JJEE
Jordan Journal of Electrical Engineering
Volume 7, Number 1, March 2021
Pages 1-14
ISSN (Print): 2409-9600, ISSN (Online): 2409-9619

Modified RSA Using Triple Keys Based Encryption/Decryption

Ghassan Kbar[1]*, Wathiq Mansoor[2]

[1] School of IT and Engineering, Melbourne Institute of Technology, Sydney, Australia
E-mail: gkbar@mit.edu.au
[2] Electrical Engineering Department, College of Engineering & IT, University of Dubai, Dubai, UAE

Received: September 17, 2020 *Revised: November 16, 2020* *Accepted: November 25, 2020*

***Abstract*—**Public key infrastructure is used for strong encryption and decryption in large applications across the internet. If the private key is known to hackers, then all previous encrypted received messages can be decrypted. In this paper, a scheme that uses triple keys for encryption/decryption is proposed to overcome several attacks that are possible on RSA algorithm. This scheme is based on the modified RSA algorithm by supporting three distinct prime numbers used for encryption/decryption. These numbers are used to generate three encryption/decryption keys that are coprime with the gcd (ϕ(p, q, r) = (p-1)(q-1)(r-1)). These keys are the server's private key (p), the secure user's private key (q), and the server's public key (r). Hackers, who manage to get a copy of the server's private key or are able to crack the server's public key, cannot decrypt the messages exchanged between the user and the server as they need to know the user's private key. This key is only known to initiators who have the authentication parameters. Therefore, hackers are unable to crack the message encrypted using the proposed technique. In addition, cryptanalyst would require double the time to break the system compared to RSA as revealed by the obtained results.

***Keywords*—** Modified RSA; Public key infrastructure; Asymmetric encryption; Security; Cryptography.

1. INTRODUCTION

Cryptography is a technique used for securing the communication in the presence of third parties [1]. Modern cryptography would ensure the various security aspects such as data confidentiality, data integrity, authentication, and non-repudiation [2, 3]. Two kinds of cryptosystems can be used which are symmetric and asymmetric. In symmetric systems, the same secret key with minimal length is used to encrypt and decrypt a message [4]. In asymmetric systems, a public key is used to encrypt a message and a private key is used to decrypt it [5].

In the section of literature review, we cover the different techniques used in symmetric key, asymmetric key, enhanced asymmetric key, and combination of symmetric and asymmetric key. Symmetric key technique suffers from two main issues that are the ability to exchange the key securely to parties involved in the communication, and the possibility of cracking the key that can be enhanced using the technique of cipher block chaining (CBC).

In the asymmetric key technique using RSA, the maximum message has to be less than or equal to 4096 bits, it factors the modulus n where this factorization can be broken using a quantum computer in polynomial time, and it suffers from slow processing for decryption which can be enhanced using the Chinese Remainder Theorem (CRT).

Enhanced RSA using different values of the public key 'e' could lead to better security and speed of execution [6]. Another technique called improved RSA algorithm using two public key pairs rather than sending one public key has also been proposed, where a brute force attack will be more difficult to crack the message as the encryption keys are sent

* Corresponding author

Article's DOI: 10.5455/jjee.204-1600376137

separately [7]. However, these proposed techniques using the enhanced and improved RSA didn't cover the complexity issue for cracking the key as will be described in our technique.

A modified RSA with three prime numbers has been proposed in this paper. It overcomes the deficiencies of the RSA proposed by Collins et al. [8], who used multiple prime keys but without demonstrating the complexity of cracking as in our proposed method. In addition, we are using a new mathematical model to prove our method. The prime numbers are used to generate one public key and a secure user's key in addition to a private key. In this approach, a user can access the server using a public key, then he/she can send its sensitive message encrypted by a secure public and user's key. Attackers, in this case, will be unable to decrypt these messages as the user's key used for encryption is not known to them, and the private key used for encryption is only known to server. Another application - based on decrypting messages using two keys - can also be used by the user where he can rely on the public key to communicate to server, and its messages will be decrypted by a private key first, and then by its user's key that can be stored in the authentication server. Attackers in this case are unable to decrypt the messages as they need to know the private key and the user's key which are kept secret by the server and the authentication server. Two symmetric keys are also used by the client and server to exchange their data securely. The following sections cover the mathematical formula for the new algorithm to support three keys, the complexity for deciphering approach used by cryptanalyst, two applications of the modified RSA algorithm using three keys, and implementation prototypes to prove the correctness of the new algorithm. In section 2, literature review has been presented, and in section 3 the proposed algorithm has been presented. In section 4, the complexity for deciphering approach used by cryptanalyst has been determined and proven to be more efficient than RSA. In section 5, the application of the modified RSA algorithm using three keys has been covered, and in section 6 a prototype implementation of the proposed algorithm has been done and proven to work well for encryption and decryption using the three keys.

2. LITERATURE REVIEW

Block and stream ciphering can be used for symmetric key encryption such as advanced encryption standard (AES) and data encryption standard (DES)/Triple DES (3DES) [9]. This requires both parties communicating to each other to agree on the key encryption which will be sent over insecure channel. In symmetric electronic code book (ECB) mode, input blocks will always map to identical output blocks [10]. An active attack can enable an attacker to decrypt ciphertexts encrypted using ECB mode, where it uses the oracle which will encrypt that data, followed by a secret suffix S. A better solution that can be used to address the issue of ECB is the CBC, where plaintext blocks are XORed with the previous ciphertext block before being encrypted by the block cipher. Since there is no previous ciphertext block to XOR it with the first plaintext block, an initializing vector (IV) that should be random and unpredictable can be used. The CBC still suffers from bit filliping attacks, where attackers can add a very long string of Z bytes in their user name to recover the decrypted user name [10]. In addition, CBC can also suffer from CBC padding attacks, where an attacker uses an oracle function to determine the actual value of the padding [11].

In public key infrastructure (PKI), the public/private key has a key pair. One public key which can be freely distributed, and a private one in which the initiator will keep it. It uses a long key and becomes slow. Diffie-Hellman (DH) with discrete logarithms or with elliptic curves addressed the issue of key exchange, where it allows the exchange of common base numbers that are based on the large prime (p) and the base (g), and random numbers that will be generated by each party and kept secret by them [12]. DH still has an issue where both parties need to authenticate each other and the integrity of the message needs to be guaranteed [10]. RSA algorithm, developed by Ronald **R**ivest, Adi**S**hamir, and Len **A**dleman, answered the challenge for creating the first publicly known examples of high quality public-key algorithms, and has been among the most widely used algorithms [13]. A single 2048-bit RSA encryption takes 0.29 megacycles and the decryption takes 11.12 megacycles [14]. AES with Galois/Counter Mode (AES-GCM) with hardware acceleration or Salsa20/ChaCha20 only need about 2 to 4 cycles per byte, making the difference even larger [10]. However, RSA cannot encrypt anything larger than its modulus, which is less than or equal to 4096 bits. This is far smaller than the largest messages that people like to send. RSA's problem is to factor the modulus n, but the factorization using a quantum computer can be performed in polynomial time. Furthermore, a theoretical hardware device challenges the security of 1024 bits [15]. RSA suffers from common modulus attacks, where users may avoid generating a different modulus n for each user in a group. Once n is factored, they can recover other private key d from the public key e [16]. Low private exponent (d) [5], and low public exponent (e) [17] would make attacks simpler, since the factorization of n can be found in polynomial time. RSA asymmetric based keys are much slower than DES and other symmetric cryptosystems. Usually RSA is used to transmit the symmetric key to other parties. To enhance the speed of RSA cryptosystem, different research approaches have been proposed. To increase the speed of decrypting the message using RSA algorithm with the help of CRT, two smaller secret keys (dp, dq) generated from the original secret key (d) have been used [18]. The speedup over standard RSA (with CRT) is approximately SCRT= (2. (n/2)3)/ (2. (n/k)3)= k3/8. Rebalanced RSA is a proposed algorithm for fast signature generation/decryption time, where it generates two distinct random (n/2) bit prime numbers for s ≤ n/2 bits [5]. The private prime number d is chosen as dp mod p-1, and dq mod q-1 which are small numbers, where e=d-1 mod (n) and the speedup over standard RSA with CRT is SCRT=n/2s. Authors of [19] proposed another algorithm that combines the algorithm Rebalanced RSA [5] with MultiPower RSA [8]. The theoretical speedup of their proposed scheme as with standard RSA and with CRT is SCRT=nk2/8 s. The improvement of decryption time is done at the expense of increasing the encryption/signature verification time.

A new concept in RSA cryptosystem to enhance the RSA algorithm has been proposed by adding a third prime number in the composition of the public and private key with reduced size [20]. In this method, they generated a large variable for n, and the process of analysis of the factors O(n3) was more complex than the original algorithm O(n2). A scheme that has speed improvement on the RSA's decryption side by using the CRT has been proposed [21]. In this proposed scheme, two chipper text values (C1, C2) have to be created, and the decryption will go through a complex process of 6 steps. In addition, if the server would encrypt messages and send it to client for authentication, it needs to execute the 6 steps for encrypting the message which would take a long time. Four prime numbers have been used in the modified RSA algorithm, and instead of sending the public key directly, two key pairs of public keys are sent to the receiver [22]. This scheme has a speed enhancement on the

RSA's decryption side by using CRT. In this approach, two chipper texts have to be generated and 6 complex steps need to be used for the decryption and it would be complex for digital signature verification. Different values of the public key 'e' are generated by using a modified RSA and could lead to better security and speed of execution by creating a new public key f = e*2 +1 [6]. These extra steps for generating a new public key are not needed as 'e' can be calculated easily from f; hence, the security level would be the same. An improved RSA algorithm by using two public key pairs rather than sending one public key has also been proposed by Jahan et al. [7]. They created two public keys y and x, where e=y/x. In this case, y is a multiple of x and e is the public key in the normal RSA algorithm. The paper claimed that a brute force attack will be more difficult as the encryption keys are sent separately. Since e=y/x must be calculated by the sender, the effect of a brute force attack would be the same as sending only e because a hacker can easily calculate e from y and x.

Public-key cryptosystems are used in conjunction with secret-key cryptosystems [10] to address the issues that we were facing in exchanging the secret keys with a large number of people. Message authentication codes (MAC) are used in symmetric authentication schemes, while a digital signature is used for public key authentication [10]. The hash functions can be used to produce both signature schemes as well as message authentication schemes [10]. A hash functions can take an input of indeterminate length and produce a fixed-length value, also known as a "digest". Popular asymmetric key encryption algorithms include EIGamal, RSA, DSA and Elliptic curve techniques. One typical technique for discovering the public key is using digital certificates in a client-server model of communication. However, asymmetric encryption takes relatively more time than the symmetric encryption in Secure Sockets Layer (SSL) protocol. In addition, one of the drawbacks to public key encryption systems is that they need relatively complicated mathematics to work, making them very computationally intensive. When someone gets their hands on a symmetric key, they can decrypt everything encrypted with that key [23]. If hackers can use the symmetric key or asymmetric private key to decrypt all messages sent previously, breaking the private key into user's private key and server's private key would make it very hard for them to know both private keys and therefore would be unable to decrypt the previous messages.

A complete cryptosystem such as the one used in Secure Socket Layer and Transport Layer Security SSL/TLS uses both symmetric and asymmetric algorithms in the application. Signature algorithms can be used to authenticate peers and public key algorithms can be used to negotiate shared secrets and authenticated certificates. It has one major flaw, when an attacker gets access to the server's private key, he/she can decrypt all past communication. TLS allows peers to agree on the pre-master secret using a Diffie-Hellman exchange, either based on discrete logs or elliptic curves [10]. The service usually authenticates the user using passwords, and occasionally by using a two-factor authentication. However, there are no systems that are easy to use for a technical people who rely on client certificates. CRIME is an attack by the authors of BEAST using an innovative side channel attack that relies on the TLS compression leaking information about secrets in the plaintext [10]. Patel and Panchal proposed a hybrid approach by combining the two most important algorithms, RSA algorithm and Diffie Hellman algorithm [24]. The security of the message by combining encryption and bitwise x-or operation can be improved but the complexity of the message is also increased. Authors claim that their new proposed model provides more security compared to a normal RSA algorithm. A content based double encryption algorithm has been proposed using symmetric key cryptography, where the authors implemented a binary addition operation, a circular bit shifting operation, a folding method and a symmetric key cryptography [25]. In order for this to work, both parties need to know about the algorithm, which is a challenge for this proposal.

3. MODIFIED RSA ALGORITHM TO SUPPORT THREE KEYS

The proposed modified algorithm is based on three distinct primes (p, q, and r) that are used to generate two encryption numbers (e, f) which are coprime with $\lambda = (p-1)(q-1)(r-1)$, and one decryption number (d) that is coprime with λ. The following steps 1 to 7 describe the algorithm and other steps prove the proposed algorithm.

Algorithm:

Given positive integers n, e, f, and d such that:

1. $n = pqr$, where p q and r, are distinct primes (1)
2. $\gcd(e, \phi(n)) = 1$. $\phi(n)= \lambda$ (2)
3. $d.e.f \equiv 1 \pmod{\phi(n)}$, e and f are coprime of λ (3)
4. Define the public and private key algorithms of a message m, for $0 \le m < n$,
5. mRSAPublic(m) = me mod n, (4)
6. mRSAPrivate(m) = mfd mod n (5)
7. mRSA: is modified RSA based on three prime numbers and three encryption numbers.

Proof of the algorithm:

We need to prove that message m can be encrypted by Eq. (6) and decrypted by Eq. (7).

m = mRSAPrivate (mRSAPublic(m)) (6)

m = mRSAPublic (mRSAPrivate(m)) (7)

Prove that Eq. (6) and Eq. (7) can be used inversely to obtain the message m, or "Does mRSA encryption actually work?"

By substituting Eqs. (4) and (5) into Eqs. (6) and (7), respectively, we can say that:

mRSAPrivate (mRSAPublic(m)) = (me mod n)fd mod n = mfde mod n.

We can also say that:

RSAPublic (RSAPrivate(m)) = (mfd mod n)e mod n = mfde mod n.

Therefore, Eq. (6) and Eq. (7) are equivalent, or:

RSAPrivate (RSAPublic(m)) = RSAPublic (RSAPrivate(m)).

If we can prove:

m = mfde mod n, then, the proof will be complete.

It is given (in Eq. (3)) that

$dfe \equiv 1 \pmod{\phi(n)}$.

By definition of mods, we can write Eq. (3) as:

$$\phi(n) \mid dfe - 1. \quad (8)$$

Since $\phi(n) = \phi(p)\phi(q)\phi(r)$ only when p, q and r are relatively prime, as in this case, we have:

$\phi(n) = \phi(p)\phi(q)\phi(r)$.

By substitution into (8) we have:

$\phi(p)\phi(q)\phi(r) \mid dfe - 1$.

By properties of divisors, we can write:

$\phi(p) \mid dfe - 1$,
$\phi(q) \mid dfe - 1$, and
$\phi(r) \mid dfe – 1$.

where there must be an integer k such that:

$dfe - 1 = k\phi(p)$.

Since p is prime, the Euler phi function states that $\phi(p) = p - 1$, so

$dfe - 1 = k(p - 1)$. (9)

By the symmetric property of mods, we can write:

$mdfe \equiv mdfe \pmod{p}$
$\equiv mdfe - 1 + 1 \pmod{p}$.

This can also be written as:

$mdfe \equiv (mdfe - 1) * m \pmod{p}$. (10)

Substituting Eq. (9) into Eq. (10), we obtain:

$mdfe \equiv (mk(p - 1)) * m \pmod{p}$. (11)

Since p is prime, any integer m for Eq. (11) will be either: relatively prime to p or a multiple of p.

When m is relatively prime to p, Fermat's Little Theorem states that:

$mp - 1 \equiv 1 \pmod{p}$.

By properties of mods, we can write:

$mk(p - 1) \equiv 1k \pmod{p}$, or
$mk(p - 1) \equiv 1 \pmod{p}$. (12)

By combining Eqs. (11) and (12), we obtain:

$mdfe \equiv 1 * m \pmod{p}$, or
$mdfe \equiv m \pmod{p}$. (13)

In the second case, where m is a multiple of p, if

$p\, l \mid m$, then for any integer k
$p \mid mk$.

From the properties of mods we can write:

$mdfe \equiv 0 \pmod{p}$, and
$m \equiv 0 \pmod{p}$.

Thus, we can write:

$mdfe \equiv m \pmod{p}$.

Therefore, for all m,

$mdfe \equiv m \pmod{p}$, and applying the same process for q and r we can write
$mdfe \equiv m \pmod{q}$ and
$mdfe \equiv m \pmod{r}$.

By the modular property of congruence which states that when m and n are relatively prime (as in our given statements), $a \equiv b \pmod{m}$, and $a \equiv b \pmod{n}$, then $a \equiv b \pmod{mn}$ (proof-of-the-rsa-algorithm). We can also state if mn and l are relatively prime then, $a \equiv b \pmod{mn}$, and $a \equiv b \pmod{l}$, then $a \equiv b \pmod{mnl}$, we can then write the following:

$mdfe \equiv m \pmod{pqr} \equiv m \pmod{n}$. (14)

By the modular property of symmetry, we can write

$m \equiv mdfe \pmod{n}$. (15)

Since we have limited m to $0 \leq m < n$, only one integer will satisfy Eq. (15), and so, m = mdfe mod n. (16)

If we substitute Eq. (16) with our original equations;

mRSAPrivate (mRSAPublic(m)) = mdfe mod n, and

mRSAPublic (mRSAPrivate(m)) = mdfe mod n

We obtain, for $0 \le m < n$,

mRSAPrivate (mRSAPublic(m)) = m, and

mRSAPublic (mRSAPrivate(m)) = m.

4. COMPLEXITY FOR DECIPHERING APPROACH USED BY CRYPTANALYST

The following subsections describe the complexity of the new proposed method that can be used to determine the hardness level for the cryptanalyst to break the key.

4.1. Computing ϕ(n) Without Factoring

If a cryptanalyst could compute two ϕ(n), then he/she could break the system by computing d as the multiplicative inverse of e modulo ϕ(n) and f modulo ϕ(n). Compared to RSA, cryptanalyst needs to compute ϕ(n) to break the system, but in the proposed scheme he/she needs 2 ϕ(n) (one inverse of e modulo ϕ(n), and another one inverse of f modulo ϕ(n)). Therefore, this scheme has double the time for cryptanalyst to break the system, and the proof of this claim is described below in this section.

We argue that this approach is harder than factoring n since it enables the cryptanalyst to easily factor n using two ϕ(n). This approach to factor n has not turned out to be practical.

How can n be factored using ϕ(n)?

First, (p + q + r) is obtained from n and ϕ(n) =n -(p+q+r)+1.

Then,

[(p – q) + (p - r)]2 (17)

should equal to:

$(p + q)^2 + (p + r)^2 - 4n.(q+r)/r.q + 2(p – q)(p - r)$ (18)

Finally, q is half the difference of (p + q) and (p - q), and r is half the difference of (p + r) and (p - r). Therefore, breaking our system would require computing q and r which requires factor of two n for computing ϕ(n). This is harder than breaking other system by only factoring one n, or by breaking m-RSA system using p, q and r only for computing ϕ(n) without the need for factoring n. This is why n must be composite; ϕ(n) is trivial to compute if n is prime.

To prove the formula (p – q) +(p - r) = square root of ((p + q)2 + (p + r)2 - 4n.(p+r)/r.p + 2(p – q)(p - r)), substitute both Eqs. (17) and (18) in their equivalent term:

[(p – q) + (p - r)]2 = (p + q)2 + (p + r)2 - 4n.(q+r)/r.q - 2(p – q)(p - r)

(p - q)2+ (p - r)2 + 2(p – q)(p - r) = (p + q)2 + (p + r)2 - 4n.(q+r)/r.q + 2(p – q)(p - r)

p2+q2-2pq + p2+r2-2pr = p2+q2+2pq + p2+r2+2pr - 4n.(q+r)/r.q

4n.(q+r)/r.q = 4pq + 4pr = 4p(q+r)

r.q (4n.(q+r))/r.p = 4p (q+r) r.q = 4pqr = 4n

4n = 4n (19)

4.2. Determining d Without Factoring n or Computing ϕ(n)

Coprime d should be chosen from a large enough set so that a direct search for it is unfeasible. n can be factored using any multiple of ϕ(n) [26]. Therefore, if n is large, a cryptanalyst should not be able to determine d any easier than he/she can factor n.

A cryptanalyst may hope to find a d0, which is equivalent to the d secretly held by a user of the public key cryptosystem. If such values of d0 were common, then a brute force search could break the system. However, all such d0 differ by the least common multiple of (p-1) (q -1) and (r -1), and finding one enables n to be factored. In Mϕ (n) ≡ 1 (mod n) and e.d ≡1 (mod ϕ(n) can be replaced by lcm (p - 1; q – 1; r -1). Finding any such d0 is, therefore, as difficult as factoring n keywords.

5. APPLICATION OF THE MODIFIED RSA ALGORITHM USING THREE KEYS

To improve the security of RSA that is related to easily breaking the key by cryptanalyst and use the key to decrypt all previous sent messages, three keys are proposed that are based on ($n=p \times q \times r$) that is the multiplication of three prime numbers p, q and r. This is in addition to e, f and d that are coprime of ϕ(n) or λ. The new algorithm was presented in the previous section. Two different applications can benefit from this idea. First, a user/client can use a user's private key as a replacement to the public key known by everyone for encryption/decryption. This will enhance the security as messages will be decrypted by user's private key that is known to the user only instead of public key that is known to everyone. Another application can also be supported by encrypting the message using a user's public key, and decrypt it by the server using two keys that are the server's private key and the user's private key. In this second example, if hackers managed to crack the server's private key, they are unable to crack previous encrypted messages, as they require the user's private key, which is kept secret by the user.

In the first application, a user/client can register with an authentication server and request to obtain a user private key as a replacement to the public key known by everyone. Then, the user can communicate with the server by obtaining its private key from the authentication server in order to encrypt its messages and send them to the server. Whereas, the server would use the server's private key associated with the user's private key to decrypt the messages as shown in steps 1-15 in Fig. 1. At step 1.a and 1.b, the client would request to get the server's public key from certificate authority server. At step 2, the client sends a registration request to the server encrypted by the server's public key. This request is comprised of the user name/password and a temporary key (TK) that is used for encrypting the reply message by the server, and a label request key. At step 3, the server decrypts the request, and at step 4, the server stores user's credential for a specific period and sends confirmation to client at step 5. At step 6, the client sends a connection request to obtain the secured user's keys from the server. At step 7, the server verifies the user's credentials. If it passes, it generates triple key's parameters and stores them for following encryption/decryption sessions with the client as shown in step 8 of Fig. 1. At step 9, the server sends the user's public key parameters (e, n) encrypted by the client's temporary key to the client. At step 10, the client sends a request to connect to the server by supplying its user's credentials, temporary key and the secure connection label (SC), where this request is encrypted by the server's public key. At step 11, the server decrypts the request, verifies the user's credentials. If it passes, it retrieves the encryption parameters associated with the user and client as shown in step 12. At step 13, the server sends the user's key parameters (f, n) to the client encrypted by the client's temporary key. At step 14, the client generates the secured user's public key pu from (e, f, n), and also generates its symmetric key cssk(i) at step 15. At step 16, the client sends its symmetric key encrypted by its secure user's public key (pu) to the server. It uses the formula Pu = $(\text{cssk(i)})^{\text{fi.ei}}$ mod ni, and the server will decrypt the message by the private key Pru using (di, ni) as shown in step 17. Then, the server responds by sending its session key sski to the client encrypted by the client's symmetric key csski as shown in step 19 of Fig. 1. Then, at steps 20 and 21, the client sends its message to the server encrypted by the server's symmetric key sski. At steps 22 and 23, the server replies to the client by encrypting the message by the client's symmetric key csski. For strong security, the client and server use different symmetric keys.

In the second application the message can be encrypted using a user's public key, and decrypt it by the server using two keys that are the server's private key and the user's private key. These keys' parameters can be obtained from the authentication and database server (B DB Auth server) as shown in steps 1-17 of Fig. 2. Similar to previous application of Fig. 1, two session keys related to the client and server are used for encrypting the message as shown in steps 18 to 21 of Fig. 2.

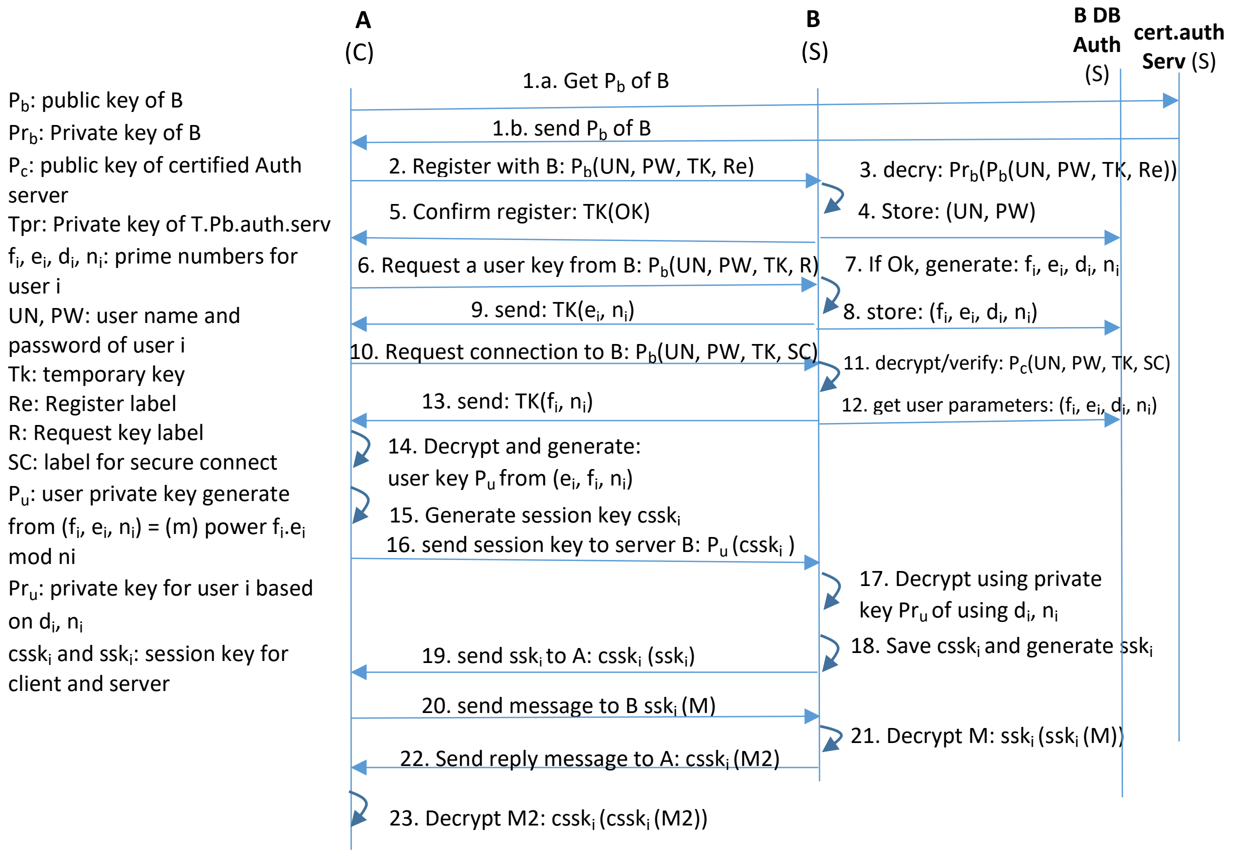
A
(C)
B
(S)
B DB Auth (S)
cert.auth Serv (S)
Pb: public key of B
Prb: Private key of B
Pc: public key of certified Auth server
Tpr: Private key of T.Pb.auth.serv
fi, ei, di, ni: prime numbers for user i
UN, PW: user name and password of user i
Tk: temporary key
Re: Register label
R: Request key label
SC: label for secure connect
Pu: user private key generate from (fi, ei, ni) = (m) power fi.ei mod ni
Pru: private key for user i based on di, ni
csski and sski: session key for client and server
1.a. Get Pb of B
1.b. send Pb of B
2. Register with B: Pb(UN, PW, TK, Re)
3. decry: Prb(Pb(UN, PW, TK, Re))
4. Store: (UN, PW)
5. Confirm register: TK(OK)
6. Request a user key from B: Pb(UN, PW, TK, R)
7. If Ok, generate: fi, ei, di, ni
8. store: (fi, ei, di, ni)
9. send: TK(ei, ni)
10. Request connection to B: Pb(UN, PW, TK, SC)
11. decrypt/verify: Pc(UN, PW, TK, SC)
12. get user parameters: (fi, ei, di, ni)
13. send: TK(fi, ni)
14. Decrypt and generate: user key Pu from (ei, fi, ni)
15. Generate session key csski
16. send session key to server B: Pu (csski)
17. Decrypt using private key Pru of using di, ni
18. Save csski and generate sski
19. send sski to A: csski (sski)
20. send message to B sski (M)
21. Decrypt M: sski (sski (M))
22. Send reply message to A: csski (M2)
23. Decrypt M2: csski (csski (M2))

Fig. 1. First application of the modified RSA algorithm using three keys: client uses two keys' parameters for encryption, and the server uses one private key's parameter for decryption.

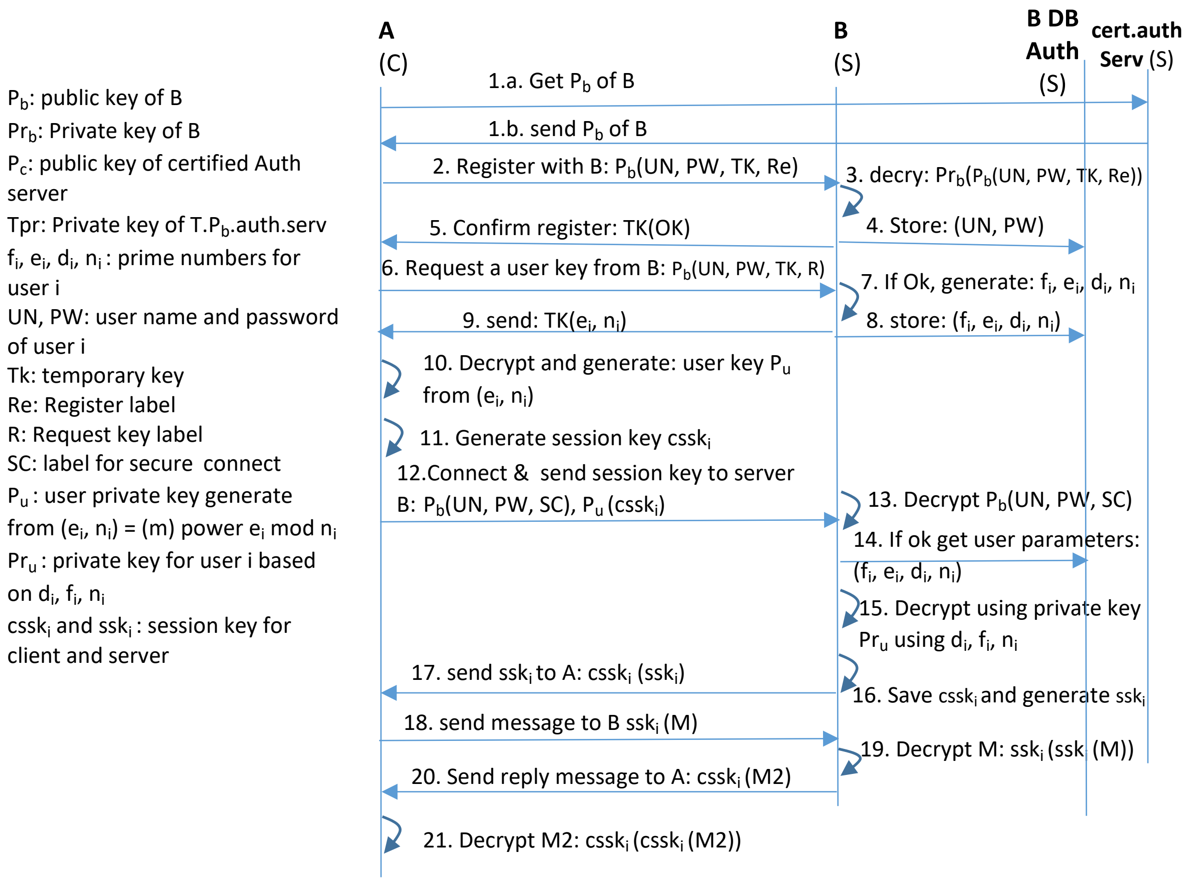
Pb: public key of B
Prb: Private key of B
Pc: public key of certified Auth server
Tpr: Private key of T.Pb.auth.serv
fi, ei, di, ni : prime numbers for user i
UN, PW: user name and password of user i
Tk: temporary key
Re: Register label
R: Request key label
SC: label for secure connect
Pu : user private key generate from (ei, ni) = (m) power ei mod ni
Pru : private key for user i based on di, fi, ni
csski and sski : session key for client and server
A (C)
B (S)
B DB Auth (S)
cert.auth Serv (S)
1.a. Get Pb of B
1.b. send Pb of B
2. Register with B: Pb(UN, PW, TK, Re)
3. decry: Prb(Pb(UN, PW, TK, Re))
4. Store: (UN, PW)
5. Confirm register: TK(OK)
6. Request a user key from B: Pb(UN, PW, TK, R)
7. If Ok, generate: fi, ei, di, ni
8. store: (fi, ei, di, ni)
9. send: TK(ei, ni)
10. Decrypt and generate: user key Pu from (ei, ni)
11. Generate session key csski
12.Connect & send session key to server B: Pb(UN, PW, SC), Pu (csski)
13. Decrypt Pb(UN, PW, SC)
14. If ok get user parameters: (fi, ei, di, ni)
15. Decrypt using private key Pru using di, fi, ni
16. Save csski and generate sski
17. send sski to A: csski (sski)
18. send message to B sski (M)
19. Decrypt M: sski (sski (M))
20. Send reply message to A: csski (M2)
21. Decrypt M2: csski (csski (M2))

Fig. 2. Second application of the modified RSA algorithm using three keys: the client uses one key's parameter for encryption, and the server uses two private keys' parameters for decryption.

In this section, we presented two different applications for enhancing the system security. In the first application, user private key has been used along with public key to encrypt messages and can be used for authenticating the user securely. In the second application, user private key is used for decryption along with server private key which prevents hackers who are bale to break the private key in decrypting previous encrypted messages as they still need to know the user private keys which are very hard to break.

6. PROTOTYPE IMPLEMENTATION

The following examples illustrate the implementation of the proposed algorithms by creating two sets of prime numbers as shown in example-1 and example-2 subsections. We referred to www.broeserling.com to choose the prime numbers (p, q, r) [27]; www.mathsisfun.com was used to choose (e, f) to be coprime with λ [28]; planetcalc.com was used to calculate d to be coprime with λ [29]; and www.cs.drexel.edu was used to encrypt and decrypt the messages [30]

6.1. Example-1

Choose prime numbers from the list given [28]

p = 137, q= 149, r= 211

n=p.q.r = 4307143 [based on Eq. (1)]
λ = (p-1)(q-1)(r-1) = 528360 [based on Eq.(2)]

Choose e to be coprime with λ, which has been calculated on [28] as e= 347. Choose f to be coprime with λ using the same site [28] as f= 317

Calculate e.f= 109999, and d, where gcd (e.f, λ) = 1. d is calculated using [29], then d = 309079, and d.f = 97978043.

6.1.1. Example1-1

Encryption by public key, and decryption by server's private key and user's private key.

Encryption: using [30] to calculate the encrypted message.

message m = "hello world"

encrypted message C = me mod n = 1449017 2865180 1826387 3099645 2547429 2376576 (in numeric form), based on Eq. (15).

Decryption of C using df:

Cdf mod n = 104101 108108 111032 119111 114108 100 (in numeric format) = "hello world" in text format, based on Eq. (16).

6.1.2. Example1-2

Encryption by public key, and decryption by server's private key and user's private key.

Encryption: Message m= "hello world. I am living in Sydney". C= me mod n = 1449017 2865180 1826387 3099645 2547429 370144 1756576 3176912 3803626 432229 1275213 3585206 3990470 962194 1408100 2768807 1867409 (in numeric format)

Decryption of C using df:

Cdf mod n = 104101 108108 111032 119111 114108 100046 32073 32097 109032 108105 118105 110103 32105 110032 83121 100110 101121 (in numeric format) = "hello world. I am living in Sydney" in text format.

6.2. Example-2

Choose prime numbers from list given at [28]

p = 347, q= 149, r= 197
n=p.q.r = 10244197
λ = (p-1)(q-1)(r-1) = 630924

Choose e to be coprime to λ using [28] as e= 563. Choose f to be coprime to λ using [28] as f= 601.

Calculate e.f= 338363, and d, where gcd (e.f, λ) = 337199. Calculate d using [29], then d = 337199, and d.f = 202656599.

6.2.1. Example2-1

Encryption: using [30]

Message m = "hello world. I am living in Sydney" encrypted message C = me mod n = 450001 10133403 1117636 4579235 7300900 1522428 5736764 8444222 6857431 6568682 5536880 5187087 242882 2017025 4366488 7986837 4041348 (in numeric form)

Decryption of C using df:

Cdf mod n = 104101 108108 111032 119111 114108 100046 32073 32097 109032 108105 118105 110103 32105 110032 83121 100110 101121 (in numeric format) = "hello world. I am living in Sydney" in text format.

6.2.2. Example 2-2

Encryption by private key and user's private key, and decryption by server's public key.

Encryption by server: server send encrypted message to client: m= "pass Auth". C = mdf mod n, using df= 202656599, n= 10244197, c= 9114145 3567592 3901192 8752026 5760825.

Decryption by client: client decrypt message C received from server using public key e= 563 and same n: Ce mod n = 65117 116104 32112 97115 115 (in Binary format) and = "pass Auth" in text format

6.2.3. Example 2-3

Encryption by public key and decryption by server's private key and user's key.

Encryption by public key and decryption by common private key (e= 563, d= 129995, L= 630924, n =p.q.r = 10244197).

M= "Hello world"

Encryption: C = me mod n = 3532603 10133403 1117636 4579235 7300900 4859852 (in numeric format).

Decryption: Cdf mod n = m = 72101 108108 111032 119111 114108 100 (in numeric format) = Hello world.

7. CONCLUSIONS

A triple keys based encryption/decryption has been proposed to improve the information security against attacks. A new algorithm based on modified RSA has been proposed and verified by generating three encryption/decryption keys that are based on three prime numbers. This new method for implementing a public-key along with a user's private key and a server's private key would enhance the performance of cryptosystem whose security rests on the difficulties of factoring two large numbers. In addition, attackers who tried to obtain the private key would be unable to succeed in their attacks as they need to know the user's private key that is mapped to user's credentials and this is hard to obtain. The users' or clients' terminal can use the normal PKI to get the public key of the server, and then request its own private user's key parameters to establish the secure connection. Clients can also get one secure public key for encryption while the server can use the server's private key and the user's private key for decryption. One prototype has been implemented to prove the concept of encrypting a message by two keys and decrypting it by a private key. Another prototype has also been implemented to encrypt a message by one key and decrypt it by two keys. As shown in section 4 a cryptanalyst would require double the time compared to RSA to break the system. Further work needs to be done in the future to assess the impact of using triple keys on the processing speed.

Acknowledgement: The authors would like to thank Ms. Gina Karlin the English instructor at university of Dubai for her support in proofreading the paper.

REFERENCES

[1] R. Rivest, *Algorithms and Complexity: Cryptography*, Elsevier, 1990.

[2] A. Menezes, P. Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, Boca Raton: CRC Press, 1997.

[3] F. Brecht, B. Fabian, S. Kunz, S. Mueller, "Are willing to wait longer for internet privacy," *Proceedings of European Conference on Information Systems*, pp. 1-12, 2011.

[4] M. Blaze, W. Diffie, R. Rivest, B. Schneier, T. Shimomura, *Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security*, A Report by an Ad Hoc Group of Cryptographers and Computer Scientists, Computer Science, 1996.

[5] M. Wiener, "Cryptanalysis of RSA with short secret exponents," *IEEE Transactions on Information Theory*, vol. IT-36, pp. 553-558, 1990.

[6] C. Intila, B. Gerardo, R. Medina, "A study of public key 'e' in RSA algorithm," *The International Conference on Information Technology and Digital Applications*, pp. 1-9, 2019.

[7] I. Jahan, M. Asif, L. Rozario, "Improved RSA cryptosystem based on the study of number theory and public key cryptosystems," *American Journal of Engineering Research*, vol. 4, no. 1, pp. 143-149, 2015.

[8] T. Collins, D. Hopkins, S. Langford, M. Sabin, *Public Key Cryptographic Apparatus and Method*, US Patent #5, 848, 159, 1998.

[9] G. Kessler, "An overview of cryptography," 2020. <https://www.garykessler.net/library/crypto.html>

[10] L. Van Houtven, *Crypto101 Book*, LaurensVanHoutven, 2013-2014.

[11] R. Housley, "Cryptographic message syntax," *RFC 5652*, 2019. <https://tools.ietf.org/html/rfc5652#section-6.3>

[12] W. Diffie, M. Hellman, "Multi-user cryptographic techniques," *Proceedings of the National Computer Conference and Exposition*, pp. 109–112, 1976.

[13] R. Rivest, A. Shamir, L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[14] W. Stephen, *A New Kind of Science*, Wolfram Media, 2002.

[15] A. Shamir, E. Tromer, "Factoring large numbers with the TWIRL device," *Annual International Cryptology Conference*, Berlin, pp. 1-26, 2003.

[16] D. Boneh, "Twenty years of attacks on the RSA cryptosystem," *Notices of the AMS*, vol. 46, no. 2, pp. 203-213, 1999.

[17] D. Coppersmith, "Small solutions to polynomial equations and low exponent RSA vulnerabilities," *Journal of Cryptology*, vol. 10, no. 4, pp. 233-260, 1997.

[18] J. Quisquater, C. Couvreur, "Fast decipherment algorithm for RSA public-key cryptosystem," *Eletronic Letters*, vol. 18, pp. 905-907, 1982.

[19] D. Garg, S. Verma, "Improvement over public key cryptographic algorithm," *IEEE International Advance Computing Conference*, pp. 734-739, 2009.

[20] A. Al-Hamami, I. Aldariseh, "Enhanced method for RSA cryptosystem algorithm," *IEEE International Conference on Advanced Computer Science Applications and Technologies*, pp. 402-408, 2012.

[21] N. Somani, D. Mangal, "An improved RSA cryptographic system," *International Journal of Computer Applications,* vol. 105, no. 16, pp. 18-22, 2014.

[22] A. Rai, S. Jain, "Modified RSA cryptographic system with two public keys and chinese remainder theorem," *International Journal of Computer Science and Engineering*, vol. 4, no. 7, pp. 22-25, 2017.

[23] G. Jackson, "Advantages and disadvantages of symmetric key encryption," 2019. <https://itstillworks.com/disadvantages-publickey-encryption-1980.html>

[24] G. Patel, K. Panchal, "Hybrid encryption algorithm," *International Journal of Engineering Development and Research*, vol. 2, no. 2, pp. 2064-2070, 2014.

[25] S. Chandra, B. Mandalb, S. Alam, S. Bhattacharyya, "Content based double encryption algorithm using symmetric key cryptography," *Procedia Computer Science*, vol. 57, pp. 1228–1234, 2015.

[26] G. Miller, "Riemann's hypothesis and tests for primality," *Proceedings of Seventh Annual ACM Symptoms on the Theory of Comptng*, New Mexico, USA, pp. 234-239, 1975.

[27] Broeserling, "Generate prime numbers," <https://www.browserling.com/tools/prime-numbers>

[28] Mathsisfun, "Coprime calculator," <https://www.mathsisfun.com/numbers/coprime-calculator.html>

[29] planetcalc, "Modular multiplicative inverse," 2013. <https://planetcalc.com/3311/>

[30] J. Popyack, "RSA express encryption/decryption calculator," <https://www. cs.drexel.edu /~jpopyack/Courses/CSP/Fa17/notes/10.1_Cryptography/RSA_Express_EncryptDecrypt_v2. html>