Vol. 41 No. 8 IPSJ Journal Aug. 2000

Regular Paper

Generating Shared RSA Parameters for Two Communicating Parties

Ari Moesriami Barmawi,[†] Shingo Takada[†] and Norihisa Doi[†]

The RSA encryption system is a widely used cryptographic protocol, requiring the generation of several parameters. Boneh and Franklin proposed a protocol to efficiently generate shared RSA parameters, but it needs a third party. Cocks, Poupard-Stern and Gilboa proposed improvements of Boneh-Franklin's protocol that do not need the help of a third party, but their protocols have a large computational complexity. We propose a protocol for generating shared RSA parameters for two communicating parties. Our protocol does not need the help of a third party and has less computational complexity than the protocols proposed by Cocks, Poupard-Stern and Gilboa. We assume that before both parties execute the protocol, they agree on the size of the modulus number which will be generated and the hash function that will be used. Our protocol generates a public modulus number without the parties knowing the factors of that number. Although the encryption key is publicly known, each party holds only a part of the key that is used to decrypt the received messages.

1. Introduction

The RSA encryption system is a widely used cryptographic protocol requiring the generation of three parameters: (1) a public modulus number N which is a product of two prime numbers p and q; (2) a public encryption key e; and (3) a secret decryption key d. The two prime numbers p and q are kept secret between the two communicating parties, and the three parameters have the following relation: ($de \equiv 1 \bmod \phi(N)$), where $\phi(N)$ is the Euler Totient Function of N. Using the public parameters, one party can encrypt any message M (which is represented as a positive integer less than N) as ($E = M^e \bmod N$), and the other party can decrypt this encrypted message E using his secret key d, using the formula ($D = E^d \bmod N$). This results in the decrypted message D which is the same as the original message M.

There are several cryptographic protocols that require an RSA modulus number for which none of the parties know the factorization, such as Feige-Fiat-Shamir[3)], Fiat-Shamir[4)], Ohta-Okamoto[5)] and Ong-Schnorr[6)]. Thus it becomes necessary for the communicating parties to jointly generate the modulus number N, while keeping the factors of N secret from the communicating parties.

Boneh-Franklin[1)], Cocks[2)], Poupard-Stern[12)] and Gilboa[15)] have proposed methods for generating these shared parameters. However, Boneh-Franklin's method needs the help of a third party. Cocks', Poupard-Stern's, and Gilboa's methods do not need the help of a third party, but their computational complexity is quite large and Cocks' protocol needs to produce several random numbers to secure each party's number.

We propose a protocol for generating shared parameters that does not require the help of a third party and also does not require producing several random numbers to secure each party's number. This protocol consists of procedures for generating the modulus number and the decryption keys, and also a procedure for recovering the encrypted message. The parameters are generated based on a Basic Protocol. Furthermore, although the encryption key e is publicly known, the decryption key d itself is shared among the communicating parties in a way which enables threshold decryption. This means that the decryption key d is split into two parts d_A and d_B, and each party holds one or the other. Thus, no party is able to recover the original message M from the encrypted message E by itself.

This paper first describes the Basic Protocol. Section 3 describes our proposed protocol. Section 4 then analyzes our protocol in terms of security. Section 5 discusses the primality test and compares our approach with previous proposed protocols (such as those of Boneh-Franklin[1)], Cocks[2)], Poupard-Stern[12)] and Gilboa[15)]). Section 6 makes concluding remarks.

[†] Department of Computer Science, Graduate School of Science and Technology, Keio University

2. The Basic Protocol

In this section, we describe first Gilboa's method [15] followed with the details of the Basic Protocol which will be used in the procedures for generating the shared RSA modulus number and the shared decryption keys.

2.1 Gilboa's Method

We first describe Gilboa's method, since this method is used in the Basic Protocol.

Gilboa's method is a method for converting additive shares and multiplicative shares into multiplicative shares and additive shares respectively. During the conversion, no parties can obtain information of other parties' additive or multiplicative shares. Suppose Alice has a multiplicative share a and Bob has b, then using this method Alice and Bob can obtain the additive shares x and y respectively, such that $x + y = ab$ and vice versa.

Converting Multiplicative Shares into Additive Shares

Suppose Alice holds multiplicative shares a and Bob holds b. Then:

(1) Bob selects independently and randomly θ ring elements denoted by $s_0, \dots, s_{\theta-1} \in \boldsymbol{R}$. $\boldsymbol{R}$ is a ring whose elements can be encoded using θ bits (where $\theta = \log|\boldsymbol{R}|$). Bob then defines θ pairs using the elements in $\boldsymbol{R}$: $(t_0^0, t_0^1), \dots, (t_{\theta-1}^0, t_{\theta-1}^1)$. For every i $(0 < i < \theta-1)$ Bob defines $t_i^0 = s_i$ and $t_i^1 = 2^i b + s_i$.

(2) Let the binary representation of a be $a_0, \dots, a_{\theta-1}$. Alice and Bob execute θ one out of two oblivious transfers [17]. In the i-th execution, Alice chooses $t_i^{a_i}$ from the pair (t_i^0, t_i^1).

(3) Alice calculates $(x = \sum_{i=0}^{\theta-1} t_i^{a_i})$ and Bob calculates $(y = -\sum_{i=0}^{\theta-1} s_i)$.

At the end of this protocol, Alice and Bob hold the additive shares x and y respectively, such that $(x + y = ab)$.

Converting Additive Shares into Multiplicative Shares

Suppose Alice and Bob holds additive shares x and y respectively. Then:

(1) Bob chooses at random r. Then Alice and Bob invoke the protocol for converting multiplicative shares into additive shares to perform $(m_A + m_B = rx)$, where m_A and m_B are held by Alice and Bob respectively.

(2) Bob sends $(m_B + ry)$ to Alice.

Message 1. Alice → Bob: $X_A Y_A, F_3$

Message 2. Bob → Alice: $X_B Y_B, F_6, W_A$,

Alice and Bob convert (F_1, y_B, F_2, x_B) into

$(Sh_{A1}, Sh_{B1}, Sh_{A2}, Sh_{B2})$

Alice and Bob convert (F_4, y_A, F_5, x_A) into

$(Sh_{A3}, Sh_{B3}, Sh_{A4}, Sh_{B4})$

Message 3. Alice → Bob: $W_B, H(J)$

Message 4. Bob → Alice: $H(J)$

Fig. 1 The basic protocol.

(3) Alice calculates $(m_A + m_B + ry = r(x + y))$ and Bob calculates r^{-1}.

At the end of this protocol, Alice holds $(r(x + y))$ and Bob holds r^{-1} as their multiplicative shares.

2.2 Details of the Basic Protocol

Suppose there are two communicating parties Alice and Bob who intend to jointly calculate a number which is a product of two prime numbers $(J = (x_A + x_B)(y_A + y_B)$, where x_A, y_A are Alice's secret numbers, and x_B, y_B are Bob's) such that no party can obtain the factors of J. Assuming that the size of J is agreed upon in advance, they can realize this using the following Basic Protocol (**Fig. 1**):

(1) Alice chooses two large prime numbers X_A, Y_A (where the size of $X_A Y_A$ is a few digits greater than the size of J), a number n_A where $2 < n_A < \phi(X_A Y_A)$ and a number α_A where $(gcd(\alpha_A, X_A Y_A) = 1)$. She then chooses her secret numbers x_A and y_A which will determine the number J, and calculates:
- $F_1 \equiv (\alpha_A x_A^{1-n_A} y_A^{-n_A}) \bmod X_A Y_A$.
- $F_2 \equiv (\alpha_A x_A^{-n_A} y_A^{1-n_A}) \bmod X_A Y_A$.
- $F_3 \equiv (\alpha_A x_A^{-n_A} y_A^{-n_A}) \bmod X_A Y_A$.

She sends F_3 along with $X_A Y_A$ to Bob.

(2) After receiving F_3 and $X_A Y_A$ from Alice, Bob chooses two large prime numbers X_B, Y_B (where the size of $X_B Y_B$ is a few digits greater then the size of J), a number n_B where $2 < n_B < \phi(X_B Y_B)$ and a number α_B where $(gcd(\alpha_B, X_B Y_B) = 1)$. He then chooses his secret numbers x_B and y_B which will determine the modulus number J. Bob then calculates:
- $F_4 \equiv (\alpha_B x_B^{1-n_B} y_B^{-n_B}) \bmod X_B Y_B$
- $F_5 \equiv (\alpha_B x_B^{-n_B} y_B^{1-n_B}) \bmod X_B Y_B$
- $F_6 \equiv (\alpha_B x_B^{-n_B} y_B^{-n_B}) \bmod X_B Y_B$

(3) Alice and Bob convert their multiplicative shares F_1 and y_B into the additive shares Sh_{A1} and Sh_{B1} such that: $(Sh_{A1} + Sh_{B1} \equiv F_1 y_B \bmod X_A Y_A)$. At this point, Alice holds Sh_{A1} and Bob holds Sh_{B1}. They perform a similar procedure for converting their multiplicative shares F_2 and x_B into the additive shares Sh_{A2} and Sh_{B2} such that: $(Sh_{A2} + Sh_{B2} \equiv F_2 x_B \bmod X_A Y_A)$ and Alice holds Sh_{A2} and Bob holds Sh_{B2}. Next, he calculates $W_A = (F_3 x_B y_B + Sh_{B1} + Sh_{B2}) \bmod X_A Y_A$ and then sends F_6, W_A and $X_B Y_B$ to Alice.

(4) Alice and Bob, convert their multiplicative shares F_4 and y_A into the additive shares Sh_{A3} and Sh_{B3} using a method similar to Step (3) such that Alice holds Sh_{A3} and Bob holds Sh_{B3}. They also convert their multiplicative shares F_5 and x_A such that Alice holds Sh_{A4} and Bob holds Sh_{B4} as their additive shares.

(5) After receiving F_6, W_A and $X_B Y_B$ from Bob, Alice calculates:
- $W_B = (F_6 x_A y_A + Sh_{A3} + Sh_{A4}) \bmod X_B Y_B$
- $J \equiv ([W_A + Sh_{A1} + Sh_{A2}] \bmod X_A Y_A \; ([(\alpha_A)^{-1} \; (x_A y_A)^{n_A}] \bmod X_A Y_A) + (x_A y_A) \bmod X_A Y_A$
- $H(J)$, where H is any hash function that has been agreed upon by Alice and Bob in advance

and sends $H(J)$ and W_B to Bob.

(6) Bob calculates:
- $J \equiv ([W_B + Sh_{B3} + Sh_{B4}]) \bmod X_B Y_B \; ([(\alpha_B)^{-1} (x_B y_B)^{n_B}] \bmod X_B Y_B) + (x_B y_B) \bmod X_B Y_B$
- $H(J)$

and sends $H(J)$ to Alice.

At this point, either Alice or Bob compares the $H(J)$ (where $(J = (x_A + x_B)(y_A + y_B))$ sent by the other party and the $H(J)$ he/she just calculated. If the values are equivalent, then both parties will agree on J as their jointly calculated number. If the values are not equivalent it means that one party did not calculate the value of $H(J)$ correctly since he/she does not know the correct hash function H.

3. Proposed Protocol for Generating Shared RSA Parameters

This section first gives an overview of our proposed protocol, and then presents the details of each procedure, i.e., the generation of a modulus number, the generation of shared decryption keys, and the recovery of encrypted messages.

3.1 Overview

Suppose that Alice and Bob wish to generate shared RSA parameters, which include the RSA modulus number N, the encryption key e and the decryption key d. After executing our proposed protocol, N and e are public, while d will be shared between Alice and Bob in a way which enables threshold decryption. Alice and Bob should be convinced that N is indeed a product of two prime numbers, but neither Alice nor Bob knows the factors of N.

The proposed protocol is based on the procedures proposed by Boneh and Franklin [1], especially the primality test, which is used to check that a generated modulus number is valid, and the generation of the shared public/secret key. However, we have extended their procedures to improve computational efficiency without weakening the protocol's security.

The proposed protocol consists of the following three procedures:

(1) The generation of the RSA modulus number N which is the product of two prime numbers p and q. Neither Alice nor Bob knows the factors of N.

(2) The generation of d_A and d_B of the secret decryption key d, for a given encryption key e.

(3) The recovery of an encrypted message.

3.2 Generating Modulus Number N

The RSA modulus number N must be a product of two prime numbers p and q. Our proposed protocol determines N based on a set of secret numbers that Alice and Bob has. Alice and Bob each has two secret numbers (p_A, q_A for Alice and p_B, q_B for Bob), and the two prime numbers are defined as $p = p_A + p_B$ and $q = q_A + q_B$.

The procedure for generating the modulus number consists of two sub-procedures:

(1) Trial generation of the RSA modulus number N from both parties' numbers p_A, q_A, p_B, and q_B.

(2) Primality testing of N.

After generating a number N, both parties have to test the primality of this number. If N is not a product of two prime numbers, then the parties have to repeat the two sub-procedures until they find an N that satisfies the primality test. The rest of this subsection describes the two subprocedures.

3.2.1 Trial Generation of N

The generation of N is based on the Basic Protocol, where J, x_A, y_A, x_B, y_B are substituted with N, p_A, q_A, p_B, q_B, respectively. There are a few conditions put on the Basic Protocol. When Alice chooses p_A and q_A (i.e., x_A and y_A) in Step (1) of the Basic Protocol, we assume that p_A and q_A are congruent to (3 mod 4). Also, when Bob chooses p_B and q_B (i.e., x_B and y_B) in Step (2), p_B and q_B should be congruent to (0 mod 4), so that the resulting modulus number N will be a Blum number [11], which is a number that is congruent to (3 mod 4). The modulus number N should be a Blum number because we will use a primality test which is best suited for Blum numbers [1]. If a non-Blum number is used, the test may leak a few bits of information depending on the power of two dividing $lcm(p-1, q-1)$ [1].

Appendix shows that the resulting N is the product of p and q. However, N may or may not be the product of two prime numbers, and thus the primality test is executed next.

3.2.2 The Primality Test

The primality test is an extension of the test proposed by Boneh and Franklin [1], and consists of three subtests:

- Base test.
- Test I.
- Test II.

The Base Test and Test I are the same as the ones given by Boneh and Franklin, except Test I uses our Basic Protocol for computation.

The three subtests are executed as shown in **Fig. 2**. Test I is executed because the Base Test allows a few invalid numbers to pass. Similarly, Test II is executed because Test I rejects a few valid numbers. If a number does not pass this primality test, another number N must be generated and the primality test is administered again.

Base Test

(1) Alice and Bob agree on a number g, where the Jacobi symbol of g over N or (g/N) must be 1, i.e., $(g/N) = 1$.

(2) Alice calculates ($t_A = g^{(N-p_A-q_A+1)/4}$ mod N) and Bob calculates ($t_B = g^{(p_B+q_B)/4}$ mod N), and they exchange their results. They verify that:

$$t_A \equiv \pm t_B \bmod N$$

If this is FALSE, N is not a valid number. If this is TRUE, N may be a valid number.

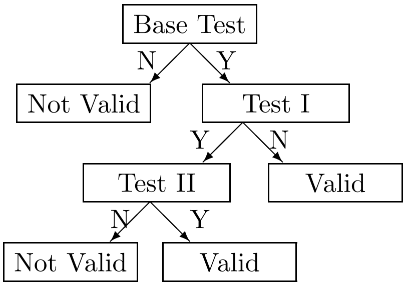

Fig. 2 Primality test.

Test I

If we consider $N = pq$ where:

- $p = r_1^{d_1}$
- $q = r_2^{d_2}$
- $q \equiv 1 \bmod (r_1^{d_1-1})$

the two steps in the Base Test will pass *incorrectly*, i.e., the equation in the Base Test Step (2) is TRUE even though N is not valid. We thus have to further check whether N satisfies the following expression [1]:

$$gcd(p+q-1, N) > 1$$

If this expression is FALSE, N is a valid number, otherwise N may not be a valid number. We realize this test by first calculating:

$$z \equiv R(p+q-1) \bmod N$$

where $R = r_A + r_B$ (r_A and r_B are random numbers chosen by Alice and Bob respectively, whose sizes are less then half of N's). This is jointly generated by both parties using the Basic Protocol.

$$\begin{aligned} z &\equiv [(r_A + r_B)(p+q-1)] \bmod N \\ &\equiv [(r_A + r_B)(p_A + p_B + q_A + \\ &\quad q_B - 1) \bmod N] \end{aligned}$$

where x_A, y_A, x_B, y_B in the Basic Protocol is substituted with (r_A mod N), (($p_A + q_A$) mod N), (r_B mod N), (($p_B + q_B - 1$) mod N).

By assuming that $gcd((r_A + r_B), N) = 1$ (since N is a very large integer, the probability that $gcd((r_A + r_B), N) \neq 1$ is very low) then:
$gcd(p+q-1, N) = gcd((r_A+r_B)(p+q-1), N)$
According to Boneh-Franklin [1]

$$gcd((r_A + r_B)(p+q-1), N) = gcd(z, N)$$

Thus, if $gcd(z, N) > 1$, they will reject this number.

Test II

Unfortunately, Test I will also eliminate a few valid numbers, i.e., moduli $N = pq$ where p, q are prime and ($q = 1$ mod p). To check whether N is a valid prime number, we

Message 1. Alice →Bob: $g^{(q_A^2-q_A) \bmod N} \bmod N$, $g^{q_A \bmod N} \bmod N$
Message 2. Bob → Alice: $g^{(q_B^2-q_B) \bmod N} \bmod N$, $g^{q_B \bmod N} \bmod N$
$g^{(2q_Aq_B) \bmod N} \bmod N$, OK/N-OK
Message 3. Alice →Bob: OK/N-OK

Fig. 3 Protocol for Test II.

check $((q^2 - q) \bmod N \equiv 0 \bmod N)$ or $(g^{(q^2-q) \bmod N} \bmod N \equiv 1 \bmod N)$ using Test II. If this expression is TRUE then N is a valid modulus number, otherwise it should be eliminated.

Test II is executed as follows (**Fig. 3**):

(1) Alice calculates $(g^{(q_A^2-q_A) \bmod N} \bmod N)$ and $(g^{q_A \bmod N} \bmod N)$. Then she sends these values to Bob. We can use the value of g used in the Base Test.

(2) After receiving Alice's message, Bob calculates:
- $g^{(2q_Aq_B) \bmod N} \bmod N$
- $g^{(q_B^2-q_B) \bmod N} \bmod N$
- $g^{q_B \bmod N} \bmod N$

and sends these values to Alice. Bob also calculates:

$$\begin{aligned}\rho \equiv\ & [g^{(q_A^2-q_A) \bmod N} \bmod N] \\ & [g^{(q_B^2-q_B) \bmod N} \bmod N] \\ & [g^{(2q_Aq_B) \bmod N} \bmod N] \\ \equiv\ & g^{((q_A+q_B)^2-(q_A+q_B)) \bmod N} \bmod N \end{aligned} \quad (1)$$

and sends Alice "OK" if the value of ρ is 1, otherwise she will send Bob "N-OK".

(3) Finally, Alice calculates ρ and sends Bob "OK" if the value of ρ is 1 or "N-OK" if the value of ρ is not equal to 1.

"OK" means that the modulus number N is valid. "N-OK" means that N is not valid. Section 5.1 gives the proof for this test.

3.3 The Generation of Shared Decryption Keys

We generate the shared decryption keys based on Gilboa's[15] and Boneh-Franklin's methods.

We now describe the procedure for generating public/secret keys. After both parties have successfully calculated N, they will generate decryption keys d_A and d_B such that $(d_A + d_B = d)$ and $(d = e^{-1} \bmod \phi(N))$ (where $\phi(N)$ is the Euler Totient Function of N) for some agreed upon value of e. Each party will have one of these decryption keys. We will use the combination of the methods proposed by Boneh-Franklin[1] and Gilboa[15]. Our method though will not need the help of a third party.

Since $N = pq$, $p = p_A + p_B$, and $q = q_A + q_B$:

$$\begin{aligned}\phi(N) &= (p-1)(q-1) = N - p - q + 1 \\ &= N - p_A - p_B - q_A - q_B + 1\end{aligned}$$

Suppose:

$$\begin{aligned}\phi(N)_A &= N - p_A - q_A + 1 \\ \phi(N)_B &= -p_B - q_B\end{aligned}$$

Then, we can consider $\phi(N) = \phi(N)_A + \phi(N)_B$ as a sharing of $\phi(N)$ between Alice and Bob. Recall that $(d = e^{-1} \bmod \phi(N))$. Thus, to calculate d both parties need the value of $\phi(N)$, but $\phi(N)$ is shared between both parties in a way that they have to jointly calculate $\phi(N)$ in order to use it. Therefore, in order to calculate $(e^{-1} \bmod \phi(N))$ without first calculating $\phi(N)$, we use an inversion algorithm of Boneh-Franklin as follows:

(1) Choose e, where $gcd(e, \phi(N)) = 1$, i.e., e is odd.

(2) Calculate $\gamma = -(\phi(N))^{-1} \bmod e$.

(3) Calculate $(E = 1 + \gamma\phi(N))$ and observe that $(E \equiv 0 \bmod e)$.

(4) Since $(de \equiv 0 \bmod e)$ then we can set $de = E$. Hence, $d = E/e$ and $d = e^{-1} \bmod \phi(N)$.

To realize this inversion algorithm, we use the following procedure:

(1) Alice and Bob picks a random number l_A and l_B respectively, where $l_A, l_B \in \boldsymbol{Z_e}$. We define $\boldsymbol{Z_e}$ as a set $\{0, \ldots, e-1\}$, equipped with two operations, + and −.

(2) Both parties jointly calculate $(\Psi = (l_A + l_B)(\phi_A + \phi_B) \bmod e)$ using the Basic Protocol. Alice holds ϕ_A, l_A and Bob holds ϕ_B, l_B, where $(\phi_A = (N - p_A - q_A + 1) \bmod e)$ and $(\phi_B = (-p_B - q_B) \bmod e)$.

(3) Each party calculates $(\Psi^{-1} \bmod e)$.

(4) Alice calculates $(\epsilon_A = l_A\Psi^{-1} \bmod e)$ and $(\epsilon_B = l_B\Psi^{-1} \bmod e)$ such that

$$-(\epsilon_A + \epsilon_B) = -\phi(N)^{-1} \bmod e$$

Since $(-\phi(N)^{-1} \bmod e)$ should be hidden, then they will perform the next step instead of broadcasting the values of ϵ_A and ϵ_B.

(5) Both parties choose an arbitrary odd integer P ($> 2N^2e$).

(6) Alice and Bob convert the additive sharing $((-\epsilon_A - \epsilon_B) \bmod P)$ into multiplicative sharing using Gilboa's method. At this point Alice and Bob holds T_A and T_B respectively, where

$$T_AT_B \equiv -(\epsilon_A + \epsilon_B) \bmod P$$

(7) Alice calculates $(T_A\phi_A)$ and both parties share w_A and w_B using Gilboa's method, such that

$$w_A + w_B \equiv T_A\phi_AT_B \bmod P$$

(8) Similarly, Bob calculates $(T_B\phi_B)$, and they share v_A and v_B using Gilboa's method[15] such that

$$v_A + v_B \equiv T_B\phi_BT_A \bmod P$$

(9) At this point $(v_A + w_A + v_B + w_B + 1)$ can be divided by e since

$$\begin{aligned}(v_A + v_B + w_A + w_B) + 1 \\ &= (T_A\phi_AT_B + T_B\phi_BT_A) + 1 \\ &= T_AT_B(\phi_A + \phi_B) + 1 \\ &= -(\epsilon_A + \epsilon_B)(\phi_A + \phi_B) + 1 \\ &= -(\phi(N)^{-1}\phi(N)) + 1 \\ &\equiv 0 \bmod e\end{aligned}$$

We define :

$$de = (v_A + v_B + w_A + w_B) + 1$$

Thus, the decryption key d is as follows:

$$d = \frac{(v_A + v_B + w_A + w_B) + 1}{e}$$

Since the decryption key d has to be shared between Alice and Bob, suppose we set $d = d_A + d_B$ where d_A is Alice's decryption key and d_B is Bob's. Alice can calculate her decryption key d_A as follows:

$$d_A = \left\lfloor \frac{(v_A + w_A) + 1}{e} \right\rfloor \tag{2}$$

while Bob can calculate his decryption key d_B as follows:

$$d_B = \left\lceil \frac{(v_B + w_B)}{e} \right\rceil \tag{3}$$

3.4 Recovery of an Encrypted Message

Suppose Alice sends Bob a message M encrypted with e which is equal to $(y = M^e \bmod N)$. Alice cannot send just this message, since Bob also needs Alice's decryption key to decrypt the message. Therefore, Alice sends Bob y and $(J_1 = y^{d_A} \bmod N)$. Bob can now decrypt the message sent by Alice by first calculating $(J_2 = y^{d_B} \bmod N)$ and obtaining M by calculating $(J_1J_2 \bmod N)$.

If a third party is authorized to obtain the decryption of y, then Alice and Bob have to send the third party J_1 and J_2. The third party may obtain M by calculating the product of J_1 and J_2.

4. Security Analysis

This section analyzes the ability of calculating the factors of modulus number N during the procedure of calculating the modulus number and the shared decryption key, obtaining the factors through the mathematical view and also summarizes security requirements.

4.1 Finding Factors of *N* During the Generation of the Modulus Number

This section will analyze how far the messages which Alice and Bob obtained will leak information for obtaining both parties' secret numbers during the execution of the Basic Protocol and Test II. We use Lemma 1 and Lemma 2 for analyzing the Basic Protocol and Test II respectively.

The following lemma informally means that at the end of the protocol, both parties learn nothing more than the value of N.

Before discussing lemma 1, we first define:

- $View1$: Alice's view when she interacts with Bob.
- $View2$: Alice's view when she guesses all of Bob's secret numbers (during the simulation).
- $View3$: Alice's view when Bob's actual secret numbers are used.

Lemma 1 Given their own private data, Bob and Alice can each simulate the transcript of the Basic Protocol or the protocol is perfectly zero-knowledge[18]. This means that the distribution of the view when they interact with each other is indistinguishable from the distribution that can be computed from the simulation i.e., the following expression holds:

$$\sum_{View_3 \in V} |Prob(View_1 = View_3) - Prob(View_2 = View_3)| \leq |x|^{-c}$$

where $|x|$ is the size of the random input, $c > 0$ and V is the set of all values for $View_3$.

Proof. We show how Alice's view can be simulated or said to be perfectly zero-knowledge, but the same proof holds for Bob. We define the view of Alice to be everything she sees during the execution of the protocol.

To simulate the basic protocol, we have to perform the following procedure:

- Let Alice have the string RP_A in her random tape and she picks x'_B, y'_B, X'_B, Y'_B, α'_B, n'_B which is a substring of RP_A.
- She also has an extra input HS which is the history of previous interactions that Alice is trying to use to get knowledge from Bob.
- Put the variables x_A, y_A, X_A, Y_A, α_A, n_A, in her private input tape.

We prove Lemma 1 by calculating the following probabilities:

(1) Probability of $View1 = View3$.
(2) Probability of $View2 = View3$.
(3) Comparing the above two probabilities. If they are equal, then according to Goldwasser, et al.[14], $View1$ and $View2$ are indistinguishable, i.e., Alice cannot use Bob's message to obtain his actual secret numbers.

The above three steps are also carried out for the cases when Bob guesses Alice's secret numbers.

Each view of Alice depends on Alice's random input RP_A and the history HS. We denote $View1$ as follows:

$$View1 = View_A(RP_A, HS)$$

Suppose Alice guesses Bob's chosen numbers to be x'_B, y'_B, X'_B, Y'_B, α'_B, n'_B, then:

$$View2 = View'_A(RP_A, HS)$$

Furthermore, Alice's view when Bob's actual chosen numbers are x''_B,y''_B,X''_B,Y''_B,α''_B,n''_B is denoted as:

$$View3 = View''_A(RP_A, HS)$$

According to Goldwasser, et al.[14] we can say that the value of $View1$ during the simulation is equal to

$$(H(N), RP_A, HS, F_3, F_6, W_A, W_B)$$

i.e., the concatenation of the seven parameters. The value of Alice's $View_2$ is equal to:

$$(H(N), RP_A, HS, F_3, F'_6, W'_A, W'_B)$$

Similarly, the value of $View3$ is

$$(H(N), RP_A, HS, F_3, F''_6, W''_A, W''_B)$$

We do not include the variable written in Alice's private tape, since it would make no differenc[14]. Let one value of N have at most one value of $H(N)$. Since Alice's private values such as RP_A, HS, x_A, y_A, X_A, Y_A, n_A and α_A are fixed, Alice's functions which depend on these values such as F_1, F_2 and F_3 are also fixed. Furthermore, the probability of obtaining a certain value of Alice's view depends on the probability of finding F_6, W_B and W_A.

We now analyze the probability distribution of F_6, W_A, $H(N)$, W_B as follows:

- Since ($F_6 \equiv \alpha_B(x_B y_B)^{-n_B} \bmod X_B Y_B$), the number of possible values for F_6 depends on the number of possible values of ($\alpha_B \bmod X_B Y_B$) and ($(x_B y_B)^{-n_B} \bmod X_B Y_B$). The number of possible values for ($\alpha_B \bmod X_B Y_B$) is ($\phi(X_B Y_B)$) (where ($\phi(X_B Y_B)$) is the Euler Totient Function of ($X_B Y_B$)) because (α_B and $X_B Y_B$) are relatively prime. Furthermore, since ($x_B y_B$) and ($X_B Y_B$) are relatively prime, the number of possible values for ($(x_B y_B)^{-n_B} \bmod X_B Y_B$) is ($\lambda(X_B Y_B) - 1$) (where ($\lambda(X_B Y_B) = lcm(\phi(X_B), \phi(Y_B))$) and is also called the Charmichael Function[19]). Thus, there are (($\lambda(X_B Y_B) - 1)(\phi(X_B Y_B)$)) possible values for F_6. However, since (($\lambda(X_B Y_B) - 1)(\phi(X_B Y_B)$)) is greater than ($X_B Y_B$), the number of possible values for F_6 is ($X_B Y_B$). This happens because the value of F_6 cannot exceed $X_B Y_B$. In this case, the probability of obtaining the value of (F_6 is $1/(X_B Y_B)$).
- Recall that ($W_B \equiv (Sh_{A3} + Sh_{A4} + F_6 p_B q_B) \bmod X_B Y_B$). Since ($Sh_{A3} + Sh_{A4} + F_6 p_B q_B$) can be congruent with any values less than ($X_B Y_B$), the number of possible values for W_B is ($X_B Y_B$). In other words, the probability for obtaining a certain value of W_B is ($1/(X_B Y_B)$). Similar discussion also holds for W_A, such that the probability for obtaining a certain value of (W_A is $1/(X_A Y_A)$).
- Suppose the size of N is $|N|$ bits, thus the probability for obtaining a certain value of $H(N)$ is ($1/2^{|N|}$).

Let $P(X)$ denote the probability of X. From the above discussion, we know or can determine that the probability of obtaining a certain value for the view of Alice is $1/[(X_B Y_B)^2 (X_A Y_A)(2^{|N|})]$. Thus, the probability of ($View_1 = View_3$) (i.e., the view of Alice taking the same certain value) is $1/[(X_B Y_B)^2 (X_A Y_A)(2^{|N|})]$, and the following two equations hold:

$$P(View1 = View3) = 1/[(X_B Y_B)^2 (X_A Y_A)(2^{|N|})]$$

$$P(View2 = View3) = 1/[(X_B Y_B)^2 (X_A Y_A)(2^{|N|})]$$

Thus,

$$|P(View1 = View3) - P(View2 = View3)\,| = 0 \qquad (4)$$

Since the difference between the two probabilities in Eq. (4) is equal to 0, according to Goldwasser, et al. [14], the view of Alice during the simulation is indistinguishable from the view of Alice when she actually interacts with Bob. Since these are indistinguishable, the protocol is said to be perfectly zero-knowledge. A similar proof also holds true for Bob. ■

The following lemma informally means that at the end of the protocol, both parties learn nothing more than the value of ρ.

Before discussing Lemma 2, we first define:

- $View4$: Alice's view when she interacts with Bob.
- $View5$: Alice's view when she guesses all of Bob's secret numbers (during the simulation).
- $View6$: Alice's view when Bob's actual secret numbers are used.

Lemma 2 Given N, Alice's and Bob's own private data, Bob and Alice can each simulate the transcript of the protocol for Test II or the protocol is perfectly zero-knowledge. This means that the distribution of the view when they interact with each other is indistinguishable from the distribution that can be computed from the simulation i.e., the following expression holds,

$$\sum_{View_6 \in VI} |Prob(View_4 = View_6) - Prob(View_5 = View_6)| \leq |x|^{-c}$$

where $|x|$ is the size of the random input, $c > 0$ and VI is the set of the values of $View_6$.

Proof. For simplicity, let $(Q_{A1} = (q_A^2 - q_A) \bmod N)$, $(Q_{A2} = q_A \bmod N)$, $(Q_{A3} = 2q_Aq_B \bmod N)$, $(Q_{B1} = (q_B^2 - q_B) \bmod N)$, $(Q_{B2} = q_B \bmod N)$.

To simulate the Protocol II, we have to perform the following procedure:

- Let Alice have the string RT_A in her random tape and she picks q'_B which is a substring of RT_A.
- She also has an extra input HST which is the history of previous interactions that Alice is trying to use to get knowledge from Bob.
- Put the variables q_A, in her private input tape.

We prove Lemma 2 in a fashion similar to Lemma 1:

(1) Calculate the following two probabilities:
(2) Probability of $View4 = View6$.
(3) Probability of $View5 = View6$.
(4) Compare the above two probabilities. If they are equal, then according to Goldwasser, et al. [14], $View4$ and $View5$ are indistinguishable, i.e., Alice cannot use Bob's message to obtain his actual secret numbers.

The view of Alice depends on the input written in the random tape RT_A and HST. We can denote $View4$ as follows:

$$View4 = View_A(RT_A, HST)$$

Suppose Alice guesses that Bob chose q'_B, then:

$$View5 = View'_A(RT_A, HST) \qquad (5)$$

When Bob's actual chosen number is q''_B, Alice's view is:

$$View6 = View''_A(RT_A, HST) \qquad (6)$$

According to Goldwasser, et al. [14] we can say from Fig. 3 that the value of Alice's view when she interacts with Bob ($View4$) is:

$$(RT_A, g^{Q_{B1}} \bmod N, g^{Q_{B2}} \bmod N, g^{Q_{A3}} \bmod N,\ \rho,\ HST)$$

(i.e., the concatenation of the six variables). Note that $(\rho \equiv g^{q^2-q \bmod N} \bmod N)$. Similarly, the values of $View5$ and $View6$ are respectively:

$$(RT_A, g^{Q'_{B1}} \bmod N, g^{Q'_{B2}} \bmod N, g^{Q'_{A3}} \bmod N, \rho,\ HST)$$

$$(RT_A, g^{Q''_{B1}} \bmod N, g^{Q''_{B2}} \bmod N, g^{Q''_{A3}} \bmod N, \rho, HST)$$

Since we simulate Alice's view, Alice's private values such as Q_{A1} and Q_{A2} are fixed. Thus, the probability of obtaining a certain value of Alice's view depends on the probability of finding $(g^{Q_{A3}} \bmod N)$, $(g^{Q_{B1}} \bmod N)$ and $(g^{Q_{B2}} \bmod N)$. Since these three functions are modular exponentiation functions and $(gcd(g, N) = 1)$, the numbers of possible values for each function are equal and they are equal to $\lambda(N)-1$ (where $\lambda(N) = lcm(\phi(p), \phi(q))-1)$. Hence, the number of possible values for Alice's view is $((lcm(\phi(p), \phi(q)) - 1)^3)$. Finally, we obtain that the probability for obtaining a certain value of Alice's view (for example $View_6$) is $(1/(lcm(\phi(p), \phi(q)) - 1)^3)$.

Furthermore, the following two equations hold:

$P(View4 = View6) = 1/(lcm(\phi(p), \phi(q)) - 1)^3$
$P(View5 = View6) = 1/(lcm(\phi(p), \phi(q)) - 1)^3$

Thus,

$$|P(View4 = View6) - P(View5 = View6)| = 0 \qquad (7)$$

Since the difference between the two probabilities in Eq. (7) is equal to zero, according to Goldwasser, et al.[14], the view of Alice during the simulation is indistinguishable from the view of Alice when they interact with each other. Since these are indistinguishable, Alice cannot use the values sent from Bob to obtain the actual q_B, i.e., Alice learns nothing about these values.

A similar proof also holds true for Bob. Thus, the transcript is perfectly simulatable. ■

4.2 Finding Factors of N During Shared Decryption Keys Generation

Recently, Susilo, et al.[16] proposed a method (called Miller-Bach algorithm) to find the factors of a number N which has at least two distinct prime factors. The factors of N can be found if one party or both of them know a multiple of the Euler Totient Function $\phi(N)$ and N itself.

We can say that a protocol is secure against Miller-Bach Algorithm, if both parties cannot obtain the multiple of $\phi(N)$. This section shows that our procedure for generating shared decryption keys is secure against the Miller-Bach algorithm.

Lemma 3 In the protocol for generating the shared decryption key, each party does not gain any knowledge concerning the multiple of $\phi(N)$.

Proof. Since Alice and Bob can find the factors of the modulus number N if both parties know a multiple of $\phi(N)$[16], we first check whether both parties can obtain a multiple of $\phi(N)$.

According to the inversion algorithm proposed by Boneh-Franklin[1], both parties can calculate the shared decryption key if they can calculate $([-(\phi(N) \bmod e)(\phi(N)^{-1} \bmod e)] + 1)$ which is a multiple of $(\phi(N) + 1)$. To protect from an attack based on Miller-Bach algorithm, the shared decryption key needs to be calculated without the parties having to calculate $([-(\phi(N) \bmod e)(\phi(N)^{-1} \bmod e)] + 1)$.

As mentioned before, according to Gilboa's method, both parties never broadcast or exchange their additive and multiplicative shares directly, but through oblivious transfer. Thus, since our proposed protocol uses Gilboa's method, Alice and Bob can calculate their shared decryption keys without calculating $([-(\phi(N) \bmod e)(\phi(N)^{-1} \bmod e)] + 1)$. Instead of exchanging their shared Euler Totient function $(\phi(N)_A \bmod e)$ and $(\phi(N)_B \bmod e)$ each party only sends the other party random numbers and pairs of numbers which indirectly include the shared Euler Totient Function (see Section 3.3) using oblivious transfers. This means it is not possible for each party to obtain the other party's shared Euler Totient Function and calculate a multiple of $\phi(N)$, since the multiple of $\phi(N)$ can be obtained only by calculating $([-(\phi(N) \bmod e) \times (\phi(N)^{-1} \bmod e)] + 1)$. ■

4.3 Obtaining Factors of N Using a Mathematical Method

This section will discuss how far either Alice or Bob can obtain the value of the other party's secret numbers using a mathematical method.

Suppose Bob intends to obtain the secret numbers of Alice. He can obtain Alice's secret numbers if he can determine F_1 and F_2 since

$$F_3^{-1} F_1 \bmod X_A Y_A = p_A \bmod X_A Y_A$$

and

$$F_3^{-1} F_2 \bmod X_A Y_A = q_A \bmod X_A Y_A$$

However, since Bob only has the additive shares of $F_1 q_B$ (Sh_{A1}) and $F_2 p_B$ (Sh_{A2}) instead of the values of F_1 and F_2, there is no possibility for Bob to obtain the values of F_1 and F_2 although he can find the multiplicative inverse of F_3. Thus, there is no possibility for Bob to obtain Alice's secret numbers. Similar discussion holds true for obtaining Bob's secret numbers by Alice.

4.4 Security Requirements

Based on the above discussion, there are a few conditions that need to be satisfied to keep the factors of modulus number N secret. These conditions can be summarized as follows:

- Alice and Bob have to agree on the length of the modulus number N in advance.
- Either Alice or Bob has to choose the length of $X_A Y_A$ or $X_B Y_B$ to be a few digits longer than the length of N.
- Alice has to choose p_A and q_A such that the length of $p_A q_A$ is about the length of N. The same holds for Bob.
- Either Alice or Bob has to choose α_A or α_B which is an odd number.

5. Discussion

5.1 Primality Test

Among the three subtests in the primality test, the Base Test and Test I are the same as the ones used by Boneh and Franklin. They have already proven the validity of their primality test. They, however, have noted that when ($N = pq$, $q = 1 \bmod p$), a valid N will be rejected. Our Test II makes these Ns valid. We will next prove the validity of Test II.

Lemma 4 If

$$g^{((q_A+q_B)^2-(q_A+q_B)) \bmod N} \bmod N$$

is congruent to ($1 \bmod N$) then the number N is valid (i.e., N is a product of two prime numbers p and q where $q \equiv 1 \bmod p$ and $q = q_A + q_B$).

Proof. Suppose $N = pq$ and $q \equiv 1 \bmod p$. Then $q = kp + 1$ where k is an integer. Thus:

$$\begin{aligned} q^2 &= q(kp+1) \\ &= kpq + q \\ &= kN + q \\ (q^2 - q) &= kN \qquad (8) \end{aligned}$$

Since $q = q_A + q_B$, Eq. (8) will be as follows:

$$\begin{aligned} (q_A + q_B)^2 - (q_A + q_B) &= kN \\ &\equiv 0 \bmod N \qquad (9) \end{aligned}$$

This means that $g^{((q_A+q_B)^2-(q_A+q_B)) \bmod N} \bmod N$ or ρ (see Eq. (1)) is congruent to ($g^0 \bmod N$) or ($1 \bmod N$) ■

Thus, the Base Test, Test I and Test II have been shown to be valid, and we conclude that only valid Ns will remain after the execution of the Primality Test.

5.2 Comparison with Other Protocols

Protocols have been proposed for jointly generating RSA parameters, such as those proposed by Boneh-Franklin[1], Cocks[2], Poupard-Stern[12] and Gilboa[15]. Our protocol is an improvement over all four protocols.

First, Boneh-Franklin's protocol needs a third party to help both parties calculate the modulus number N. As shown in Section 2, none of the steps in our protocol needs the help of a third party. As for correctness of N, assuming that both parties are honest and correctly carries out our protocol, Appendix shows that N ($= pq$) can be correctly generated, and Section 5.1 shows that the primality test will correctly filter out any invalid N that has been generated. Thus, both parties can obtain the correct value of N without the help a third party.

Second, Cocks' protocol requires the decryption of "$3K$" messages to calculate the modulus number N. This means that both parties have to perform $3K$ modular exponentiations. Suppose the size of N is 1024 bits. Then according to Cocks[2], K will be 433 (where $\frac{(3K)!}{(K!)^3}$ must exceed N^2), i.e., both parties will need to perform 1299 modular exponentiations, while our proposed protocol needs only 9 modular exponentiations. Thus our protocol can reduce the time to generate the modulus number by 9/1299. Since our protocol does not depend on the size of N, this reduction will become larger as the size of N becomes larger.

Third, our protocol needs less modular exponentiations compared with Poupard-Stern's and Gilboa's. Since Poupard and Stern[12] use the ANDOS protocol[13], where for generating modulus number N each party needs to perform at least t modular exponentiations (where t is the size of each party's secret numbers p_{AorB} and q_{AorB}). Suppose the size of p_{AorB} and q_{AorB} is $|N|/2$ (where $|N|$ is the size of N), then each party needs to perform $|N|/2$ modular exponentiations. If the size of N is about 300 bits, then it means that each party has to perform 150 modular exponentiations. Gilboa's method needs about $(h + 3 - h'/2)$ modular exponentiations (where h and h' can be obtained from a table in Gilboa[15]) for generating one modulus number N. According to this table, for a 1024-bits N, $h = 133$ and $h' = 76$. This means that for generating 1024 bits of N Gilboa's method needs 98 modular exponentiations.

Thus, using our proposed protocol each party indeed needs less modular exponentiations than both Poupard-Stern's and Gilboa's.

6. Conclusion

We have proposed a protocol for jointly generating parameters in RSA encryption. Our protocol does not need the help of a third party, and it needs less time to generate the modulus number compared to previous protocols.

Future work includes a protocol for generating RSA parameters among more than two parties and security against malicious party.

References

1) Boneh, D. and Franklin, M.: Efficient Generation of Shared RSA Keys, *Advances in Cryptology – Crypto '97*, LNCS, pp.423–439, Springer-Verlag (1997).

2) Cocks, C.: Split Knowledge Generation of

RSA Parameters, Cryptography and Coding, *Proc. 6th International Conference of IMA*, LNCS, pp.89–95, Springer-Verlag (1997).

3) Feige, U., Fiat, A. and Shamir, A.: Zero-Knowledge Proofs of Identity, *J. Cryptology 1*, pp.77–94 (1988).

4) Fiat, A. and Shamir, A.: How to Prove Yourself: Practical Solution to Identification Problems, *Crypto '86*, pp.186–194 (1986).

5) Ohta, K. and Okamoto, T.: A Modification of Fiat-Shamir Scheme, *Crypto '88*, pp.232–243 (1988).

6) Ong, H. and Schnorr, C.: Fast Signature Generation with a Fiat-Shamir-like Scheme, *Eurocrypt '90*, pp.432–440 (1990).

7) Rhee, M.Y.: *Cryptography and Secure Communications.*, McGraw-Hill Series on Computer Communications (1994).

8) Rivest, R.L., Shamir, A. and Adleman, L.: Method for Obtaining Signatures and Public-Key Cryptosystems, *Comm. ACM*, pp.120–126 (1978).

9) Schneier, B.: *Applied Cryptography: Protocols, Algorithms, and Source Code in C.* John Wiley and Sons (1994).

10) Micali, S.: Fair Cryptosystems, MIT Technical Report, MIT/LCS/TR-579.h (1993).

11) Menezes, A.J., Oorschot, P.C.v. and Vanstone, S.A.: *Handbook of Applied Cryptography*, CRC (1996).

12) Poupard, G. and Stern, J.: Generation of Shared RSA Keys by Two Parties, *Advances in Cryptology – Asiacrypt '98*, LNCS, Vol.1514, pp.11–24, Springer-Verlag (1998).

13) Brassard, G., Crepeau, C. and Robert, J: All-or Nothing Disclosure of Secrets, *Crypto '86*, LNCS, Vol.263, pp.234–238, Springer-Verlag (1987).

14) Goldwasser, S., Micali, S. and Rackoff, C.: The Knowledge Complexity of Interactive Proof System, *SIAM J. Comput.*, Vol.18, No.1, pp.186–208 (1989).

15) Gilboa, N.: Two RSA Key Generation, *Crypto 99*, LNCS, Vol.1666, pp.116–129, Springer-Verlag (1999).

16) Susilo, W., Safavi-Naini, R. and Pieprzyk, J.: RSA-Based Fail-Stop Signature Schemes, *Proc. 1999 ICPP Workshops*, pp.161–166, IEEE Computer Society (1999).

17) Naor, M. and Pinkas, B.: Oblivious Transfer and Polynomial Evaluation, *Proc. 31st STOC*, pp.245–254 (1999).

18) Goldreich, O.: *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*, Springer-Verlag (1999).

19) Adler, A. and Coury, J.E.: *The Theory of Numbers: A Text and Source Book of Problems*, Jones and Bartlett (1995).

Appendix: Calculation of N

This appendix shows why the N calculated by Alice using the Basic Protocol is correct, i.e., $N = pq$. We first review the messages sent by each party in the Basic Protocol:

In the first message Alice sends:

$$F_3 = (\alpha_A)(p_A)^{-n_A}(q_A)^{-n_A} \bmod X_AY_A$$

Then, Alice and Bob execute Gilboa's method for converting their multiplicative shares (F_1 and q_B) into additive shares (Sh_{A1} and Sh_{B1}), such that:

$$Sh_{A1} + Sh_{B1} \equiv F_1 q_B \bmod X_AY_A$$

At this point Alice holds Sh_{A1} and Bob holds Sh_{B1}.

Using the similar method, they convert their multiplicative inverse F_2 and q_B into their additive shares Sh_{A2} and Sh_{B2}.

After executing Gilboa's method for converting their multiplicative shares into additive shares, Bob calculates:

$$\begin{aligned} W_A &= (F_3 p_B q_B + Sh_{B1} + Sh_{B2}) \bmod X_AY_A \\ &= (\alpha_A)((p_A)^{-n_A}(q_A)^{-n_A}(p_B q_B \bmod X_AY_A \\ &\quad + Sh_{B1} + Sh_{B2}) \bmod X_AY_A \end{aligned}$$

After receiving the second message, Alice calculates N as follows:

$$\begin{aligned} N &= [(W_A + Sh_{A1} + Sh_{A2}) \bmod X_AY_A][(p_A)^{n_A} (q_A)^{n_A}(\alpha)^{-1}] + (p_Aq_A) \bmod X_AY_A \\ &= [(F_3p_Bq_B + Sh_{B1} + Sh_{B2} + Sh_{A1} + Sh_{A2}) \bmod X_AY_A][((p_A)^{n_A}(q_A)^{n_A})(\alpha_A)^{-1} \bmod X_AY_A] + p_Aq_A \bmod X_AY_A \\ &= [(F_1q_B + F_2p_B + F_3p_Bq_B) \bmod X_AY_A] [((p_A)^{n_A}(q_A)^{n_A})(\alpha_A)^{-1} \bmod X_AY_A] + p_Aq_A \bmod X_AY_A \\ &= [\alpha_A(p_A)^{1-n_A}(q_A)^{-n_A}q_B \bmod X_AY_A + \alpha_A (p_A)^{-n_A}(q_A)^{1-n_A}p_B \bmod X_AY_A + \alpha_A (p_A)^{-n_A} (q_A)^{-n_A}p_Bq_B \bmod X_AY_A][(p_A)^{n_A} (q_A)^{n_A}(\alpha_A)^{-1} \bmod X_AY_A] + (p_Aq_A) \bmod X_AY_A \\ &= [p_Aq_B + p_Bq_A + p_Bq_B] \bmod X_AY_A + [(p_Aq_A) \bmod X_AY_A] \\ &= (p_Aq_B + p_Bq_A + p_Bq_B + p_Aq_A) \bmod X_AY_A \end{aligned}$$

As described in Sectiion 2.2, Alice has to choose (X_AY_A) to be a few digits greater than the size of N, which means that N should be the residue of $([p_Aq_B + q_Ap_B + p_Bq_B + p_Aq_A] \bmod X_AY_A)$. From the residue of the above equation Alice may obtain: $N = [p_Aq_B + q_Ap_B + p_Bq_B + p_Aq_A]$ or $N = (p_A + p_B)(q_A + q_B)$. This method can also be applied for calculating N by Bob.

(Received November 11, 1999)
(Accepted July 1, 2000)

Ari Moesriami Barmawi received the B.E. in Electronic Engineering from Bandung Institute of Technology (Indonesia), in 1985. She was a system engineer in Computer Centre of Indonesian Aircraft Industry during 1986–1992. Since 1993 she has been a lecture of the Polytechnique of Bandung Institute of Technology. She received M.E. in Computer Science from Keio University in 1997. She is currently a Ph.D. Candidate in Computer Science at Keio University. Her interest is in computer security.

Shingo Takada received the B.E. in Electrical Engineering, and M.E. and Ph.D. in Computer Science from Keio University, Yokohama, Japan in 1990, 1992, and 1995, respectively. From 1995 to 1999, he was a Research Associate at Nara Institute of Science and Technology. He is currently an Assistant Professor at Keio University (Faculty of Science and Technology). His interests include information exploration, software engineering, and cognitive science. He is a member of the Information Processing Society of Japan, Japan Society for Software Science and Technology, Japanese Society for Artificial Intelligence, ACL, ACM, and IEEE-CS.

Norihisa Doi has been a professor in the Department of Computer Science at Keio University since 1986, and has been the director of the Institute of Computer Education at Keio University since 1991. He received a B.E., M.E., and Ph.D. in computer science from Keio University in 1964, 1966, and 1975, respectively. During 1975–1976, he was a visiting faculty member in the Department of Computer Science at Carnegie-Mellon University. In 1976, he was a visiting professor in the Computer Communication Networks Group at the University of Waterloo. He was a member of the executive boards of the Japan Society for Software Science and Technology (JSSST), the Information Processing Society of Japan (IPSJ), the Japanese Society for Artificial Intelligence (JSAI), and the Japan Society for Security Management, the chief editor of the transactions of the JSSST, the chief editor of the transactions of the IPSJ, the head of the Japanese delegation of ISO/IEC JTC1/SC22. Currently, he is a member of the Science Council of Japan, the chairman of the National Committee of Information and Computer Science, a chairman of OMG (Object Management Group) Japan-SIG, a councilor for the Japan Society for Software Science and Technology, the chairman of the Technical Committee of the Information Technology Promotion Agency (IPA), the chairman of the Committee for Investigating the Standardization of Agent-Oriented Computing of the Japanese Standards Association, and the chairman of the Business Object Consortium. He has written many research papers and books, including "Introduction to the Language C" (Iwanami Shoten), "Introduction to BASIC and FORTRAN" (Iwanami Shoten), "Functions and Organization of Operating Systems" (Iwanami Shoten), "How to Program" (Iwanami Shoten), "Encyclopedia * Basics of Computers" (Iwanami Shoten), "Introduction to PASCAL" (Baifukan), "Introduction to FORTRAN 77" (Baifukan), and "Computer Security" (Kyouritu Shuppan).