m_{28}	01100101	00000000	01100101
m_{29}	01101100	11000100	10101000
m_{30}	01100001	01100101	00000100

m_{31}	01111010	00001111	01110101
m_{32}	01111001	00111100	01000101
m_{33}	01100100	00000000	01100100
m_{34}	01101111	11000100	10101011
m_{35}	01100111	01100101	00000010

3. Get the 1's complement (fi) and then convert it to decimal equivalent as shown in the table.

Table 3. Decimal Equivalent (fi)

	m_i XOR k_i	**f_i's (binary)**	**f_i's (decimal)**
f_1	01011011	10100100	164
f_2	01010100	10101011	171
f_3	01100101	10011010	154
f_4	10110101	01001010	74
f_5	00010000	11101111	239
f_6	01100110	10011001	153
f_7	01011111	10100000	160
f_8	01101011	10010100	148
f_9	10100110	01011001	89
f_{10}	00010111	11101000	232
f_{11}	01100000	10011111	159
f_{12}	01001011	10110100	180
f_{13}	01101110	10010001	145
f_{14}	10100010	01011101	93
f_{15}	00001010	11110101	245
f_{16}	01110111	10001000	136
f_{17}	01010110	10101001	169
f_{18}	01110101	10001010	138
f_{19}	10101001	01010110	86
f_{20}	00010101	11101010	234
f_{21}	01111100	10000011	131
f_{22}	01010011	10101100	172
f_{23}	01110110	10001001	137
f_{24}	10100001	01011110	94
f_{25}	00010111	11101000	232
f_{26}	01111011	10000100	132
f_{27}	01010100	10101011	171
f_{28}	01100101	10011010	154
f_{29}	10101000	01010111	87
f_{30}	00000100	11111011	251
f_{31}	01110101	10001010	138
f_{32}	01000101	10111010	186
f_{33}	01100100	10011011	155

f_{34}	10101011	01010100	84
f_{35}	00000010	11111101	253

4. Get the 1's complement (fi) and then convert it to decimal equivalent as shown in the table.

Table 4. RSA Encryption

c_i =	f_i^e (mod n)	Encrypted Message
c_1 =	164^{26} (mod 70 200) =	17,536
c_2 =	171^{26} (mod 70 200) =	57,321
c_3 =	154^{26} (mod 70 200)) =	4
c_4 =	74^{26} (mod 70 200) =	17,176
c_5 =	239^{26} (mod 70 200) =	24,361
c_6 =	153^{26} (mod 70 200) =	65,529
c_7 =	160^{26} (mod 70 200) =	2,200
c_8 =	148^{26} (mod 70 200) =	31,264
c_9 =	89^{26} (mod 70 200) =	21,961
c_{10} =	232^{26} (mod 70 200) =	30,424
c_{11} =	159^{26} (mod 70 200) =	34,641
c_{12} =	180^{26} (mod 70 200) =	32,400
c_{13} =	145^{26} (mod 70 200) =	21,025
c_{14} =	93^{26} (mod 70 200) =	32,049
c_{15} =	245^{26} (mod 70 200) =	36,625
c_{16} =	136^{26} (mod 70 200) =	32,536
c_{17} =	169^{26} (mod 70 200) =	47,281
c_{18}=	138^{26} (mod 70 200) =	21,384
c_{19} =	86^{26} (mod 70 200) =	9,736
c_{20} =	234^{26} (mod 70 200) =	5,616
c_{21} =	131^{26} (mod 70 200) =	12,481
c_{22} =	172^{26} (mod 70 200) =	1,504
c_{23}=	137^{26} (mod 70 200) =	9,409
c_{24} =	94^{26} (mod 70 200) =	39,256
c_{25} =	232^{26} (mod 70 200) =	30,424
c_{26} =	132^{26} (mod 70 200) =	40,824
c_{27} =	171^{26} (mod 70 200) =	57,321
c_{28} =	154^{26} (mod 70 200) =	16,696
c_{29} =	87^{26} (mod 70 200) =	45,009
c_{30} =	251^{26} (mod 70 200) =	63,001
c_{31} =	138^{26} (mod 70 200) =	21,384
c_{32} =	186^{26} (mod 70 200) =	36,936
c_{33} =	155^{26} (mod 70 200) =	625
c_{34} =	84^{26} (mod 70 200) =	16,416
c_{35} =	253^{26} (mod 70 200) =	35,929

5. Get the summation of the encrypted values.
   =965284

Schnorr Algorithm

1. Sign the summation of the encrypted message using Schnorr's signature algorithm.
   a. Compute h ≡ H (tci.||x) (mod y). Use MD4 as a hashing function algorithm and convert it to decimals.
      h ≡ H (tci.||j) (mod y)
      ≡ H (965,284||16) (mod 7)
      = 025efe5eaf8f02cd28dcd844
      c98fac44d3a7c6315 (mod 7)
      =135364103308361698270041195 1
      0719734808113038897 (mod 7)
      = 6
   b. Compute u ≡ r + sh (mod y).
      u ≡ r + sh (mod y)
      ≡ (5 + 4 × 6) (mod 7)
      ≡ 29 (mod 7)
      = 1
2. Send values (h,u) = (6,1) to receiver B to verify the identification.

Encrypted Message:
17536, 57321, 4, 17176, 24361, 65529, 2200, 31264, 21961, 30424, 34641, 32400, 21025, 32049, 36625, 32536, 47281, 21384, 9736, 5616, 12481, 1504, 9409, 39256, 30424, 40824, 57321, 16696, 45009, 63001, 21384, 36936, 625, 16416, 35929.

Data Decryption

1. Authenticate the signature given and accept it as valid if h = h'.
   a. Calculate for the value of v ≡ $a^y \cdot \lambda^h$ (mod x).
      v ≡ a^y x λ^h (mod x)
      ≡ 7^1 x 〖24〗^6 (mod 29).
      ≡ 1,337,720,832 (mod 29)
      ≡ 16
   b. Compute h ≡ H (tci.||x) (mod y). Use MD4 as a hashing function algorithm and convert it to decimals.
      h ≡ H (tci.||j) (mod y)
      ≡ H (965,284||16) (mod 7)
      = 025efe5eaf8f02cd28dcd844
      c98fac44d3a7c6315 (mod 7)
      =135364103308361698270041195 1
      0719734808113038897 (mod 7)
      = 6

   Since the value obtained is identical to the value of h, therefore, user B accepts the signature as valid.
2. Decrypt the ciphertext values Compute ci = s_i^e (mod n) using the public keys (e, n) and encrypt generated values as shown in Table 5. Note that the value of n =70 200 and e = 26.

Table 5. RSA Decryption

c_i =	c_i^e (mod n)	w_i's (decimal)
c_1 =	$27\,197^{47}$ (mod 54 225) =	164
c_2 =	$31\,259^{47}$ (mod 54 225) =	171
c_3 =	$44\,975^{47}$ (mod 54 225) =	154
c_4 =	$2\,925^{47}$ (mod 54 225) =	74
c_5 =	$21\,044^{47}$ (mod 54 225) =	239
c_6 =	$4\,014^{47}$ (mod 54 225) =	153
c_7 =	$9\,551^{47}$ (mod 54 225) =	160
c_8 =	$39\,312^{47}$ (mod 54 225) =	148
c_9 =	$31\,259^{47}$ (mod 54 225) =	89
c_{10} =	$48\,038^{47}$ (mod 54 225) =	232
c_{11} =	$28\,341^{47}$ (mod 54 225) =	159
c_{12} =	$15\,350^{47}$ (mod 54 225) =	180
c_{13} =	$31\,858^{47}$ (mod 54 225) =	145
c_{14} =	$44\,017^{47}$ (mod 54 225) =	93
c_{15} =	783^{47} (mod 54 225) =	245
c_{16} =	$14\,492^{47}$ (mod 54 225) =	136
c_{17} =	$10\,100^{47}$ (mod 54 225) =	169
c_{18}=	$36\,736^{47}$ (mod 54 225) =	138
c_{19} =	$23\,225^{47}$ (mod 54 225) =	86
c_{20} =	$16\,931^{47}$ (mod 54 225) =	234
c_{21} =	$10\,000^{47}$ (mod 54 225) =	131
c_{22} =	$5\,375^{47}$ (mod 54 225) =	172
c_{23}=	$35\,068^{47}$ (mod 54 225) =	137
c_{24} =	$5\,375^{47}$ (mod 54 225) =	94
c_{25} =	$48\,038^{47}$ (mod 54 225) =	232
c_{26} =	$37\,471^{47}$ (mod 54 225) =	132
c_{27} =	$31\,259^{47}$ (mod 54 225) =	171
c_{28} =	$44\,975^{47}$ (mod 54 225) =	154
c_{29} =	$54\,162^{47}$ (mod 54 225) =	87
c_{30} =	$27\,826^{47}$ (mod 54 225) =	251
c_{31} =	$38\,551^{47}$ (mod 54 225) =	138
c_{32} =	$11\,623^{47}$ (mod 54 225) =	186
c_{33} =	$20\,583^{47}$ (mod 54 225) =	155
c_{34} =	$6\,525^{47}$ (mod 54 225) =	84
c_{35} =	$15\,768^{47}$ (mod 54 225) =	253

3. Convert the decimal value of fi's to its binary form. After that, convert it to its 1's complement equivalent as shown in Table 6.

Table 6. XOR-ed Decimal Equivalent (fi)

	f_i's (decimal)	**f_i's (binary)**	**m_i XOR k_i**
f_1	164	10100100	01011011
f_2	171	10101011	01010100
f_3	154	10011010	01100101
f_4	74	01001010	10110101
f_5	239	11101111	00010000
f_6	153	10011001	01100110
f_7	160	10100000	01011111
f_8	148	10010100	01101011
f_9	89	01011001	10100110
f_{10}	232	11101000	00010111
f_{11}	159	10011111	01100000
f_{12}	180	10110100	01001011
f_{13}	145	10010001	01101110
f_{14}	93	01011101	10100010
f_{15}	245	11110101	00001010
f_{16}	136	10001000	01110111
f_{17}	169	10101001	01010110
f_{18}	138	10001010	01110101
f_{19}	86	01010110	10101001
f_{20}	234	11101010	00010101
f_{21}	131	10000011	01111100
f_{22}	172	10101100	01010011
f_{23}	137	10001001	01110110
f_{24}	94	01011110	10100001
f_{25}	232	11101000	00010111
f_{26}	132	10000100	01111011
f_{27}	171	10101011	01010100
f_{28}	154	10011010	01100101
f_{29}	87	01010111	10101000
f_{30}	251	11111011	00000100
f_{31}	138	10001010	01110101
f_{32}	186	10111010	01000101
f_{33}	155	10011011	01100100
f_{34}	84	01010100	10101011
f_{35}	253	11111101	00000010

4. Using the XOR operation, XOR the binary forms of the one's complement wi'S with the corresponding secret key ki and denote mi as shown in Table 7.

Table 7. XOR-ed Equivalent (fi)

	Message	Key used	m_i XOR k_i
m_1	01010100	00001111	01011011
m_2	01101000	00111100	01010100
m_3	01100101	00000000	01100101
m_4	01110001	11000100	10110101
m_5	01110101	01100101	00010000
m_6	01101001	00001111	01100110
m_7	01100011	00111100	01011111
m_8	01101011	00000000	01101011
m_9	01100010	11000100	10100110
m_{10}	01110010	01100101	00010111
m_{11}	01101111	00001111	01100000
m_{12}	01110111	00111100	01001011
m_{13}	01101110	00000000	01101110
m_{14}	01100110	11000100	10100010
m_{15}	01101111	01100101	00001010
m_{16}	01111000	00001111	01110111
m_{17}	01101010	00111100	01010110
m_{18}	01110101	00000000	01110101
m_{19}	01101101	11000100	10101001
m_{20}	01110000	01100101	00010101
m_{21}	01110011	00001111	01111100
m_{22}	01101111	00111100	01010011
m_{23}	01110110	00000000	01110110
m_{24}	01100101	11000100	10100001
m_{25}	01110010	01100101	00010111
m_{26}	01110100	00001111	01111011
m_{27}	01101000	00111100	01010100
m_{28}	01100101	00000000	01100101
m_{29}	01101100	11000100	10101000
m_{30}	01100001	01100101	00000100
m_{31}	01111010	00001111	01110101
m_{32}	01111001	00111100	01000101
m_{33}	01100100	00000000	01100100
m_{34}	01101111	11000100	10101011
m_{35}	01100111	01100101	00000010

5. Convert the mi's value to its ASCII equivalent without space as shown in Table 8.

Table 8. ASCII Message Equivalent (fi)

Binary	ASCII	Message
01010100	84	T
01101000	104	H
01100101	101	E
01110001	113	Q
01110101	117	U

01101001	105	I
01100011	99	C
01101011	107	K
01100010	98	B
01110010	114	R
01101111	111	O
01110111	119	W
01101110	110	N
01100110	102	F
01101111	111	O
01111000	120	X
01101010	106	J
01110101	117	U
01101101	109	M
F01110000	112	P
01110011	115	S
01101111	111	O
01110110	118	V
01100101	101	E
01110010	114	R
01110100	116	T
01101000	104	H
01100101	101	E
01101100	108	L
01100001	97	a
01111010	122	z
01111001	121	y
01100100	100	d
01101111	111	o
01100111	103	g

Decrypted Message: The quick brown fox jumps over the lazy dog

Conclusion and Future Works

Stand-alone algorithms are already difficult to decipher and decrypt. The proposed algorithm is composed of those stand-alone algorithms and it is the combination of El Gamal cryptosystem, chaos-based cryptography, Rivest-Shamir-Adleman Algorithm (RSA), and secured with the Schnorr digital signature algorithm using MD4 as its hashing function. Based on the results and data that were gathered, it is proven that the combination of the proposed algorithm makes an effective high level of security from unauthorized hackers.

References

[1] Al-Shabi, M. A. (2019). A survey on symmetric and asymmetric cryptography algorithms in information security. *International Journal of Scientific and*

Research Publications (IJSRP), 9(3), 8779. https://doi.org/10.29322/ijsrp.9.03.2019.p8779

[2] Alegro, J. K. P., Arboleda, E. R., Pereña, M. R., & Dellosa, R. M. (2019). Hybrid Schnorr, RSA, and AES cryptosystem. *International Journal of Scientific and Technology Research, 8*, 1770–1776.

[3] Arboleda, E. R. (2019). Secure and fast chaotic El Gamal cryptosystem. *International Journal of Engineering and Advance Technology (IJEAT), 8*(5), 1693–1699.

[4] Bansal, V. P., & Singh, S. (2016). A hybrid data encryption technique using RSA and Blowfish for cloud computing on FPGAs. In *2015 2nd International Conference on Recent Advances in Engineering and Computational Sciences (RAECS 2015)* (pp. 1–5). https://doi.org/10.1109/RAECS.2015.7453367

[5] Chinnasamy, P., Padmavathi, S., Swathy, R., & Rakesh, S. (2021). Efficient data security using hybrid cryptography on cloud computing. *Lecture Notes in Networks and Systems, 145*, 537–547. https://doi.org/10.1007/978-981-15-7345-3_46

[6] Enriquez, M., Garcia, D. W., & Arboleda, E. (2019). Enhanced hybrid algorithm of secure and fast chaos-based AES, RSA, and ElGamal cryptosystems. *Indian Journal of Science and Technology, 10*(27).

[7] Espalmado, J. M. B., & Arboleda, E. R. (2017). DARE algorithm: A new security protocol by integration of different cryptographic techniques. *International Journal of Electrical and Computer Engineering, 7*(2), 1032–1041.

[8] Kocarev, L. (2013). Cryptography. *IEEE Transactions on Circuits and Systems, 60*(1), 5–21. https://doi.org/10.1109/7384.963463

[9] Kumari, P., Kumar, U., & Singh, S. K. (2019). *Schnorr digital signature to improve security using quantum cryptography.* Springer Singapore. https://doi.org/10.1007/978-981-13-3185-5

[10] Magsino, J. P., Arboleda, E. R., & Corpuz, R. R. (2019). Enhancing security of El Gamal encryption scheme using RSA and chaos algorithm for e-commerce application. *International Journal of Scientific and Technology Research, 8*(11), November.

[11] Maxwell, G., Poelstra, A., Seurin, Y., & Wuille, P. (2019). Simple Schnorr multi-signatures with applications to Bitcoin. *Designs, Codes, and Cryptography, 87*(9), 2139–2164. https://doi.org/10.1007/s10623-019-00608-x

[12] Mo, F., Hsu, Y. C., Chang, H. H., Pan, S. C., Yan, J. J., & Liao, T. L. (2017). Design of an improved RSA cryptosystem based on synchronization of discrete chaotic systems. In *Proceedings of the 2016 International Conference on Information System and Artificial Intelligence (ISAI 2016)* (pp. 9–13). https://doi.org/10.1109/ISAI.2016.0012

[13] Padmavathi, B., & Kumari, S. R. (2013). A survey on performance analysis of DES, AES, and RSA algorithm along with LSB substitution technique. *International Journal of Science and Research, 2*(4), 2319–7064. http://www.ijsr.net

[14] Rivera, L. B., Bay, J. A., Arboleda, E. R., Pereña, M. R., & Dellosa, R. M. (2019). Hybrid cryptosystem using RSA, DSA, ElGamal, and AES. *International Journal of Scientific and Technology Research*.

[15] Singh, G., & Supriya, S. (2013). A study of encryption algorithms (RSA, DES, 3DES, and AES) for information security. *International Journal of Computer Applications, 67*(19), 33–38. https://doi.org/10.5120/11507-7224

[16] Ukwuoma, H., Studies, S., & Hammawa, M. (2015). Optimized key generation for RSA encryption. *Innovative Systems Design and Engineering, 6*(11), 35–45.

[17] Vekariya, M. (2015). Comparative analysis of cryptographic algorithms and advanced cryptographic algorithms. *International Journal of Computer Engineering and Sciences, 1*(1), 1. https://doi.org/10.26472/ijces.v1i1.20

Conflict of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.