**Data Science : Journal Of Computing And Applied Informatics**
Journal homepage: https://jocai.usu.ac.id

# A Hybrid Cryptosystem Using Rprime RSA And Extended Tiny Encryption (XTEA) For Securing Message

Zikri Akmal Santoso[*1] , Mohammad Andri Budiman[2] , Syahril Efendi[3]

[1,2,3]*Master of Informatics Program, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Medan, Indonesia*
[*]Corresponding Author: mandrib@usu.ac.id

**ARTICLE INFO**

**Article history:**
Received 29 May 2024
Revised 13 June 2024
Accepted 4 July 2024
Available 29 January 2025

E-ISSN: 2580-829X
P-ISSN: 2580-6769

**How to cite:**
Z. A. Santoso, M. A. Budiman and S. Efendi, "A Hybrid Cryptosystem Using Rprime RSA And Extended Tiny Encryption (XTEA) For Securing Message," Journal of Computing and Applied Informatics, vol. V9, no.1, Jan. 2025, doi: 10.32734/jocai.v9.i1-16574


http://doi.org/10.32734/jocai.v9.i1-16574

**ABSTRACT**

Ensuring the security of messages in sending message publicly is very important, we must ensure the security of messages with one of security method called cryptography. Focusing solely on security can affect the speed of message delivery processes. Therefore, this research is conducted to provide solutions to both of these issues. Thus, this research will discuss the Analysis of Hybrid Cryptography Scheme in the combination of RPrime RSA and XTEA (Extended Tiny Encryption) in securing instant messages. Hybrid cryptography is one of the methods in cryptography that allows to enhance speed of message delivery with messages encrypted by symmetric algorithms and the symmetric algorithm keys will be encrypted using asymmetric algorithm's public keys. RPrime RSA is an asymmetric public key algorithm and one variant of RSA, which is a combination of Rebalanced RSA and MPrime RSA algorithms. XTEA is a symmetric key algorithm and improved version of the TEA algorithm. This research tested by using strings with the value of k in RPrime RSA from 2 to 6 with unconstrained modulus digits. The result of the test indicates that the required time for encryption and decryption is proportional with the length of character, the time processing for factorization to get d is proportional to the value of *k*.
**Keyword:** RPrime RSA, Extended Tiny Encryption Algortihm, Hybrid Cryptosystem, Cryptography, Instant Messaging

**ABSTRAK**

Memastikan keamanan pesan dalam pengiriman pesan secara publik sangat penting. Kita harus menjamin keamanan pesan dengan metode keamanan yang disebut kriptografi. Jika fokus pada keamanan dapat mempengaruhi kecepatan proses pengiriman pesan. Maka, penelitian ini menyediakan solusi untuk kedua masalah tersebut. Dalam penelitian ini membahas Analisis Kriptografi skema Hibrida pada kombinasi RPrime RSA dan XTEA (*Extended Tiny Encryption*) dalam pengamanan pesan cepat. Kriptografi Hibrida adalah metode dalam kriptografi yang dapat meningkatkan kecepatan dalam pengiriman pesan dengan melakukan enkripsi pesan menggunakan algoritma simetris. Serta kunci dari algoritma simetris akan dienkripsi menggunakan kunci publik dari algoritma asimetris. RPrime RSA adalah algoritma asimeris kunci publik dari salah satu varian RSA, yang merupakan kombinasi dari Algoritma Rebalanced RSA dan Mprime RSA. XTEA adalah algoritma kunci simetris yang merupakan peningkatan versi dari algoritma TEA. Penelitian telah di uji menggunakan teks dengan nilai *k* pada RPrime RSA dari 2 sampai 6 dengan modulus digit yang tidak dibatasi. Hasil pengujiannya didapati bahwa waktu yang dibutuhkan untuk proses enkripsi dan dekripsi berbanding lurus dengan panjang karakter dan waktu proses faktorisasi untuk mendapatkan nilai *d* adalah berbanding lurus dengan nilai *k*.
**Keyword:** Rprime RSA, Extended Tiny Encryption Algorithm, Kriptografi, Kriptografi Hibrida, Pesan Cepat

## 1. Introduction

The proliferation of digital communication has brought forth numerous challenges, chief among them being the prevalence of theft and interception of messages. Securing digital information from unauthorized access and manipulation has become paramount. Cryptography emerges as a fundamental technique to address these concerns.

Hybrid cryptosystem is a method that combine between Asymmetric and Symmetric cryptography algorithm. Speed of message delivery process can be deliver with this method because to encrypt and decrypt message symmetric algorithm can be used to get small computation and to approach the security of key of symmetric algorithm, Asymmetric algorithm can be use to encrypt or decrypt the key with public and private key.

This research will discuss the analysis of Hybrid cryptosystem scheme in combining Rprime RSA and XTEA Algorithm for securing the message. Rprime RSA is an asymmetric public key algorithm which one variant of RSA, RPrime RSA is combination between Rebalanced RSA and MPrime RSA, It has been standardized in PKCS #1 and is reported to be 27 times faster than the conventional RSA algorithm [5]. Extended Tiny Encryption Algorithm (XTEA) is a symmetric key algorithm and improved version of the TEA algorithm, XTEA was designed to correct weaknesses of TEA Algorithm [7]. XTEA has more complex operations and the sequence within the Shift, $\oplus$ operations, and several additional additions [1].

## 2. Literature Review

### *2.1. Cryptography*

Cryptography is the art of securing data and validating its integrity once it's secured. It involves two main steps, encryption and decryption. Encryption renders data unreadable to unauthorized parties through the use of a key, turning the data into ciphertext. Decryption reverses the encryption process, making the data readable again for authorized recipients.

A cryptographic algorithm based on a key can be categorized into two main types: symmetric and asymmetric. Symmetric, also known as conventional, cryptography utilizes the same key for both encryption and decryption. In contrast, asymmetric cryptography employs different keys for these processes. In asymmetric cryptography, the encryption key, known as the public key, can be openly distributed, while the decryption key, known as the private key, is kept securely for personal use. Hence, asymmetric cryptography is also referred to as public key cryptography.

### *2.2. Hybrid Cryptosystem*

Hybrid Cryptosystem is a method that used for combining two types of cryptography algorithm which that symmetric cryptography and asymmetric cryptography, Hybrid cryptosystem prevent asymmetric algorithm or public key scheme to encrypt the large amount of data [4], with this method the plaintext will be encrypted by symmetric algorithm and than the key of symmetric algorithm will be secured by using asymmetric algorithm[4]. With this method the sender can encrypt the large amount of data using symmetric with small size of cipher text and the key of symmetric will be secured by asymmetric algorithm with small cost because the key is not large as data that will be encrypt even with big computation of asymmetric algorithm [6].

## 3. Method

### *3.1. RPrime RSA*

Rprime RSA is an asymmetric public-key algorithm, a fusion of Rebalanced RSA and MPrime RSA. It has been standardized in PKCS #1 and is reported to be 27 times faster than the conventional RSA algorithm [5]. The security of RPrime RSA, as well as Rebalanced RSA, relies on the strength provided by the private exponent d. As Public Key Algorithm, Rprime RSA is divided into three process Namely, key generation, encryption and decryption are as follows:

Key Generation :

1. Determine $k$ and select $k$ prime numbers: $p_1, p_2, \dots, p_k$ such that $GCD(p_1 - 1, p_2 - 1, \dots, p_k - 1) = 2$.

2. Calculate :

   $n = p_1 \times p_2 \times \dots \times p_k$

   $\phi(n) = (p_1 - 1)(p_2 - 1) \dots (p_k - 1)$

Generate $k$ random numbers $dp1, dp2, \dots, dpk$ such that:
$dp_1 \equiv dp_2 \equiv \cdots \equiv dp_k \pmod 2$,
$GCD\,(dp_1, p_1 - 1) = 1$,
$GCD\,(dp_2, p_2 - 1) = 1$,
and so on up to $dp_k$ , $GCD(dp_k, p_k - 1) = 1$.

3. Calculate $d$ d using the Chinese Remainder Theorem (CRT) such that:
   $d \equiv dp_1 \pmod{p_1 - 1}$
   $d \equiv dp_2 \pmod{p_2 - 1}$
   $d \equiv dp_k \pmod{p_k - 1}$
   And so on for the remaining prime factors.

4. Calculate $e = d^{-1} \left(mod\ \phi(n)\right)$

5. Public key is $(n, e)$ and private key is $(p_1, p_2, \dots, p_k, dp_1, dp_2, \dots dp_k)$

Encryption:
1. Input plaintext as $(M)$
2. Calculate ciphertext using $C = M^e \bmod n$

Decryption:
1. Input ciphertext as $(C)$
2. Calculate cipher text using $M = C^d \bmod n$

*3.2. Extended Tiny Encryption Algorithm*

XTEA (Extended Tiny Encryption Algorithm) is a symmetric key algorithm. It is an evolution of the TEA (Tiny Encryption Algorithm) algorithm. The distinction from the previous algorithm lies in more complex operations and the sequence within the Shift, $\oplus$ operations, and several additional additions. It employs a distinct round function that alters the sections of the key utilized in each round. The i-th iteration of XTEA, bears resemblance to the i-th iteration of TEA [1].
Encryption and decryption will follow this steps:
The 128-bit inputted key will be divided into 4 sub-blocks, each consisting of $S[0] = 32$ bits, $S[1] = 32$ bits, $S[2] = 32$ bits, and $S[3] = 32$ bits :

1. The plaintext block, which is 64 bits, is then split into 2 sub-blocks, $v0$ = 32 bits and $v1$ = 32 v bits. The process is initialized with a variable $i$ = 1.
2. During encryption, the plaintext sub-block $v0$ is processed using keys according to the formula $v0 \mathrel{+}= (((v1 \ll 4) \oplus (v1 \gg 5) + v1) \oplus (sum + S[sum \text{ ^ } 3])$. Similarly, the plaintext sub-block $v1$ is processed using keys as $v1 \mathrel{+}= (((v0 \ll 4) \oplus (v0 \gg 5) + v0) \oplus (sum + S[(sum \gg 11) \text{ ^ } 3])$.
3. During decryption, the ciphertext sub-block $v1$ is processed using keys according to the formula $v1 \mathrel{-}= (((v0 \ll 4) \oplus (v0 \gg 5) + v0) \oplus (sum + S[(sum \gg 11) \text{ ^ } 3])$. Similarly, the plaintext sub-block $v0$ is processed as $v0 \mathrel{-}= \big(((v1 \ll 4) \oplus (v1 \gg 5) + v1) \oplus (sum + S[sum \text{ ^ } 3])\big)$.
4. Then, the process increments $i = i + 1$.
5. If the process has not reached 32 rounds, then repeat the process from step 5. [8]

*3.3. RPrime RSA and XTEA Using Hybrid Cryptosystem.*

As symmetric algorithm XTEA will be used to encrypt the plain text / original message. And as asymmetric key algorithm RPrime RSA will be used to encrypt the encryption key of XTEA. In figure 1, we can see that Alice (sender) want to send a message to bob (recipient) through public network/unsecure network. Alice's message/plaintext will be encrypted by using XTEA with key encryption and the output will be ciphertext, then the key encryption (plainkey) of plaintext will be encrypted by using RPrime RSA with Bob's Public key and the output will be cipherkey, then Alice send the ciphertext and cipherkey to Bob, when ciphertext and cipherkey received by Bob, the cipherkey will be decrypted by using RPrime RSA with Bob's

private key, once cipherkey decrypted as key, the ciphertext will be decrypted using XTEA with decrypted key and the output will be plaintext / original message that sent by Alice.

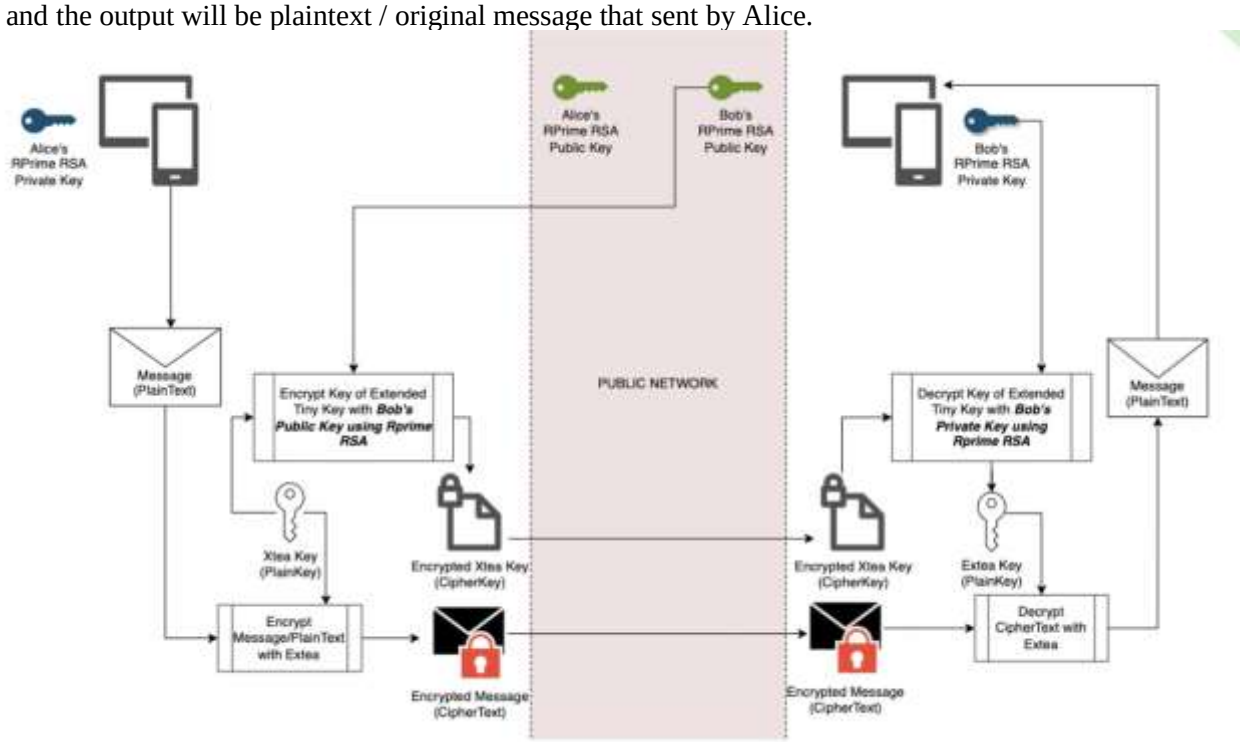


Figure 1. Rprime RSA and EXTEA using hybrid cryptosystem

### *3.4. Kraitchik's Factorization Algorithm*

The Kraitchik Factorization Algorithm is a factorization algorithm developed by Maurice Kraitchik in 1920. It is an extension of Fermat's difference of squares algorithm. Since Fermat's difference of squares does not always return prime numbers as its factors and the modulus of RPrime RSA is generated from several prime number multiplications, Fermat's difference of squares needs to be modified to factorize the modulus [2][3]. Below are the steps involved in the Kraitchik factorization algorithm:

1. Take the value of *n* which is the integer to be factorized.
2. Calculate $x = \lceil \sqrt{n} \rceil$
3. While $x2 - kn \geq 0$ do :
   a. Initialize the value of $k = 1$
   b. While $x2 - kn \geq 0$ do :
      i. $y = \sqrt{(x^2 - kn)}$
      ii. if *y* is a perfect square and $(x + y) mod\, n \neq 0$ and $(x - y)\ mod\ n \neq 0$ then the factor of n are $p = GCD(x + y, n)$ dan $q = GCD(x - y, n)$, then stop.
      iii. Otherwise, $k = k + 2$
   c. Within the iteration $x = x + 1$

## 4. Result and Discussions

### *4.1. Key Generation using RPrime RSA*

In hybrid cryptosystem the recipient should have private key and public key to receive the message from sender.

Calculate $p_1, p_2, \ldots p_k$ with $k = 3$.
System will generate randomely $p_1, p_2, p_3$ and the results are:
$p_1 = 17,\ p_2 = 11,\ p_3 = 23$
$GCD(17 - 1{,}11 - 1) = GCD(16{,}10) = 2$
$GCD(11 - 1{,}23 - 1) = GCD(10{,}22) = 2$
$GCD(17 - 1{,}23 - 1) = GCD(16{,}22) = 2$

Calculate $n = p_1 \times p_2 \times \ldots p_k$ and $\varphi(n) = (p_1 - 1) \times (p_2 - 1) \times (p_k - 1)$.
$n = 17 \times 11 \times 23 = 4301$
$\varphi(n) = (17 - 1) \times (11 - 1) \times (23 - 1) = 16 \times 10 \times 22 = 3520$
System will generate randomly $dp_1, dp_2, \ldots, dp_k$ as much as $k$.
with these conditions:
$dp_1 = dp_2 = \ldots = dp_k \pmod 2$
$GCD(dp_1, p_1 - 1) = 1$
$GCD(dp_2, p_2 - 1) = 1$
$GCD(dp_k, p_k - 1) = 1$
$dp_1 = 9, dp_2 = 3, dp_3 = 5$
$GCD(9,16) = 1$
$GCD(3,10) = 1$
$GCD(5,22) = 1$

Calculate $d = x_1 \cap x_2 \cap x_3$ with CRT

$d \equiv 9 \pmod{16}$
$d \equiv 3 \pmod{10}$
$d \equiv 5 \pmod{22}$

$x_1 = \{9, 25, 41, 57, 73, 89, 105, 121, 137, 153, \ldots, 249, 265, 281, 297, \mathbf{313} \ldots\}$
$x_2 = \{3, 13, 23, 33, 43, 53, 63, 73, 83, 93, \ldots, 253, 263, 273, 283, 293, 303, \mathbf{313}, \ldots\}$
$x_3 = \{5, 27, 49, 71, 93, 115, 137, 159, 181, 203, 225, 247, 269, 291, \mathbf{313}, \ldots\}$

$d = x_1 \cap x_2 \cap x_3 = 313$

Calculate e with $e = d^{-1} \bmod \varphi(n)$
$e = d^{-1} \left(mod\ \varphi(n)\right)$
$e = 313^{-1} \ (mod\ 3520) = 1417$

Public key results are:
$n = 4301$
$e = 1417$

Private key results are:
$p_1 = 17, \quad p_2 = 11, \quad p_3 = 23$
$dp_1 = 9, \quad dp_2 = 3, \quad dp_3 = 5$

*4.2. Message encryption using XTEA*

key : "RAHASIAKITASEMUA"
message : "ZIKRI AS"

128-bit will be divided to 4 sub blocks
with $s[0] = 32\ bit, s[1] = 32\ bit, s[2] = 32\ bit, s[3] = 32\ bit$
key : "RAHASIAKITASEMUA" = {82 65 72 65 83 73 65 75 73 84 65 83 69 77 85 65}

$S[0] = \left((82\ \hat{}\ 255) \ll 24\right) \vee \left((65\ \hat{}\ 255) \ll 16\right) \vee \left((72\ \hat{}\ 255) \ll 8\right) \vee \left((65\ \hat{}\ 255)\right)$
$S[0] = \left((82 \ll 24) \vee (65 \ll 16) \vee (72 \ll 8) \vee (65)\right)$
$S[0] = 1375731712 \vee 4259840 \vee 18432 \vee 65$
$S[0] = 1380010049$
$S[1] = \left((83\ \hat{}\ 255) \ll 24\right) \vee \left((73\ \hat{}\ 255) \ll 16\right) \vee \left((65\ \hat{}\ 255) \ll 8\right) \vee \left((75\ \hat{}\ 255)\right)$
$S[1] = \left((83 \ll 24) \vee (73 \ll 16) \vee (65 \ll 8) \vee (75)\right)$

$S[1] = 1392508928 \vee 4784128 \vee 16640 \vee 75$
$S[1] = 1397309771$
$S[2] = ((73 \wedge 255) \ll 24) \vee ((84 \wedge 255) \ll 16) \vee ((65 \wedge 255) \ll 8) \vee ((83 \wedge 255))$
$S[2] = ((73 \ll 24) \vee (84 \ll 16) \vee (65 \ll 8) OR (83))$
$S[2] = 1224736768 \vee 5505024 \vee 16640 \vee 83$
$S[2] = 1230258515$
$S[3] = ((69 \wedge 255) \ll 24) \vee ((77 \wedge 255) \ll 16) \vee ((85 \wedge 255) \ll 8) \vee ((65 \wedge 255))$
$S[3] = ((69 \ll 24) \vee (77 \ll 16) \vee (85 \ll 8) \vee (65))$
$S[3] = 1157627904 \vee 5046272 \vee 21760 \vee 65$
$S[3] = 1162696001$
plaintext with 64 bit will be divided to 2 subblock which that v0= 32 bit and v1= 32 bit
Plaintext : "ZIKRI AS"
$v0 = ((90 \ll 24) \vee (73 \ll 16) \vee (75 \ll 8) \vee (82))$
$= (1509949440 \vee 4784128 \vee 19200 \vee 82)$
$= 1514752850$
$v1 = ((73 \ll 24) \vee (32 \ll 16) \vee (65 \ll 8) \vee (83))$
$= (1224736768 \vee 2097152 \vee 16640 \vee 83)$
$= 1226850643$

Intiate delta with $0x9E3779B9$ ($-1640531527$ in integer) and 32 rounds (n), calculate v0 dengan with value of $sum = 0$. Calculate with 32 bit int. the limit has $-2147483648$ to $21447483647$

$delta = -1640531527\ (0x9E3779B9)$
$sum = 0$
encryption process for *plaintext subblock* v0 calculate by :
$v0 += (((v1 \ll 4) \oplus (v1 \gg 5) + v1) \oplus (sum + S[sum \wedge 3])$
and for subblocak v1 calculate by :
$v1 += ((( (v0 \ll 4) \oplus (v0 \gg 5) + v0) \oplus (sum + S[sum \gg 11 \wedge 3])$.
initiated $S, v0, v1$ result by these values :
$S[0] = 1380010049$
$S[1] = 1397309771$
$S[2] = 1230258515$
$S[3] = 1162696001$
$v0 = 1514752850$
$v1 = 1226850643$

Round 1 :
$v0 += ((1226850643 \ll 4) \oplus (1226850643 \gg 5) + 1226850643) \oplus (sum + S[sum \wedge 3])$
$v0 += ((-1845226192 \oplus 38339082) + 1226850643) \oplus (0 + S[0 \wedge 3])$
$v0 += ((-1845226192 \oplus 38339082) + 1226850643) \oplus (0 + S[0])$
$v0 += ((-1845226192 \oplus 38339082) + 1226850643) \oplus (1380010049)$
$v0 = -445293538$

$sum = sum + delta$
$sum = 0 + -1640531527$
$sum = -1640531527$

$v1 \mathrel{+}= \big((-445293538 << 4) \oplus (-445293538 >> 5) + -445293538\big) \oplus (-1640531527 + S[-1640531527 >> 11\text{^}3])$
$v1 \mathrel{+}= ((1465237984 \oplus -13915424 + -445293538) \oplus (-1640531527 + S[3])$.
$v1 \mathrel{+}= ((1465237984 \oplus -13915424 + -445293538) \oplus (-1640531527 + 1162696001)$.
$v1 = -1214821577$

continue for 32 rounds so that in 32 round we got the final result $v0 = 395610080, v1 = 1559759634$

The result for sub block 0 :
$ct[0] = v0 \gg 24\text{^}255 = 23$
$ct[0] = v0 \gg 16\text{^}255 = 148$
$ct[0] = v0 \gg 8\text{^}255 = 135$
$ct[0] = v0 \gg 0\text{^}255 = 224$
sub block 1
$ct[0] = v1 \gg 24\text{^}255 = 92$
$ct[0] = v1 \gg 16\text{^}255 = 248$
$ct[0] = v1 \gg 8\text{^}255 = 11$
$ct[0] = v1 \gg 0\text{^}255 = 18$

the final encryption result is:

Ciphertext : “ à\ø “ => [23,148,135,224,92,248,11,18] (in ascii)

*4.3. Key Encryption Using RPrime RSA*
Key : “RAHASIAKITASEMUA”
in ascii = {82 65 72 65 83 73 65 75 73 84 65 83 69 77 85 65}

Recipient’s Public Key (RPrime RSA)
$n = 4301$
$e = 1417$

Calculate with : $C = M^e \bmod n$
Calculate the cipherkey with each of key’s character:
$C[0] = 82^{1417} \bmod 4301 = 3$
$C[1] = 65^{1417} \bmod 4301 = 3046$
$C[2] = 72^{1417} \bmod 4301 = 2571$
$C[3] = 65^{1417} \bmod 4301 = 3046$
$C[4] = 83^{1417} \bmod 4301 = 3517$
$C[5] = 73^{1417} \bmod 4301 = 1117$
$C[6] = 65^{1417} \bmod 4301 = 3046$
$C[7] = 75^{1417} \bmod 4301 = 4294$
$C[8] = 73^{1417} \bmod 4301 = 1117$
$C[9] = 84^{1417} \bmod 4301 = 2107$
$C[10] = 65^{1417} \bmod 4301 = 3046$
$C[11] = 83^{1417} \bmod 4301 = 3517$
$C[12] = 69^{1417} \bmod 4301 = 460$
$C[13] = 77^{1417} \bmod 4301 = 1573$
$C[14] = 85^{1417} \bmod 4301 = 629$
$C[15] = 65^{1417} \bmod 4301 = 3046$

and the result of cipherkey is :
$\{3, 3046, 2571, 3046, 3517, 1117, 3046, 4294, 1117, 2107, 3046, 3517, 460, 1573, 629, 3046\}$

*4.4. Cipherkey Decryption Using RPrime RSA*

To decrypt the ciphertext, recipient need to decrypt the cipherkey using RPrime RSA.
Cipherkey: $\{3, 3046, 2571, 3046, 3517, 1117, 3046, 4294, 1117, 2107, 3046, 3517, 460, 1573, 629, 3046\}$

Recipient's private key*:*
$p_1 = 17, \quad p_2 = 11, \quad p_3 = 23,$
$dp_1 = 9, \quad dp_2 = 3, \quad dp_3 = 5,$
$d = 313$

Calculate with : $M = C^d \bmod n$

Calculate the plainkey (original key) with each of cipherkey's index:

$M[0] = 3^{313} \bmod 4301 = 82$
$M[1] = 3046^{313} \bmod 4301 = 65$
$M[2] = 2571^{313} \bmod 4301 = 72$
$M[3] = 3046^{313} \bmod 4301 = 65$
$M[4] = 3517^{313} \bmod 4301 = 83$
$M[5] = 1117^{313} \bmod 4301 = 73$
$M[6] = 3046^{313} \bmod 4301 = 65$
$M[7] = 4294^{313} \bmod 4301 = 75$
$M[8] = 1117^{313} \bmod 4301 = 73$
$M[9] = 2107^{313} \bmod 4301 = 84$
$M[10] = 3046^{313} \bmod 4301 = 65$
$M[11] = 3517^{313} \bmod 4301 = 83$
$M[12] = 460^{313} \bmod 4301 = 69$
$M[13] = 1573^{313} \bmod 4301 = 77$
$M[14] = 629^{313} \bmod 4301 = 85$
$M[15] = 3046^{313} \bmod 4301 = 65$

The result of decrypted key is {82 65 72 65 83 73 65 75 73 84 65 83 69 77 85 65}
Key : "RAHASIAKITASEMUA" (converted back to char)

*4.5. Ciphertext Decryption Using XTEA*

After the process above now we got the original key / plain key and we'll decrypt the ciphertext by using XTEA.

key : "RAHASIAKITASEMUA"
ciphertext : " à\ø "

128-bit will be divided to 4 sub blocks, with $s[0] = 32\ bit, s[1] = 32\ bit, s[2] = 32\ bit, s[3] = 32\ bit$, The output of this process will be same as encryption step above. And we got the result :
$S[0] = 1380010049$
$S[1] = 1397309771$
$S[2] = 1230258515$
$S[3] = 1162696001$

*ciphertext* with 64 bit will be divided to 2 subblock which that $v0 = 32\ bit$ and $v1 = 32\ bit$

$v0 = ((23 << 24) \vee (148 << 16) \vee (135 << 8) \vee (224))$
$= (385875968 \vee 9699328 \vee 34560 \vee 224)$
$= 395610080$

$v1 = ((92 << 24) \vee (248 << 16) \vee (11 << 8) \vee (18))$
$= (1543503872 \vee 16252928 \vee 2816 \vee 18)$
$= 1559759634$

Initiate delta value with $0x9E3779B9$ ($-1640531527$ in integer) and 32 rounds (n), kalkulasi nilai v0 dengan inisialisasi nilai sum = 0 * round. Calculate with 32 bit int. And has the limit between $-2147483648$ to $21447483647$
$delta = -1640531527\ (0x9E3779B9)$

$sum = delta * round$
$sum = -1640531527 * 32$
$sum = -957401312$

decryption process for *plaintext subblock* v1 calculate by :
$v1 -= (((v0 << 4) \oplus (v0 >> 5) + v0) \oplus (sum + S[sum >> 11 \wedge 3]))$

decryption process for *plaintext subblock* v0 calculate by :
$v0 -= (((v1 << 4) \oplus (v1 >> 5) + v1) \oplus (sum + S[sum \wedge 3]))$
initiated $S, v0, v1$ result by these values :

$v0 = 395610080$
$v1 = 1559759634$

$delta = -1640531527\ (0x9E3779B9)$
$sum = -957401312$

Round 1 :
$v1 -= (((v0 << 4) \oplus (v0 >> 5) + v0) \oplus (sum + S[sum >> 11 \wedge 3]))$.
$v1 -= (((395610080 << 4) \oplus (395610080 >> 5) + 395610080) \oplus (-957401312 + S[-957401312 >> 11 \wedge 3]))$
$v1 -= ((2034793984 \oplus 12362815 + 395610080) \oplus (-957401312 + S[2]))$
$v1 -= ((2034793984 \oplus 12362815 + 395610080) \oplus (-957401312 + 1230258515))$
$v1 = -832355395$

$sum = sum - delta$
$sum = -957401312 - (-1640531527)$
$sum = 683130215$

$v0 -= ((v1 << 4) \oplus (v1 >> 5) + v1) \oplus (sum + S[sum \wedge 3]))$
$v0 -= ((1559759634 << 4) \oplus (1559759634 >> 5) + 1559759634) \oplus (683130215 + S[683130215 \wedge 3]))$
$v0 -= (-1294005664 \oplus -19304571 + -617746266) \oplus (683130215 + S[3]))$
$v0 -= (-1294005664 \oplus -19304571 + -617746266) \oplus (683130215 + 1162696001))$
$v0 = 395610080$

Continue for 32 rounds so that in 32 round we got the final result $v0 = 1514752850, v1 = 1226850643$

sub block 0

$pt[0] = 1514752850 >> 24 \wedge 255 = 90$
$pt[1] = 1514752850 >> 16 \wedge 255 = 73$
$pt[2] = 1514752850 >> 8 \wedge 255 = 75$
$pt[3] = 1514752850 >> 0 \wedge 255 = 82$

sub block 1

$pt[4] = 1226850643 >> 24 \wedge 255 = 73$
$pt[5] = 1226850643 >> 16 \wedge 255 = 32$
$pt[6] = 1226850643 >> 8 \wedge 255 = 65$
$pt[7] = 1226850643 >> 0 \wedge 255 = 83$

The final result for plaintext: "ZIKRI AS"
The final result is same with the first message/plaintext before the entire hybrid encryption process.

### *4.6. Running Time Process*

The computation from key generation, encryption and decryption, and modulus factorization is done by using python programming language and the operation g system is Sonoma 14.5. Text editor visual studio code and with python 2.7. The processor is Apple M1 Pro and the memory is 16 GB. The results of running Time Rprime RSA (Key generation, key encryption, and key decryption) is shown in Table 1. And the relation between *k* and key generation RPrime RSA is depicted in Figure 2, the relation between *k* and Rprime RSA encryption process depicted in Figure 3, the relation between *k* and decryption process is depicted in Figure 4.

Table 1. Running Time RPrime RSA (Key generation, key encryption, and key decryption) 16 Characters

| *k* | Key Generation (ms) | Encryption(ms) | Decryption(ms) |
|---|---|---|---|
| 2 | 15.799999237060547 | 0.05793571472167969 | 0.049114227294921875 |
| 3 | 75.1960277557373 | 0.0591278076171875 | 0.05316734313964844 |
| 4 | 363.8792037963867 | 0.07390975952148438 | 0.06389617919921875 |
| 5 | 4553.959131240845 | 0.09584426879882812 | 0.09817413330078125 |

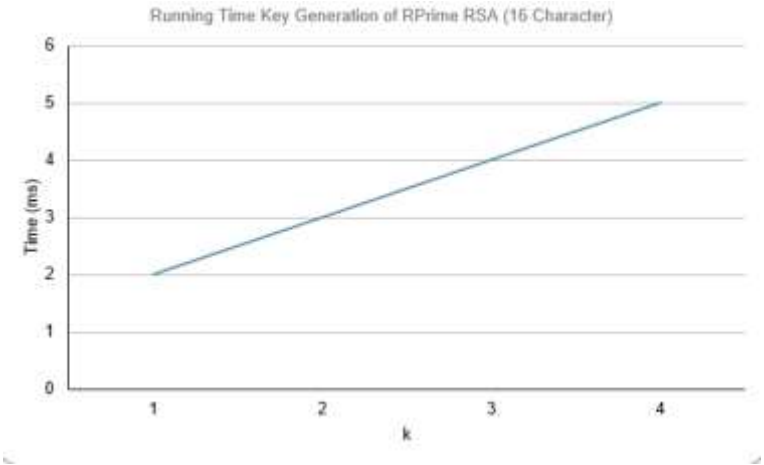


Figure 2. Running Time Key Generation of RPrime RSA (16 Character)

It can be seen in Figure 2, the key generation process time is proportional to the size of $k$ in RPrime RSA.

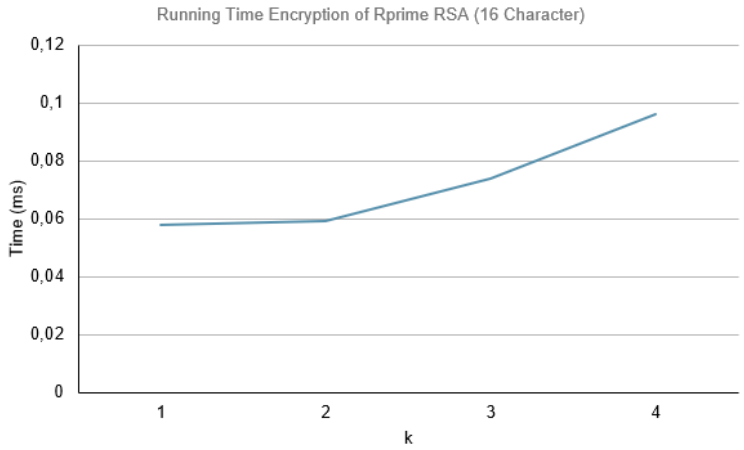


Figure 3. Running Time Key Encryption of RPrime RSA (16 Character)

In Figure 3 depicted that the encryption process time is proportional to the size of *k* in RPrime RSA with 16 characters.

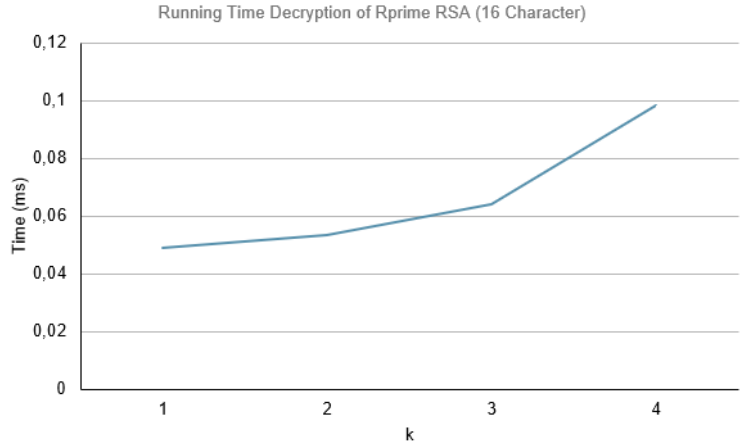


Figure 4. Running Time Key Decryption of RPrime RSA (16 Character)

It can be seen in Figure 4, the decryption process time is proportional to the size of *k* in RPrime RSA with 16 characters.

The results of running time with XTEA to encrypt and decrypt the message is shown in table 2, the relation between encryption message process and character length with XTEA depicted in Figure 5, the relation between decryption message process and character length depicted in Figure 6.

*Table 2. Running Time  XTEA (plain text message encryption)*

| Character Length | Encryption (ms) | Decryption (ms) |
|---|---|---|
| 10 | 0.06604194641113281 | 0.06318092346191406 |
| 100 | 0.39505958557128906 | 0.2129077911376953 |
| 1000 | 3.0260086059570312 | 1.6632080078125 |
| 10000 | 30.39717674255371 | 15.347957611083984 |
| 100000 | 292.49119758605957 | 149.37996864318848 |

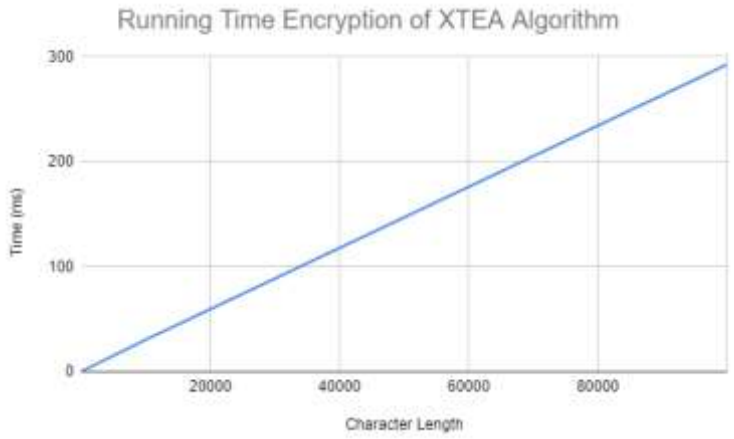


Figure 5. Running Time Key Encryption of XTEA Algorithm

It can be seen in Figure 5, the encryption process time is proportional to the size of character length in XTEA Algorithm.

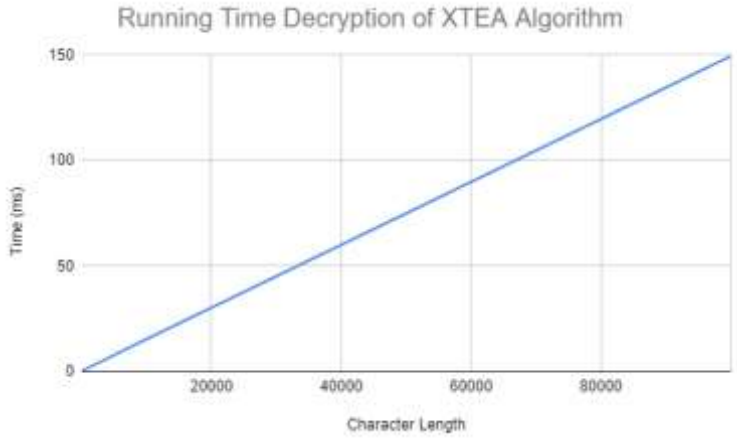


Figure 6. Running Time Key Decryption XTEA Algorithm

In Figure 6 depicted that the encryption process time is proportional to the size of character length in XTEA Algorithm.

The results of running time with Kraitchik factorization process is shown in table 3, we can see in column *k* is the value of *k*, in the column n, p and d are the values of public key n, and factorized private key p and private key *d*, in column $p_1, p_2 \ldots p_k$ (ms)is the running time process to get private key *p*1,*p*2..*p*k in millisecond and in column d (ms) is the value to get private key d in millisecond , the relation between k and $p_1, p_2, \ldots p_k$ factorization with Kraitchik factorization process depicted in Figure 7, the relation between k and d factorization with Kraitchik factorization process Figure 8.

*Table 3. Running Time Kraitchik Factorization (Modulus Factorization* for private key*)*

| *k* | *n*, *p* and *d* | $p_1, p_2 \ldots p_k$ (ms) | *d* (ms) |
|---|---|---|---|
| 2 | *n*= 267<br>*p*= [89,3]<br>*d*= 111 | 0.03790855407714844 | 0.1759529114 |
| 3 | *n*= 5727<br>*p*= [83, 23, 3]<br>*d*= 219 | 0.4799365997314453 | 0.4909038544 |
| 4 | *n*= 3814763<br>*p*= [83, 41, 19, 59]<br>*d*= 158129 | 1.2998580932617188 | 1.348018646240234 |
| 5 | *n*= 10391040<br>*p*= [3, 61, 83, 67, 17] | 4.768848419189453 | 4.804849624633789 |

| *k* | *n*, *p* and *d* | $p_1, p_2 \ldots p_k$ (ms) | *d* (ms) |
|---|---|---|---|
| 6 | *d*= 151<br>*n*= 4627623<br>*p*= [67, 7, 3, 13, 23, 11]<br>*d* = 241 | 21.602869033813477 | 21.657943725585938 |

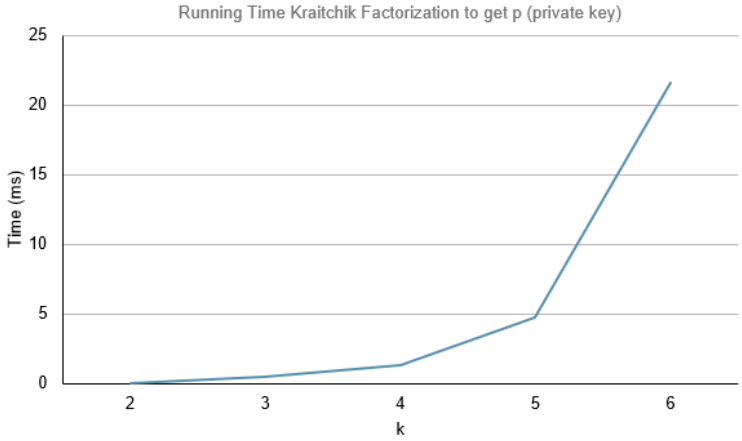


Figure 7. Running Time Kraitchik Factorization to get p (private key)

In Figure 7 depicted that factorization to get $p_1, p_2, .. p_k$ process time in Kraitchik factorization is proportional to the size of $k$.

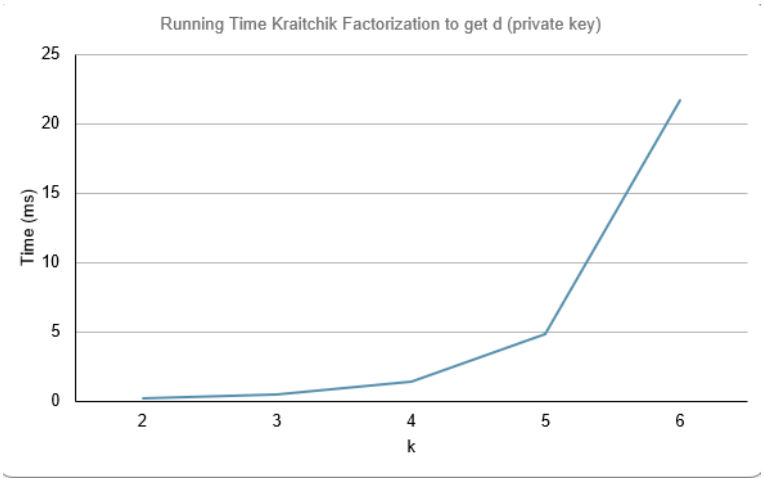


Figure 8. Running Time Kraitchik Factorization to get d (private key)

In Figure 8 depicted that factorization to get $d$ process time in Kraitchik factorization is proportional to the size of $k$.

## 5. Conclusion

Based on the results of testing and analysis in this research using the Hybrid Combination method on the RPrime RSA and XTEA (extended tiny encryption) algorithms, the following conclusions are drawn:

1. Analysis of the Hybrid method on the Rprime RSA and XTEA Algorithms was successfully conducted on plaintext consisting of a collection of strings containing uppercase letters, lowercase letters, numeric digits, and punctuation characters with different characters and string lengths.
2. During the key generation, encryption, and decryption processes in the Rprime RSA algorithm for encrypting the key used to encrypt XTEA messages, security in transmitting the key over public networks was enhanced. Moreover, the message encryption process became relatively fast with the XTEA algorithm.
3. Analysis of the time taken for key generation, encryption, and decryption using the Rprime RSA and XTEA algorithms resulted in values that are directly proportional to the number of keys generated and

the number of characters in the encrypted message. The larger the number of keys or the number of characters in the message, the longer the real running time for key generation, encryption, and decryption processes.

4. Analysis of Modulus Factorization to get private key with Kraitchiik Algortihm. resulted in values that are directly proportional to the generated modulo from value of $k$ and factorized $d$ *and* $p_1, p_2, .. p_k$ (private key). The larger the value of $k$, the more difficult and time consuming the factorization process becomes to obtain the private key.

## REFERENCES

[1] Adriaanse, A Comparative Study of the TEA, XTEA, PRESENT and Simon lightweight cryptographicschemes, Bachelor Seminar of Computer Science and Engineering.
[2] D. M. Bressoud, "Factorization and primality testing", Springer Science & Business Media, 2012.
[3] M.S. Lydia, M.A. Budiman, and D. Rachmawati, "Factorization of Small RPrime RSA Modulus Using Fermat's Difference of Squares and Kraitchik's Algorithms in Python," *Journal of Theoritical and Applied Information Technology*, vol. 99, no.11, pp.2770-2779, 2021.
[4] N. Smart, "Cryptography: An Introduction", 3rd Ed., Bristol City, 1999.
[5] Paixao, C. A. Monteiro, and D. L. G. Filho, "An efficient variant of the RSA cryptosystem," IACR Cryptology ePrint Archive 2003 : 159.
[6] D. Rachmawati, A. Sharif, Jaysilen, and M. A, "Hybrid Cryptosystem Using Tiny Encryption Algorithm and LUC Algorithm". *IOP Conference Series: Materials Science and Engineering*, vol. 300, no.1, pp.1-7, 2018.
[7] R. M. Needham and D. J. Wheeler, "Tea extensions". Report, Cambridge University, Cambridge, UK, 1997.
[8] I. Syamsuddin, S. Ihdianty, E. Tungadi, Kasim, and Irawan, "XTEA Cryptography Implementation in Android Chatting APP," *Journal of Information Technology and Its Utilization*, vol.3, no. 2, pp.36-43, 2020.