International Journal on Advanced Science Engineering Information Technology

Vol.11 (2021) No. 6
ISSN: 2088-5334

Eight Prime Numbers of Modified RSA Algorithm Method for More Secure Single Board Computer Implementation

Nanang Triagung Edi Hermawan [a], Edi Winarko [b,*], Ahmad Ashari [b]

[a] *Indonesian Nuclear Energy Regulatory Agency, Jakarta, 10120, Indonesia*
[b] *Department of Computer Science and Electronics, Universitas Gadjah Mada, Bulaksumur, Yogyakarta, 55281, Indonesia*
*Corresponding author: *ewinarko@ugm.ac.id*

Abstract— **RSA is the most popular public-key cryptography. The main strength of the algorithm is based on the difficulty of factoring in a large integer number. RSA has also been applied in a system with limited resource environments like single-board computers (SBC). To ensure data security, a recommendation to use a key size longer than 2048 bits generates challenges for implementing RSA in the SBC. This research proposes an EPNR (Eight Prime Numbers of Modified RSA) method, a modified double RSA based on eight prime numbers combined with the CRT method, to speed up the random key generation and decryption mechanism. The method is implemented in a Raspberry Pi 4 Model B+. The running time and security performances of the EPNR were analyzed and compared to the other models. Compared to the others model based on the standard RSA scheme, the proposed model is faster 21.78 times in a random key generation, 9.03 times in encryption and decryption processing. The EPNR has resistance to Wiener, statistical, and factorization attacks (GNFS and Fermat). Using standard RSA in the second encryption mechanism, the GNFS is not yet effective for attacking the proposed model. The modified Fermat Factorization algorithm is more difficult and needed more extra times for factoring a large composite number into eight prime numbers correctly. The method will be useful for implementing certificates authentication and distribution of the secret key. It is very suitable to enhance more secure RSA implementation in an SBC environment.**

Keywords— **RSA algorithm; multi-prime numbers; single-board computer; information security.**

Manuscript received 12 Nov. 2020; revised 12 Jun. 2021; accepted 24 Jul. 2021. Date of publication 31 Dec. 2021.

I. INTRODUCTION

Data and information are essential assets; therefore, their security aspects should be guaranteed. Cryptography techniques can be implemented to ensure security aspects of data, namely: confidentiality, integrity, and authentication [1] [2]. As one of the popular asymmetric cryptography methods, the RSA algorithm was developed by Rivest-Shamir-Adleman in 1978 [1]. The standard RSA algorithm generates a pair of keys based on two random prime numbers. The security level of the RSA algorithm depends on how difficult to factorize a big integer number that is used as a modulus computation into two prime numbers. With the improvement of computational capabilities, the key size should be longer than 1024 bits [1], [3], [4]. Using longer than 2048 bits is also recommended [5], [6]. Modulus factorization is the most common type of RSA cryptosystem attack. The RSA algorithm with a key size shorter than 1024 bits is easy to be broken by some factorization algorithms, such as trial division, Pollard Rho, Fermat factorization, Euler factorization, and Quadratic Sieve algorithms [7]–[9]. The Quadratic Sieve, the Number Field Sieve [10], and the Elliptic Curve Factoring Algorithm [11] are considered to be the best and most effective factoring algorithms on very large numbers [5], [12].

The RSA algorithm can be divided into three phases: random key generation (public key and private key), encryption, and decryption phases. The random key generation and decryption phases need more resources and time compared to the encryption phase. These phenomena are very critical in terms of limited or constraint systems related to speed computational, memory, and power supply, such as in a single-board computer (SBC) [13]–[16]. Some SBC samples with limited resources are implemented as a simple sensor with minicomputer-instrument for system monitoring in a remote territory, such as in smart grid line monitoring [17], [18], [19], weather monitoring [20], [21], wireless body area network [22]–[25], smart city [26], internet of things [27] and radiation monitoring [28]–[30]. Security measurement should be conducted in the transmission of monitoring data by considering the site's resources limitation and implementing suitable cryptography techniques. Especially in RSA implementation, the multi-prime numbers technique can generate keys faster than only uses two prime numbers, like in the standard RSA algorithm [31]–[33]. On the other side, the Chinese Remainder Theorem (CRT) can be implemented to increase the acceleration of the decryption process [1], [22], and [34].

Many studies have been done to improve the performance of the RSA. These studies mainly focus on improving speed execution and security level, especially from the factorization attack. Some modification implemented methods to achieve speed improvement by modifying repetitive modular multiplication and modular exponentiation for suitable modern hardware [35]. Another applied two different keys pairs in brute force attack and probability bit different keys pairs in brute force attack. Other researches improved probability bit n RSA-PAKE protocol, two different size keys, multiple public keys, and multi-prime [5].

In speed execution issues of RSA algorithm, the researchers tend to focus on increasing the key generation and decryption process speed. A combination of multi-prime numbers and the CRT methods are used for achieving the goals [31]. The modified RSA algorithms usually use more than two prime numbers than standard RSA that uses two prime numbers [36]. On the other side, the CRT method is used to shorten the decryption exponent bit size by hiding incompatible systems to have a better speed in decryption processing [37].

Other studies have generated some set of public and its corresponded private keys from two prime numbers [38] and [39]. In the transmission process, the public key is disguised or blinded by another number to improve confidentiality. The mathematical operation, with the same modulo in the system, generates vulnerability from modulus attacks. Other modified RSA by adding two random numbers and uses two different modulo have been proposed [40]. Implementation of small private keys size is the security vulnerability hole from the Wiener attack [35].

Dual RSA with two pairs of public and private keys to improving the standard RSA's security has been performed [41], [42]. The proposed dual RSA also implemented CRT to speed up the decryption process [43]. RSA modifications by utilization of multi public ($w_1, w_2, w_3, \ldots, w_k, n$) and private ($s_1, s_2, s_3, \ldots, s_k, n$) key pairs based on two prime numbers (p, and q), also has been observed [44]. Each key pair is used for each correlated segment of big matrices, corresponding to the matrices of images. It is operated as big plain matrices to produce cipher matrices. This method is suitable for encrypting images or video files.

Additionally, some studies have modified RSA algorithm by using three prime numbers ($p, q,$ and r) [45], [46], and [47]. The simulation result in [45] proves that the key generation was faster compared to standard RSA and it enhances security by two levels in a modified RSA algorithm. Three Mersenne prime numbers to construct a new RSA cryptosystem, which provides more efficiency and reliability over the network, have also been observed [46]. The Zaid and Hassan [47] results indicate that the average speed improvement is about 80% in a key generation process, more or less 96% in the decryption process, and only 4% in the encryption process. This is reached by a combination of multi primes and CRT methods.

An asymmetric Key-based Cryptographic Algorithm using Four Prime Numbers (ACAFP) to secure message communication has been proposed [48]. Four prime numbers ($w, x, y,$ and z) are not easily disintegrated and increased the networks' effectiveness. The proposed scheme is much advanced in terms of memory consumption and computational speed. Dual Modulus RSA, based on four prime numbers and Jordan-Totient function, has been developed [42]. However, both papers do not utilize the CRT in the decryption process.

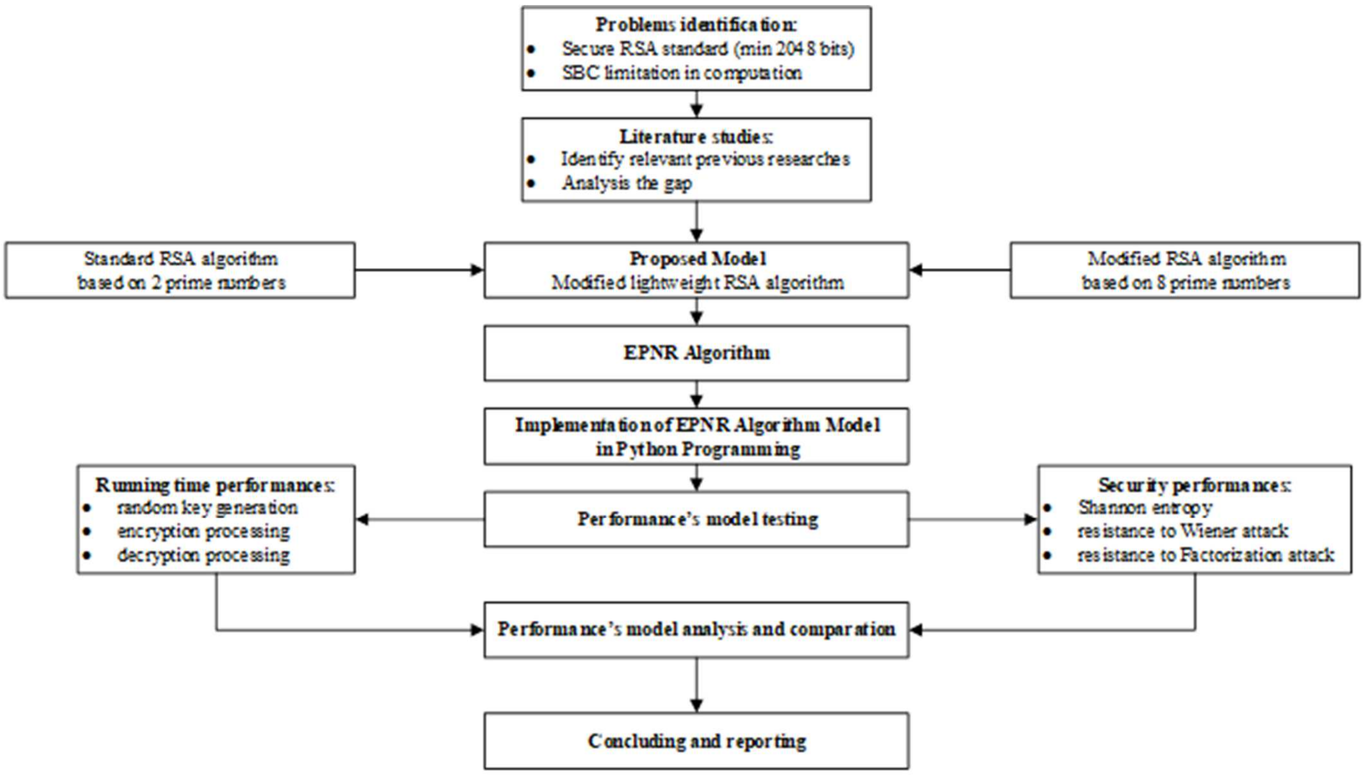
The hybrid RSA method was also proposed based on four prime numbers [49], [49]. The public and private keys are generated by implementing a modification to Euler's totient component. In the encryption and decryption process, the modulus is hidden by some random number related to the real modulus. The encryption and decryption processes are faster than the standard RSA, but processing for key generation is slower [50].

Modified RSA by four [51] and by five [52] prime numbers combined with the CRT method has been observed to enhancement the speed of the decryption mechanism. More generally, RSA modification based on multi-prime numbers by secretly keys sharing has been observed [12], [53]. Its combination with CRT also has been analyzed [31]. Two secret keys are used and combined in the calculation to improve complexity in the encryption and decryption mechanism. Secure secret keys distribution is needed seriously in this system.

Based on the above discussions, implementing the RSA algorithm in an SBC environment with limited computational resources has not been studied deeply. Multi-prime numbers combined with the CRT method will be very suitable and useful to implement in this condition. Therefore, this research proposes a method for improving the RSA algorithm implemented in a more secure SBC. The improvement is made by implementing multi-primes and the CRT based on a double encryption/decryption mechanism. In our case, the SBC will be installed in the Radiation Data Monitoring System (RDMS) device used to monitor the level of radiation in its surrounding area [28]. The function of SBC is to increase the security of the RDMS data, especially for authenticating the entity of new RDMS, secret key distribution, and encrypting the monitoring data that is sent from RDMS device to the server.

II. Materials and Method

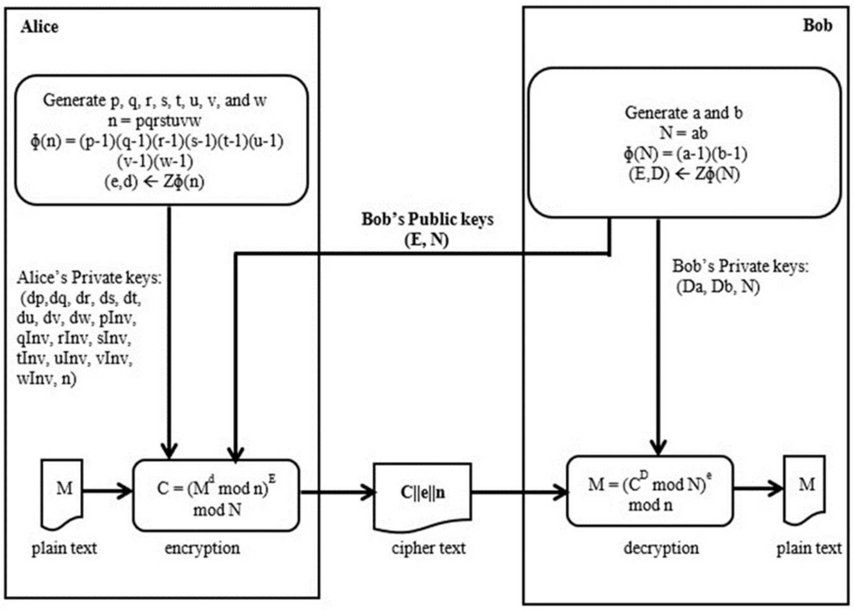
This section discusses in detail the proposed EPNR method, which is the modified version of the Dual RSA algorithm. The motivation of EPNR development is that it will run on the hardware or system with limited resources, such as the SBC. RSA Cryptosystem implementation in a long key that is longer than 2048 bits [5] to reach minimal security requirements hugely influences SBC performance. Based on the two main problems above, we conducted research method in Figure to develop an EPNR algorithm model.



Problems identification:
Secure RSA standard (min 2048 bits)
SBC limitation in computation
Literature studies:
Identify relevant previous researches
Analysis the gap
Standard RSA algorithm based on 2 prime numbers
Proposed Model
Modified lightweight RSA algorithm
Modified RSA algorithm based on 8 prime numbers
EPNR Algorithm
Implementation of EPNR Algorithm Model in Python Programming
Running time performances:
random key generation
encryption processing
decryption processing
Performance's model testing
Security performances:
Shannon entropy
resistance to Wiener attack
resistance to Factorization attack
Performance's model analysis and comparation
Concluding and reporting

Fig. 1 Research method

The method includes problem identification, literature studies, proposed EPNR model, EPNR implementation in Python programming, performance model testing and analysis, followed by concluding and reporting. Fig.2 shows the main diagram of an EPNR algorithm. Multi primes and CRT are combined as the EPNR algorithm for more secure SBC implementation is proposed in this to reach the goals.



Alice
Bob
Generate p, q, r, s, t, u, v, and w
n = pqrstuvw
ϕ(n) = (p-1)(q-1)(r-1)(s-1)(t-1)(u-1)(v-1)(w-1)
(e,d) ← Zϕ(n)
Generate a and b
N = ab
ϕ(N) = (a-1)(b-1)
(E,D) ← Zϕ(N)
Bob's Public keys (E, N)
Alice's Private keys: (dp,dq, dr, ds, dt, du, dv, dw, pInv, qInv, rInv, sInv, tInv, uInv, vInv, wInv, n)
Bob's Private keys: (Da, Db, N)
M
plain text
C = (M^d mod n)^E mod N
encryption
C||e||n
cipher text
M = (C^D mod N)^e mod n
decryption
M
plain text

Fig. 2 The main diagram of an EPNR algorithm

Based on the process as shown in Fig. 2, Bob generates a public and private key (E, D) based on two strong prime numbers (*a*, and *b*) as standard RSA in the first time. Bob also computes the private key's exponents (*Da,* and *Db*) and their inverses (*aInv,* and *bInv)*. The public key components (E, N) are sent to Alice. Alice generates a public and private key based on eight strong prime numbers (*p, q, r, s, t, u, v,* and *w*). Alice also calculates each the private key's exponents (*dp, dq, dr, ds, dt, du, dv,* and *dw*) and their inverses (*pInv, qInv, rInv, sInv, tInv, uInv, vInv,* and *wInv*).

Alice uses her private exponents and Bob's public key to encrypt a plain text (P) and send Bob's result (C). After receiving the ciphertext, Bob uses his private key (*D, N*) and Alice's public key (e, n) in the decryption processing to get the plain text. The method also implements the CRT method with eight private key exponents on Alice's side and two private key exponents on Bob's side. By the double mechanism, the random key generation and encryption based on EPNR method in Alice's side will be faster. It also can guaranty authentication and confidentiality message transmission from Alice to Bob. The mechanism is very suitable for certificates authentication of new RDMS's entity, secret key distribution, and encrypting the monitoring data that is sent from RDMS device to the server. More detail about the generation of a public key pair, encryption of the plain message, and decryption of ciphertext is given in Algorithm 1, 2, and 3, respectively.

In Algorithm 1, the proposed method uses two strong prime numbers (*a* and *b*) to generate public (PU_B) and private keys (PR_B) in Bob's side. Bob also computes the private key's exponents (*Da, and Db*) and their inverses (*aInv, and bInv*). In another side, Alice uses eight-strong prime numbers (*p, q, r, s, t, u, v,* and *w*) to generate Alice's public (PU_A) and private keys (PR_A). In order to generate Alice's private keys, the algorithm computes eight private key exponents (d_p, d_q, d_r, d_s, d_t, d_u, d_v, and d_w) and their inverses (*pInv, qInv, rInv, sInv, tInv,uInv, vInv,* and *wInv*). During the double encryption (Algorithm 2), firstly, a plain message is encrypted using Alice's private key and, secondly, using Bob's public key to produce ciphertext (*C*). In double decryption processing (Algorithm 3), Bob uses his private key in the first decryption. The final message has resulted from the second decryption with Alice's public key.

The CRT method is used for breaking a large integer of each private key into smaller exponent to simplify exponentiation computation both in encryption and decryption processing. By a double encryption/decryption mechanism, there are two layers of security. The first or outer layer of security is an encryption/decryption mechanism using Bob's public and private keys. The second or inner security layer is an encryption/decryption mechanism using Alice's public and private keys. These mechanisms are secure defense in depth implementation in our model.

Algorithm 1: Key generation

```
Bob's side:
 1. Generate random prime numbers: $a_1$, and $b_1$.
 2. Compute and find strong prime numbers:
      $a = 2a_1+1$, and $b = 2b_1+1$.
 3. Compute N as modulus a system: $N = a*b$.
 4. Compute the totient of N: $\phi(N) = (a-1)*(b-1)$.
 5. Choose integer $E$ as a public key, where $1 < E < \phi(N)$
    (randomly chosen from five digits of palindrom prime
    number set).
 6. Find $D$ as a private key, where $E*D \equiv 1 \bmod \phi(N)$.
 7. Calculate a private key exponent:
      $Da = D \bmod (a-1)$, and $Db = D \bmod (b-1)$.
 8. Calculate two private key exponents invers:
      $aInv = a^{-1} \bmod b$, and $bInv = b^{-1} \bmod a$.
 9. Bob's public key: $PU_B = \{E, N\}$.
 10. Bob's private key: $PR_B = \{Da, Db, aInv, bInv$, and $N\}$
Alice's side:
 1. Generate random prime numbers:
      $p_1, q_1, r_1, s_1, t_1, u_1, v_1$, and $w_1$.
 2. Compute and find strong prime numbers:
      $p = 2p_1+1, q = 2q_1+1, r = 2r_1+1, s = 2s_1+1, t = 2t_1+1, u = 2u_1+1, v = 2v_1+1$, and $w = 2w_1+1$.
 3. Calculate n as modulus a system: $n = p*q*r*s*t*u*v*w$.
 4. Calculate the totient of $n$:
      $\phi(n) = (p-1)*(q-1)*(r-1)*(s-1)*(t-1)*(u-1)*(v-1)*(w-1)$.
 5. Choose integer $e$ as a public key, where $1 < e < \varphi(n)$ (fixed
    setting to 65537).
 6. Find integer $d$ as a private key, where $e*d \equiv 1 \bmod \phi(n)$.
 7. Compute eight private key exponents:
      $dp = d \bmod (p-1)$; $dq = d \bmod (q-1)$; $dr = d \bmod (r-1)$;
      $ds = d \bmod (s-1)$; $dt = d \bmod (t-1)$; $du = d \bmod (u-1)$;
      $dv = d \bmod (v-1)$; $dw = d \bmod (w-1)$.
 8. Compute eight private key exponent invers:
      $pInv = p^{-1} \bmod (q*r*s*t*u*v*w)$;
      $qInv = q^{-1} \bmod (p*r*s*t*u*v*w)$;
      $rInv = r^{-1} \bmod (p*q*s*t*u*v*w)$;
      $sInv = s^{-1} \bmod (p*q*r*t*u*v*w)$;
      $tInv = t^{-1} \bmod (p*q*r*s*u*v*w)$;
      $uInv = u^{-1} \bmod (p*q*r*s*t*v*w)$;
      $vInv = v^{-1} \bmod (p*q*r*s*t*u*w)$;
      $wInv = w^{-1} \bmod (p*q*r*s*t*u*v)$.
 9. Alice's public key: $PU_A = \{e, n\}$.
 10. Alice's private keys: $PR_A = \{dp, dq, dr, ds, dt, du, dv, dw,
     pInv, qInv, rInv, sInv, tInv, uInv, vInv, wInv, n\}$.
```

Algorithm 2: Encryption of plain text

```
The encryption applies double encryption mechanism.
 1. In first encryption steps, Alice's eight private key exponents
    and the invers modulation are used to compute separated
    message by implementing CRT method as:
      $m_{1A} = (M^{dp} \bmod p)*(q*r*s*t*u*v*w)*(pInv)$;
      $m_{5A} = (M^{dt} \bmod t)*(p*q*r*s*u*v*w)*(tInv)$;
      $m_{2A} = (M^{dq} \bmod q)*(p*r*s*t*u*v*w)*(qInv)$;
      $m_{6A} = (M^{du} \bmod u)*(p*q*r*s*t*v*w)*(uInv)$;
      $m_{3A} = (M^{dr} \bmod r)*(p*q*s*t*u*v*w)*(rInv)$;
      $m_{7A} = (M^{dv} \bmod v)*(p*q*r*s*t*u*w)*(vInv)$;
      $m_{4A} = (M^{ds} \bmod s)*(p*q*r*t*u*v*w)*(sInv)$;
      $m_{8A} = (M^{dw} \bmod w)*(p*q*r*s*t*u*v)*(wInv)$.
 2. Based on the result of above separated message, compute first
    ciphertext ($C_1$) as:
      $C_1 = (m_{1A}+m_{2A}+m_{3A}+m_{4A}+m_{5A}+m_{6A}+m_{7A}+m_{8A}) \bmod n$
 3. In second encryption step, Bob's public key exponents ($E$, $N$)
    are used to compute a final ciphertext ($C$) as:
      $C = C_1^E \bmod N$
```

Algorithm 3: Decryption of ciphertext

```
 1. In first decryption steps, Bob's two private key exponents and
    the invers modulation are used to compute separated message
    by implementing CRT method as:
      $m_{1B} = (C^{Da} \bmod a)*(b)*(aInv)$
      $m_{2B} = (C^{Db} \bmod b)*(a)*(bInv$
 2. Based on the result of above separated message, compute first
    ciphertext ($C_1$) as:
      $C_1 = (m_{1B}+m_{2B}) \bmod N$
 3. In second decryption step, Alice's public key exponents ($e, n$)
    are used to compute an original plaint text ($M$) as:
      $M = C_1^e \bmod n$
```

In order to evaluate the performance of the EPNR method, we compare it to the standard RSA and a modified RSA with four prime numbers (the ACAFP). We implement these three methods in Python 3.8 and run them in a Raspberry Pi 4 Model B. The Raspberry's specifications are Cortex-A72 processor, 1.2GHz Quad-Core ARM Cortex-A53 (64Bit)

802.11 b/g/n, 4.00 GB LPDDR4 memory, and Raspbian Operating System.

We performed two sets of experiments. The first set of experiments is to compare the running time of the three methods on random key generation, encryption, and decryption steps. In these experiments, we varied the size of the key (in bits), that is, 800, 1024, 1600, 2048, 3200, 4096, 5000, 6000, 7000, and 8192 bits. The second set of experiments is to compare the robustness of the three methods from statistical and factorization attack. We use Shannon Entropy values, and the modified Fermat factorization algorithm to analyze the robustness of the three methods from these two attacks.

III. Results and Discussion

This section presents and discusses the computational (running time) and security performance of our proposed EPNR method.

A. Running Time Comparison

Computational performances of the proposed method, which includes running time of random key generation, encryption, and decryption processes, are discussed here. Based on the duration of a key pair generation process for all of its bits size, respectively, the processing time is shown in Fig. 3 and Fig. 4. It can be compared more rigid, increasing speed up on the process based on the utilization of multi-prime numbers for the same key size. Generally, it can be seen that for increasing the longer size of keys or system modulus (n), the time processing to get the RSA key pair will also increase exponentially for all of the methods.

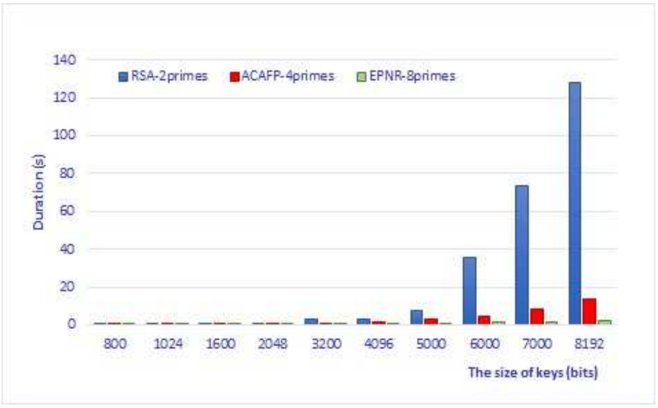


Fig. 3 The running time comparison of the random key generation

In the same size of keys, our proposed method is the fastest in key generation processing. A smaller running time means a faster process. It is caused by generating a shorter bit prime number when the multi-prime method was implemented. For example, for generating 8192 bits, two of 4096 bits are needed in standard RSA, and four of 2048 bits prime numbers are needed in the ACAFP method.

For the same operation, our proposed EPNR only needs to generate eight of 1024 bits prime numbers. Actually, generate eight of 1024 bits prime numbers is faster than generate two of 4096 bits or four of 2048 bits prime numbers. These conditions correlate closely to the primality check mechanism by executing such an algorithm as Miller-Rabin. Primality checking is very dependent on bits long. It indicates that the EPNR is the best approach.

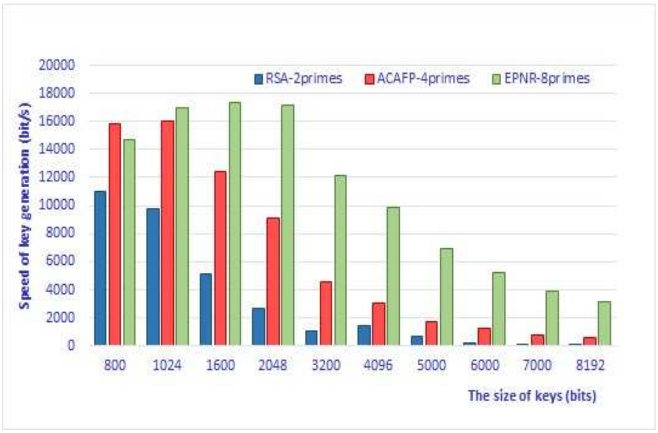


Fig. 4 The speed comparison of random key generation

Fig. 4 shows the speed of random key generation of the three methods. The speed information can indicate an optimum of the key size in their generation process. The speed v ($bits/s$) is formulated as:

$$v = \frac{b}{t} \tag{1}$$

where b is the size of the key (bits), and t is the time of key generation (*second*). The speed performances of random keys generation of the proposed method are compared to standard RSA and the ACAFP. It can be known that by increasing key size, the speed of key generation for standard RSA always decreases exponentially. This condition is hugely different compared to EPNR that from 1024 to 2048 bits of key size relatively little bit the same performance. For EPNR, from 3200 to 8192 bits of key, the speed decreases exponentially. It can be concluded that in the key generation of prime numbers, the EPNR is optimum in range 1024 to 2048 bits of key size. It is a secure range of RSA algorithm. However, so that security is always guaranteed we recommend using a key length of more than 2048 for both Alice and Bob side.

In all conditions of key size, the EPNR method's speed is always the fastest compared to ACAFP and standard RSA. By increasing the length of key bits, the speed decreases, but the speed ratio increase exponentially. Our proposed EPNR method has the highest speed ratio compared to others (in the average 21.78 compared to standard RSA and 5.76 compared to the ACAFP). It shows that eight multi-prime numbers can increase the key generation process significantly.

Alice's public key is set to a fixed public key 65537 for all experiments. The 65537 number is chosen based on reasoning recommendations for achieving a minimum security level, and it is the most common choice number [1]. In another side, Bob's public key is randomly chosen from five digits palindrome prime number set to improve complexity. An ID Code of the Radiation Data Monitoring System (example SARA01SRPAGS711S21087) was used as plain text in the experiments. The ID Code should convert into an integer number before being encrypted. The plain text is encrypted firstly using Alice's private key and then secondly using Bob's public key. Fig. 5 shows the duration of each encryption processing.

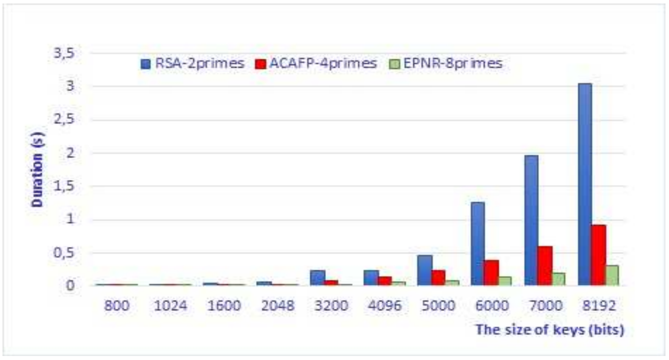


Fig. 5 Running time comparison of the encryption process

Fig. 5 describes that all durations of encryption exponentially increase by increasing of key size. The graph trendlines are relatively identic with the graph trendlines of key generation. It is caused by implementation of a double encryption mechanism. First encryption using Alice's private key needs more extra-time compared to the second encryption using Bob's public key. It can be known that the Alice's private key always extra bigger than all possible Bob's public key. The message will only be useful for the receiver if he/she decrypt the ciphertext into original plain text. Besides the accuracy of the message, the decryption mechanism itself is very important. Fig. 6 describes the complete time duration of the decryption processing that conducted by Bob. In all RSA systems, the decryption process always needs extra time. This phenomenon is different in a double RSA scheme. In a double RSA scheme, there are two pairs of random keys (one random key pair from the sender, and another from the receiver). Both in encryption and decryption process, there are two identic exponentiation calculations but in opposite order. Firstly, exponentiation calculation with a public key, and the second is exponentiation with a private key. This fact can be observed by comparing graph in Fig. 5 and Fig. 6.

Both in encryption and decryption processing, the CRT method is the best approach to solve how to speed up the exponentiation calculation process that uses the private key components. Previous research on improving the decryption processing in RSA only divides private key into two or four exponents. This proposed method uses eight exponents of private keys for implementing the CRT method in Alice's side and two exponents in Bob's side.

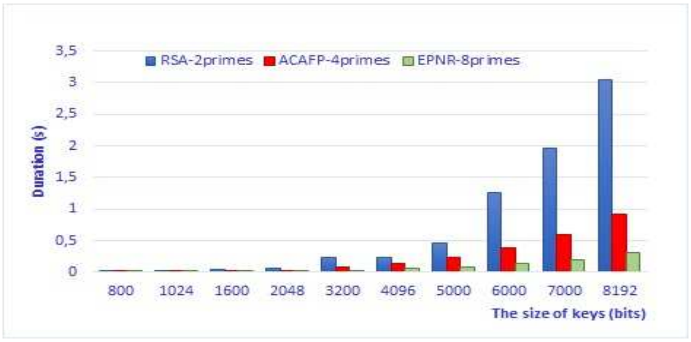


Fig. 6 Running time comparison of the decryption process

In encryption and decryption for all conditions of key size, the EPNR method's duration time of process is always the smallest compared to ACAFP and standard RSA. By increasing the length of key bits, both the duration time and duration time ratio increases exponentially. Our proposed EPNR method has the highest speed ratio compared to others (in the average 9.03 compared to standard RSA and 6.47 compared to the ACAFP). It shows that eight multi-prime numbers can increase the encryption and decryption process significantly.

Hinek formula has provided the iteration or bit operation estimation for decryption processing related to multi-prime numbers implementation, that: $t_{estimated} = \frac{3}{2r^3}(log_2 n)^3$, when r is the number of primes in the n modulo. It can be concluded that for more number of primes used in the modulus, the fewer iterations required for decryption computation [31]. Related to the maximum speed up ratio on decryption process (a), Joseph in [54] has estimated by $a = \frac{3}{2}m.n^3$ formula, where m is the number of primes in the n modulo. In comparison, uses 2, 4, 8, 16, and 32 prime numbers, speed up ratio will be 4, 16, 64, 256, and 1024. In the general formula, it presented as $a = 2^k$, with k = *2, 4, 6, 8,* and *10*. In the context of double RSA mechanism, the two above formulas still work related to modular exponentiation calculation with a private key component.

The experiment also describes that our proposed method by using eight exponents to implement the CRT method in Alice's side resulted the fastest speed on random key generation and encryption processes. It is proofing that our proposed method can improve the speed of the process in SBC faster and better without having to override the security aspect by implementing a double encryption/decryption mechanism.

B. Security Comparison

The security performances of the proposed method, that is, a randomized level of ciphertext and its attack resistances, are discussed here. Bit's randomization level of ciphertext represented by Shannon Entropy value H(x). The ciphertext with a higher Shannon Entropy value will be more statistically difficult to decompose by an attacker. The value *was* calculated *in reference to previous studies [55]–[58]*:

$$H(x) = -\sum_{i=1}^{n} p(x_i) log_2 p(x_i) \qquad (2)$$

where p(x_i) is a probability of such character appear*ing in all sets of total characters* observed (n). Table I describes the determination of Shannon Entropy values based on multi-prime numbers variation in integer and ASCII format of ciphertext.

TABLE I
THE SHANNON ENTROPY VALUE OF CIPHERTEXT (BITS)

Size of keys (bits)	**RSA (2 primes)**		**ACAFP (4 primes)**		**EPNR (8 primes)**	
	Integer	ASCII	Integer	ASCII	Integer	ASCII
800	3.30	5.4	3.29	5.4	3.29	5.4
1024	3.30	5.59	3.30	5.6	3.30	5.61
1600	3.31	5.96	3.31	5.95	3.31	5.96
2048	3.31	6.12	3.31	6.11	3.31	6.1
3200	3.32	6.31	3.32	6.31	3.31	6.32
4096	3.32	6.4	3.32	6.39	3.32	6.39
5000	3.32	6.44	3.32	6.44	3.32	6.44
6000	3.32	6.48	3.32	6.47	3.32	6.47
7000	3.32	6.51	3.32	6.5	3.32	6.5
8192	3.32	6.53	3.32	6.52	3.32	6.52

Based on the data in Table I, there are not significantly different Shannon Entropy values for ciphertext in integer format. It is mean that there is not significant influencing of multi-prime numbers used with the values into Shannon Entropy values of the ciphertext.

In the other fact that the average value is low, it is caused by the character representation of ciphertext in integer format only using decimal integer number from 0 to 9. The values can be increased by using more complex character representation, likes ASCII codes. This method can increase an average of the Shannon Entropy values up to 1.86 times. By increasing character complexity, the resistant of related ciphertexts from statistical attack also will increase.

The security of the RSA depends on computationally infeasible for an intruder to factorize large integer *(n)* into its prime number components. This research uses eight components (*p, q, r, s, t, u, v,* and *w*) in Alice's side that the strength of a huge prime number relied on. It is regarded to be hard to break the vast number into eight prime factors. The above EPNR scheme is still added with standard RSA as the second layer of security. It is the key point of security from our proposed method.

Regarding on private key using in standard RSA algorithm, [35] and [59] recommend to use a big integer of a private key ($d \geq \frac{1}{3}n^{1/4}$) for avoiding Wiener attack. Another research, give suggestions to use $d \geq \sqrt{\frac{2N^3-18N^2}{e}}$ [60] and $d \geq \sqrt{t(2\sqrt{2}+8/3)}N^{0.75}/\sqrt{e}$ [16], where *t* is chosen parameter. Along with *n* as a modulus or key size increasing, *d* will also increase into a very big integer. Especially in a double RSA scheme, [61] recommend a private key ($d \geq n^{0.368}$) for avoiding factorization attack based on lattice basis reduction algorithm. In our proposed model execution, the above minimum of private key size requirements always be fulfilled.

In principle, existing factorization algorithms are formulated based on the assumption that RSA's modulus generates based on two prime numbers. By the above assumption, our proposed method uses more than 2048 bits and is based on eight prime numbers with double encryption/decryption mechanism is secure from all existing factorization attacks.

The most efficient of factorization attack to standard RSA algorithm is the General Number Field Sieve (GNFS) method [10], [16], [62]–[65]. Granger *et al.* [66] provide an in-depth report in 2009 on the factorization of the 768-bit number RSA-768 by the number field sieve (NSF) factoring method and provide some implications for the RSA [8], [65]. The largest such semi-prime yet factored was RSA-250, an 829-bit number with 250 decimal digits, in February 2020. The total computation time was roughly 2700 core-years of computing using Intel Xeon Gold 6130 at 2.1 GHz [67], [66]. This newest fact further reinforces the assumption that 1024 bit will be factored and will not be secured enough to stand against the factorization attacks [5]. This is the main reason why our EPNR method is kept back up with standard RSA as a double encryption/decryption mechanism. To more confident on avoiding GNFS attack, the scheme must use a standard RSA with a key size greater than 2048 bits in Bob's side.

Our proposed EPNR method implements a double encryption/decryption mechanism. In the encryption processing, the first encryption conducted using modified RSA based on eight multi-prime numbers. In the context of algorithm's resistance to GNFS attack, implementation of eight multi-prime numbers will reduce its resistance to one eighth compared to standard RSA. The standard RSA is still applied to the second encryption to overcome the previous vulnerability. Thus, if the resistance to factorization attack in dual standard RSA scheme is doubled, then our proposed EPNR method only increase by one eighth. By this mechanism, the random key generation and encryption in Alice's side (incidentally on SBC environment) can be accelerated, ensuring security from GNFS factorization attacks.

On the other hand, using eight multi-prime numbers in Alice's side's second security layer makes the difference each prime number is smaller. In case an attacker can break the first layer of security (standard RSA scheme), the system becomes vulnerable to Fermat factorization attack [68], and [69]. The Fermat Factorization algorithm is modified for measuring our model's security resistance from the attack. It factorized a modulus number of *n* into eight prime components in our proposed research. Fig. 7 describes complexity steps to factorize *n* by modified Fermat Factorization.

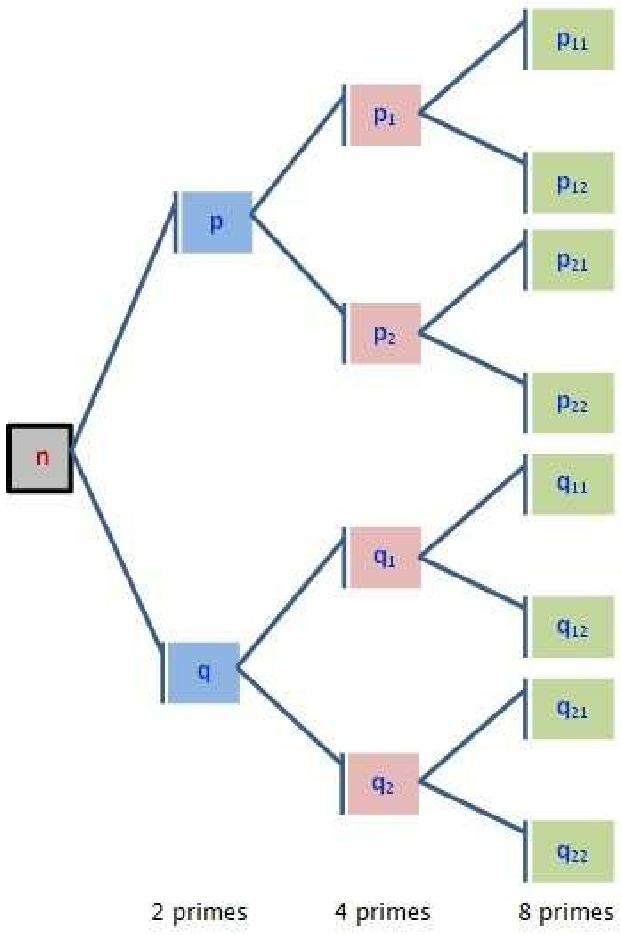


Fig. 7 Modified Fermat Factorization algorithm steps to factorize n

Based on an equation to determine the modulus $n = pq$ in standard RSA system, where p and q assumed as an odd number with small different each other, the above equation is approached with difference equation of two squares as [70], [62], and [68]:

$$n = a^2 - b^2 \tag{3}$$

By repositioning, the new equation formulated as $a^2 - n = b^2$ and then:

$$(a_0 + k)^2 - n = b^2 \tag{4}$$

With initial approaching is started from $a_0 = \sqrt{n}$, brute force iterations are conducted by starting from *k = 0, 1, 2, 3, ...* and so on until getting such a perfect squares number. In that condition, it gets:

$$p = a + b \tag{5a}$$

$$q = a - b \tag{5b}$$

To get eight primes' factors as described in Fig. 6, steps in Equation (4), (5a), and (5b) should be repeated to factorize p into p_1 and p_2, q into q_1 and q_2, p_1 into p_{11} and p_{12}, p_2 into p_{21} and p_{22}, q_1 into q_{11} and q_{12}, and q_2 into q_{21} and q_{22} completely and correctly. Prime number $p_{11}, p_{12}, p_{21}, p_{22}, q_{11}, q_{12}, q_{21}$, and q_{22} equivalent to getting prime numbers *p, q, r, s, t, u, v,* and *w* in our EPNR scheme. The minimum number of all iterations steps *(I)* can be approached by formulation:

$$I = 0.994523\left\{\left(\frac{p-q}{2\sqrt{2q}}\right)^2 + \left(\frac{p_1-q_1}{2\sqrt{2q_1}}\right)^2 + \left(\frac{p_2-q_2}{2\sqrt{2q_2}}\right)^2 + \left(\frac{p_{11}-q_{11}}{2\sqrt{q_{11}}}\right)^2 + \left(\frac{p_{12}-q_{12}}{2\sqrt{q_{12}}}\right)^2 + \left(\frac{p_{21}-q_{21}}{2\sqrt{q_{21}}}\right)^2 + \left(\frac{p_{22}-q_{22}}{2\sqrt{q_{22}}}\right)^2\right\} \tag{6}$$

The constant number 0.994523 granted empirically from experiment. It is a correction number to approach more precise the minimum number of all iteration's steps.

By Equation (6), the total estimated time for factoring *n* can be determined based on the speed of computational iteration. It analyzed that step of factorization intensely depend on different of closer prime numbers. Based on mathematical operation for the same size of short keys, it easy to predict that by implementing eight prime numbers, each closer number will be very closed and has smaller differences. It causes the factorization of modulus into eight prime numbers faster than into two prime numbers. It is very contradictory to improve the security level of the modified algorithm.

By theory and experiment, the difference from each closer prime number should be added to be more significant. It was conducted by implementing a general small random number (non-prime integer) to substitute one of a prime number. It will also be advantageous and efficient if different bits size of eight prime numbers are implemented. A simple example is the implementation of p in 20 bits, q, r, s, t, u in 10 bits, then v and w with 5 bits for 80 bits key size. The above strategy should be implemented based on strong prime numbers as stated in Algorithm 1. Equation (6) can be implemented to estimate iteration operation of factorization for the long size of keys, such as 800 bits, 1024 bits, 1600 bits, 2048 bits, 3200 bits, 4096 bits, 5000 bits, 6000 bits, 7000 bits, and 8192 bits. Table II shows the factorization iteration estimations.

TABLE II
ESTIMATION OF FACTORIZATION ITERATION FOR LONG KEY SIZE

Size of keys (bits)	**RSA (2 primes)**	**ACAFP (4 primes)**	**EPNR (8 primes)**
800	1.06832E+119	4.97808E+118	1.84691E+119
1024	3.44077E+150	5.01439E+152	7.99117E+152
1600	6.82261E+238	6.98096E+239	1.13168E+240
2048	1.88574E+306	8.36656E+306	7.50599E+374
3200	8.07506E+479	1.6116E+480	3.80980E+562
4096	6.47168E+613	3.30105E+614	1.33508E+752
5000	3.64525E+749	1.31847E+751	1.16201E+785
6000	7.40950E+900	1.16131E+901	2.09158E+1005
7000	6.53459E+1050	2.17134E+1051	1.06063E+1161
8192	1.07391E+1231	1.19817E+1232	**Overflow Error**

The Fermat factorization takes high computation cost. It also consumes high computation time to find all prime factors correctly. By double encryption/decryption mechanism using key size longer than 2048 bits and implementing the secure protocol, the EPNR method is very secure from potential factorization attacks.

IV. CONCLUSION

In this paper, we have proposed the EPNR method that modifies the dual RSA algorithm by using eight prime numbers combined with the CRT to increase the speed of random key generation and decryption processes. There are two layers of security based on a double encryption/decryption mechanism. The average speed-up ratio of random key generation can reach 21.78 compared to standard RSA and 5.76 compared to ACAFP. In the decryption process, the average speed-up ratio can reach 9.03 compared to the standard RSA and 6.47 compared to the ACAFP. In terms of data security, the modulus of the EPNR algorithm is more difficult to be factorized. If the first layer of security is broken, more effort and computational resources are needed to factorize a large integer composite number into eight compared to two or four prime number factors in standard RSA and the ACAFP.

By implementing a modified Fermat Factorization algorithm, it needed more extra times and iterations to attack the system for factoring *n* until finding each eight prime numbers correctly. Security analysis by implementing another factorization attack will be useful to improve the security performances of our proposed algorithm. The proposed method is very suitable for enhancing security and implementing in an environment with limited resources, like SBC. A more comprehensive application of the double EPNR method as a communication protocol for mutual authentication of identity and distribution of secret keys to support the Radiation Data Monitoring System in Nuclear Installation or Radiation Facility will be studied in the next research.

ACKNOWLEDGMENT

We are grateful to the Indonesian Nuclear Regulatory Agency that supported our research by the Indonesian Nuclear Regulatory Agency [memorandum of understanding numbers 01/KS 00 01/Set-PKS/II, 2016] scheme.

References

[1] W. Stallings, *Cryptography and Network Security*, Seventh Ed. Singapore: Pearson Prentice Hall, 2017.

[2] A. Shoufan and E. Damiani, "On inter-Rater reliability of information security experts," *J. Inf. Secur. Appl.*, vol. 37, pp. 101–111, 2017, doi: 10.1016/j.jisa.2017.10.006.

[3] M. Mumtaz and L. Ping, "Forty years of attacks on the RSA cryptosystem: A brief survey," *J. Discret. Math. Sci. Cryptogr.*, vol. 22, no. 1, pp. 9–29, Jan. 2019, doi: 10.1080/09720529.2018.1564201.

[4] E. Barker, *Guideline for Using Cryptographic Standards in the Federal Government : Cryptographic Mechanisms NIST Special Publication 800-175B Guideline for Using Cryptographic Standards in the Federal Government : Cryptographic Mechanisms*. USA: NIST U.S. Department of Commerce, 2016, p. 26.

[5] S. S. Al-kaabi and S. B. Belhaouari, "A Survey on Enhanced RSA Algorithms," *J. Comput. Inf. Technol. (CS IT)*, pp. 123–142, 2019, doi: 10.5121/csit.2019.90411.

[6] C. Thirumalai, S. Mohan, and G. Srivastava, "An efficient public key secure scheme for cloud and IoT security," *Comput. Commun.*, vol. 150, no. November 2019, pp. 634–643, 2020, doi: 10.1016/j.comcom.2019.12.015.

[7] A. Overmars, "Survey of RSA Vulnerabilities," *Mod. Cryptogr. - Theory, Technol. Adapt. Integr. [Working Title]*, no. June, 2019, doi: 10.5772/intechopen.84852.

[8] A. Overmars and S. Venkatraman, "Mathematical Attack of RSA by Extending the Sum of Squares of Primes to Factorize a Semi-Prime," *Math. Comput. Appl.*, vol. 25, no. 63, pp. 1–15, 2020, doi: https://doi.org/10.3390/mca25040063.

[9] M. Patel, A. M. Patel, and R. B. Gandhi, "Prime numbers and their analysis," *J. Emerg. Technol. Innov. Res.*, vol. 7, no. March, pp. 1–5, 2020.

[10] L. T. Yang, G. Huang, J. Feng, and L. Xu, "Parallel GNFS algorithm integrated with parallel block Wiedemann algorithm for RSA security in cloud computing," *Inf. Sci. (Ny).*, vol. 387, pp. 254–265, 2017, doi: 10.1016/j.ins.2016.10.017.

[11] A. Nitaj and E. Fouotsa, "A new attack on RSA and Demytko's elliptic curve cryptosystem," *J. Discret. Math. Sci. Cryptogr.*, vol. 22, no. 3, pp. 391–409, Apr. 2019, doi: 10.1080/09720529.2019.1587827.

[12] M. A. Islam, M. A. Islam, N. Islam, and B. Shabnam, "A Modified and Secured RSA Public Key Cryptosystem Based on 'n' Prime Numbers," *J. Comput. Commun.*, vol. 06, no. 03, pp. 78–90, 2018, doi: 10.4236/jcc.2018.63006.

[13] M. Mumtaz and L. Ping, "Remarks on the cryptanalysis of common prime RSA for IoT constrained low power devices," *Inf. Sci. (Ny).*, vol. 538, pp. 54–68, 2020, doi: 10.1016/j.ins.2020.05.075.

[14] P. J. Basford *et al.*, "Performance analysis of single board computer clusters," *Futur. Gener. Comput. Syst.*, vol. 102, pp. 278–291, 2020, doi: 10.1016/j.future.2019.07.040.

[15] T. M. Fernández-caramés and S. Member, "From Pre-Quantum to Post-Quantum IoT Security : A Survey on Quantum-Resistant Cryptosystems for the Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6457–6480, 2020.

[16] W. Susilo, J. Tonien, and G. Yang, "Computer Standards & Interfaces Divide and capture : An improved cryptanalysis of the encryption standard algorithm RSA," *Comput. Stand. Interfaces*, vol. 74, no. July 2020, p. 103470, 2021, doi: 10.1016/j.csi.2020.103470.

[17] H. Khalid and A. Shobole, "Existing Developments in Adaptive Smart Grid Protection : A Review," *Electr. Power Syst. Res.*, vol. 191, no. November 2020, p. 106901, 2021, doi: 10.1016/j.epsr.2020.106901.

[18] M. Bertolini, M. Buso, and L. Greco, "Competition in smart distribution grids ☆," *Energy Policy*, vol. 145, no. July, p. 111729, 2020, doi: 10.1016/j.enpol.2020.111729.

[19] O. Majeed, M. Zulqarnain, and T. Majeed, "Recent advancement in smart grid technology : Future prospects in the electrical power network," *Ain Shams Eng. J.*, vol. in progres, no. July, pp. 1–9, 2020, doi: 10.1016/j.asej.2020.05.004.

[20] A. Munandar, H. Fakhrurroja, M. I. Rizqyawan, R. P. Pratama, J. W. Wibowo, and I. A. F. Anto, "Design of Real-time Weather Monitoring System Based on Mobile Application using Automatic Weather Station," in *2017 2nd International Conference on Automation, Cognitive Science, Optics, Micro-Mechanical Systems, and Information Technology (ICACOMIT)*, 2017, pp. 44–47.

[21] D. Devapal, "Smart Agro Farm Solar Powered Soil and Weather Monitoring System for Farmers," *Mater. Today Proc.*, vol. 24, pp. 1843–1854, 2020, doi: 10.1016/j.matpr.2020.03.609.

[22] K. Hasan, K. Biswas, K. Ahmed, and N. S. Nafi, "A comprehensive review of wireless body area network," *J. Netw. Comput. Appl.*, vol. 143, no. April, pp. 178–198, 2019, doi: 10.1016/j.jnca.2019.06.016.

[23] B. Narwal and A. K. Mohapatra, "A survey on security and authentication in wireless body area networks," *J. Syst. Archit.*, no. August, p. 101883, 2020, doi: 10.1016/j.sysarc.2020.101883.

[24] S. Al-janabi, I. Al-shourbaji, M. Shojafar, and S. Shamshirband, "Survey of main challenges (security and privacy) in wireless body area networks for healthcare applications," *Egypt. Informatics J.*, vol. 18, no. 2, pp. 113–122, 2017, doi: 10.1016/j.eij.2016.11.001.

[25] X. Liu, Z. Wang, Y. Ye, and F. Li, "An efficient and practical certificateless signcryption scheme for wireless body area networks," *Comput. Commun.*, vol. 162, no. August, pp. 169–178, 2020, doi: 10.1016/j.comcom.2020.08.014.

[26] M. M. Rathore, A. Paul, A. Ahmad, N. Chilamkurti, W. Hong, and H. Seo, "Real-time secure communication for Smart City in high-speed Big Data environment," *Futur. Gener. Comput. Syst.*, vol. 83, no. Jun, pp. 638–652, 2018, doi: 10.1016/j.future.2017.08.006.

[27] F. H. Al-naji and R. Zagrouba, "A survey on continuous authentication methods in Internet of Things environment," *Comput. Commun.*, vol. 163, no. Sept, pp. 109–133, 2020, doi: 10.1016/j.comcom.2020.09.006.

[28] N. T. E. Hermawan, E. Winarko, and A. Ashari, "Securing Data Transmission for Radiation Monitoring System in Nuclear Installation," *Int. J. Comput. Appl.*, vol. 179, no. 22, pp. 32–40, 2018.

[29] K. A. P. Kumar, G. A. S. Sundaram, B. K. Sharma, S. Venkatesh, and R. Thiruvengadathan, "Advances in gamma radiation detection systems for emergency radiation monitoring," *Nucl. Eng. Technol.*, vol. 52, no. 10, pp. 2151–2161, 2020, doi: 10.1016/j.net.2020.03.014.

[30] J. H. Kim, K. H. Park, and K. S. Joo, "Development of low-cost , compact , real-time , and wireless radiation monitoring system in underwater environment," *Nucl. Eng. Technol.*, vol. 50, no. 5, pp. 801–805, 2018, doi: 10.1016/j.net.2018.03.023.

[31] M. G. Kamardan, N. Aminudin, N. Che-Him, S. Sufahani, K. Khalid, and R. Roslan, "Modified Multi Prime RSA Cryptosystem," *J. Phys. Conf. Ser.*, vol. 995, no. 1, pp. 1–6, 2018, doi: 10.1088/1742-6596/995/1/012030.

[32] K. El Makkaoui, A. Beni-Hssane, A. Ezzati, and A. El-Ansari, "Fast Cloud-RSA Cloud-RSA Scheme for Promoting Promoting Data Data Confidentiality in the the Cloud Computing," in *Procedia Computer Science*, 2017, vol. 113, pp. 33–40, doi: 10.1016/j.procs.2017.08.282.

[33] S. Nalajala, P. Ch, A. Meghana, and P. M. B, "Data Security Using Multi Prime RSA in Cloud," *Internatinal J. Recent Technol. Eng.*, vol. 7, no. 6S4, pp. 110–115, 2019.

[34] P. Matta, M. Arora, and D. Sharma, "A comparative survey on data encryption Techniques: Big data perspective," *Mater. Today Proc.*, no. xxxx, 2021, doi: 10.1016/j.matpr.2021.02.153.

[35] W. Susilo, J. Tonien, and G. Yang, "A generalised bound for the Wiener attack on RSA," *J. Inf. Secur. Appl.*, vol. 53, p. 102531, 2020, doi: 10.1016/j.jisa.2020.102531.

[36] N. A. A. Sani and H. Kamarulhaili, "RSA cryptography and multi prime RSA cryptography," in *AIP Conference Proceedings*, 2017, vol. 1870, doi: 10.1063/1.4995903.

[37] N. T. E. Hermawan, E. Winarko, and A. Ashari, "Multi prime numbers principle to expand implementation of CRT method on RSA algorithm," in *AIP Conference Proceedings*, 2021, vol. 2331, no. April, pp. 1–10, doi: 10.1063/5.0041856.

[38] C. Intila, B. Gerardo, and R. Medina, "A study of public key 'e' in RSA algorithm," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 482, no. 1, pp. 1–9, 2016, doi: 10.1088/1757-899X/482/1/012016.

[39] J. Sahu, V. Singh, V. Sahu, and A. Chopra, "An Enhanced Version of RSA to Increase the Security," *J. Netw. Commun. Emerg. Technol.*, vol. 7, no. 4, pp. 2395–5317, 2017.

[40] R. M. Pir, "Security improvement and Speed Monitoring of RSA Algorithm," *Int. J. Eng. Dev. Res.*, vol. 4, no. 1, pp. 195–200, 2016.

[41] Manu and A. Goel, "Encryption algorithm using dual modulus," in *3rd IEEE International Conference on Computational Intelligence and Communication Technology (IEEE-CICT 2017)*, 2017, pp. 1–4, doi: 10.1109/CIACT.2017.7977331.

[42] B. Swami, R. Singh, and S. Choudhary, "Dual Modulus RSA based on Jordan-Totient function," *Procedia Technol.*, vol. 24, pp. 1581–1586, 2016, doi: 10.1016/j.protcy.2016.05.143.

[43] R. S. Abdeldaym, M. A. Elkader, Hate, and R. Hussein, "Modified RSA Algorithm Using Two Public Key and Chinese Remainder Theorem," *Int. J. Electron. Eng.*, vol. 10, no. 1, pp. 51–64, 2019, doi: 10.6636/IJEIE.201903.

[44] K. D. M. AlSabti and H. R. Hashim, "A New Approach for Image Encryption in the Modified RSA Cryptosystem Using MATLAB," *Glob. J. Pure Appl. Math.*, vol. 12, no. 4, pp. 3631–3640, 2016.

[45] S. A. Jaju, "A Modified RSA Algorithm to Enhance Security for Digital Signature," *Int. Conf. Work. Comput. Commun.*, pp. 1–5, 2015, doi: 10.1109/IEMCON.2015.7344493.

[46] C. J. L. Padmaja, V. S. Bhagavan, and B. Srinivas, "RSA Encryption using Three Mersenne Primes," *Int. J. Chem. Sci.*, vol. 14, no. 4, pp. 2273–2278, 2016.

[47] M. M. A. Zaid and S. Hassan, "Lightweight RSA Algorithm Using Three Prime Numbers," *Int. J. Eng. Technol.*, vol. 7, pp. 293–295, 2018.

[48] P. Chaudhury *et al.*, "ACAFP : Asymmetric Key based Cryptographic Algorithm using Four Prime Numbers to Secure Message Communication . A Review on RSA Algorithm," in *2017 8th Annual Industrial Automation and Electromechanical Engineering Conference*, 2017, pp. 332–337, doi: 10.1109/IEMECON.2017.8079618.

[49] M. Krishnamoorthy and V. Perumal, "Secure and efficient hand-over authentication in WLAN using elliptic curve RSA," *Comput. Electr. Eng.*, vol. 64, pp. 552–566, 2017, doi: 10.1016/j.compeleceng.2017.06.002.

[50] P. K. Panda and S. Chattopadhyay, "A hybrid security algorithm for RSA cryptosystem," *2017 4th Int. Conf. Adv. Comput. Commun. Syst. ICACCS 2017*, 2017, doi: 10.1109/ICACCS.2017.8014644.

[51] A. Nivetha, P. M. S, and S. K. J, "Modified RSA Encryption Algorithm using Four Keys," *Int. J. Eng. Res. Technol.*, vol. 3, no. 07, pp. 3–7, 2015.

[52] H. Ukwuoma and M. Hammawa, "Optimised Key Generation for RSA Encryption Optimised Key Generation for RSA Encryption," *Innov. Syst. Des. Eng.*, vol. 6, no. November 2015, pp. 1–12, 2017.

[53] A. H. Lone and A. Khalique, "Generalized RSA using 2 k Prime Numbers with Secure Key Generation," *Int. J. Secur. Commun. Networks*, vol. 9, no. September, pp. 4443–4450, 2016, doi: 10.1002/sec.

[54] T. L. Grobler and W. T. Penzhorn, "Fast Decryption Methods for the RSA Cryptosystem," in *7th AFRICON Conference in Africa*, 2004, no. 9.

[55] R. Gu, "Multiscale Shannon entropy and its application in the stock market," *Phys. A Stat. Mech. its Appl.*, vol. 484, pp. 215–224, 2017, doi: 10.1016/j.physa.2017.04.164.

[56] L. Truffet, "Shannon entropy reinterpreted," *Reports Math. Phys.*, vol. 81, no. 3, pp. 303–319, 2018, doi: 10.1016/S0034-4877(18)30050-8.

[57] K. Ahmad, M. Adil, S. Khan, A. Ali, and Y. Chu, "New estimates for generalized Shannon and Zipf-Mandelbrot entropies via convexity results," *Results Phys.*, vol. 18, no. July, p. 103305, 2020, doi: 10.1016/j.rinp.2020.103305.

[58] P. M. Cincotta, C. M. Giordano, R. Alves Silva, and C. Beaugé, "The Shannon entropy: An efficient indicator of dynamical stability," *Phys. D Nonlinear Phenom.*, vol. 417, pp. 1–10, 2021, doi: 10.1016/j.physd.2020.132816.

[59] A. Dujella, "A variant of wiener's attack on RSA," *Computing*, vol. 85, no. 1–2, pp. 77–83, 2018, doi: 10.1007/s00607-009-0037-8.

[60] M. Bunder, A. Nitaj, W. Susilo, and J. Tonien, "A generalized attack on RSA type cryptosystems," *Theor. Comput. Sci.*, vol. 704, pp. 74–81, 2017, doi: 10.1016/j.tcs.2017.09.009.

[61] L. Peng, L. Hu, Y. Lu, J. Xu, and Z. Huang, "Cryptanalysis of Dual RSA," *Des. Codes Cryptogr.*, vol. 83, no. 1, pp. 1–21, 2017, doi: 10.1007/s10623-016-0196-5.

[62] D. Vogel, Y. Onayemi, and V. Murad, "Integer Factorization Algorithms," *Teach. Course - Math Proj.*, pp. 1–20, 2016.

[63] G. Pandey and S. K. Pal, "Polynomial selection in number field sieve for integer factorization," *Perspect. Sci.*, vol. 8, pp. 101–103, 2016, doi: 10.1016/j.pisc.2016.04.007.

[64] L. T. Yang, Y. Huang, J. Feng, Q. Pan, and C. Zhu, "An improved parallel block Lanczos algorithm over GF(2) for integer factorization," *Inf. Sci. (Ny).*, vol. 379, pp. 257–273, 2017, doi: 10.1016/j.ins.2016.09.052.

[65] E. J. Vuicik, D. Šešok, and S. Ramanauskaitė, "Efficiency of RSA Key Factorization by Open-Source Libraries and Distributed System Architecture," *Balt. J. Mod. Comput.*, vol. 5, no. 3, pp. 269–274, 2017, doi: 10.22364/bjmc.2017.5.3.02.

[66] R. Granger, T. Kleinjung, A. K. Lenstra, B. Wesolowski, and J. Zumbr, "Computation of a 30 750-Bit Binary Field Discrete Logarithm," 2020.

[67] F. Boudot, P. Gaudry, A. Guillevic, N. Heninger, E. Thomé, and P. Zimmermann, "New factorization and discrete logarithm record computations," Nancy, France, 2020.

[68] K. Somsuk, "The new integer factorization algorithm based on Fermat's Factorization Algorithm and Euler's theorem," *Int. J. Electr. Comput. Eng.*, vol. 10, no. 2, pp. 1469–1476, 2020, doi: 10.11591/ijece.v10i2.pp1469-1476.

[69] K. Somsuk, "The improvement of initial value closer to the target for Fermat's factorization algorithm," *J. Discret. Math. Sci. Cryptogr.*, vol. 21, no. 7–8, pp. 1573–1580, Nov. 2018, doi: 10.1080/09720529.2018.1502737.

[70] V. Zadiraka, Y. Nykolaychuk, and S. Ivasiev, "The theory of factorization multidigit numbers," *Proc. 13th Int. Conf. Exp. Des. Appl. CAD Syst. Microelectron. CADSM 2015*, pp. 221–225, 2015, doi: 10.1109/CADSM.2015.7230841.