Privacy Preserving Pattern Matching: Implementation Issues

BY Risanuri Hidayat

Submitted to the Faculty of Information and Communication Technology
in partial fulfilment of the requirements for the Master Degree
IN INFORMATION AND COMMUNICATION TECHNOLOGY
At Agder University College (Norway)

Grimstad, May 2002

Abstract

The growing of the Internet has the potential to erode personal privacies. One can monitor a private data through the Internet without data owner knowledge.

The project is to implement technique that allow pattern matching on user data and still preserve user privacy. Such technique gives possibility to place data on insecure third party site and be able to perform matching without revealing information about either data or pattern. The system uses public key cryptography to protect data on third part insecure site and to control revealing of data. The data owner participate in data decryption together with permitted data user.

Contents

Motto:

Surely with difficulty is ease
(Q: 94:05)

Acknowledgements

Thanks to Høgskolen i Agder for not putting any constraints on this project whatsoever and for giving me the opportunity to be creative, explore new technology and pursue those directions I deemed appropriate. Thanks to Gadjah Mada University for giving me supports and opportunity to the study. These same thanks attributes to my supervisor Vladimir Oleshchuk.

Thanks to my family, especially to my wife, for supporting me for the duration of the study.

Problem Description

Let us begin with an example to explain the problem we work on in this project.

Let *A* be a doctor who has patients. Using Internet, this is possible that he inserts data about the patients in a database system in a place, such that it can be loaded either from home or office (hospital). Because the data is about patients, their privacy has to be protected. The one way to protect data is to encrypt them. Only very limited number of persons with his permission can access the data.

When a patient go to another doctor, called *B*, *B* can ask histories (data) about the patient to *A*. *A* will give only data about the patient, but no other data else. In this case, *A* doesn't want to distribute the decryption key to another person, his colleagues. Because once one has the key, one might access other data (other patients) with the same key without his permission. So, *A* holds the decryption key self. If his colleague, *B*, want to get a certain data, *B* can load from the database, and *A* will read (decrypt) the data for *B*.

In this project, we develop a system to solve problem presented above. We can outline the problem description as follows.

- To develop a system to solve problem when data with high protection needs is saved in a third place. To secure the data, the owner encrypts them, and holds the key, so other people even the database manager cannot analyse the data. If one wants to access the data, although one can load the data from the database system, one must ask decryption to him before being able to use the data.

- Since data is encrypted, no one can match it to get information about the data. In pattern matching, input must match with part of data. In the classical way, one must encrypt the input, so it can be matched in the data. But we cannot use the method because it means that the key becomes distributed to others, and the data becomes insecure. Instead, we make a table that transforms the pattern (input) to the encrypted data. To prevent eavesdropping, the input is hashed before sent.

Chapter 1 Introduction

Recently the Internet has seen tremendous growth, with the ranks of new users swelling at ever-increasing rates. The Internet technology is now becoming the world's largest public electronic place. If in the recent of years, usually data is saved in a server that is the same place as the owner, now with the Internet, one can rent a server or database in a third place and save the data in the place.

Besides the potential growing of the Internet, it has the potential to erode personal privacies. The impact on personal privacy is enormous. The data owner might be not realizing that the data in the Internet have been monitored or logged by some unseen third party. To enhance privacy preserving, some research and commercial applications is done. Some author published their researches and applications in order to enhance the privacy preserving [1, 4, 5, 6, 9, 11].

The project is the implementation of the privacy preserving. We make a system that supports data saved in an insecure third place in the Internet. The data is protected with encryption such that it is secure in the place. Only the owner has the private key and is able to decrypt data. A client who wants to get the data must ask the owner to decrypt the data before he/she can read it.

We apply pattern matching for the system. The owner protects the data with encryption and makes hash table for it. A user that wants to get the data sends a query-containing pattern he wants to match in the data. The query is hashed and sent to the administrator. If the query is matched with the table in the administrator, the data can be sent. The user can use the data after the data has been decrypted by the owner. We use algorithm developed by Oleshchuk for this goal [11].

We also apply the system if there is more than one owner of data. It can be for example two persons produce data together and they have copyright for that. Everyone can use the data with their permission. We limit for two owners to make simple the system. But it can be expanded to be more owners. The data is encrypted in the administrator, and there are only the owners that share the keys. The shared keys are never be revealed. A user must have permission from all of the two owners before he/she can get the data. The owners hold the same public key and they share the private key. Therefore if one of the owners wants to use the data, he also must have permission also from the other. Some algorithms have been developed to share the private key [2, 3, 7, 8, 9].

The thesis is organized as follows. Chapter 1 is Introduction that previews about the project. Chapter 2 discuss about the background and the related works. This discuss about some related research for this project. We discuss details the papers by Oleshchuk and Boneh in the chapter since it is most used in the project. We also discuss about the Java Security technology in Chapter 3 since we use Java programming in this project. Then we discuss about the system scenario in Chapter 4. Chapter 5 and 6 shows the implementation and the result. Finally, Chapter 7 is the conclusion and the planning for future research.

Chapter 2 Related Works and Background

In the section we discuss about papers related with the project. The main paper is "Private Informational Retrieval by Pattern Matching" by Oleshchuk [11]. We use the algorithm in the paper in our implementation with some modifications. We also discuss about other papers especially in threshold cryptography topics. The threshold cryptography gives possibility to divide a (private) key to become shared keys [10].

2. 1 Privacy Preserving

The term privacy is usually described as "the right to be let alone," and is related to solitude, secrecy, and autonomy. However, when associated with consumer activities that take place in the arena of the Internet, privacy usually refers to personal information. The invasion of privacy is usually interpreted as the unauthorized collection, disclosure, or other use of personal information.

The increased use of the Internet for everyday activities is bringing new threats to personal privacy. Internet holds a tremendous potential for businesses and consumers, but it may also cause privacy violations. The awareness of privacy preserving began when the Internet was quickly becoming the world's largest public electronic market place. Some research and commercial applications have been done in order to solve the privacy-preserving problem.

Goldberg presented an overview of existing and potential privacy enhancing technologies for the Internet in the past, present and also for future research [5]. Wang outlined taxonomy to categorize and analyze consumer privacy concerns [4]. Organization, like Truste, which concern about privacy for the Internet, is formed to enhance privacy protection [6]. Some researches have also been done to enhance privacy protection. Lau proposed a set of guidelines for designing privacy interfaces in a automated sharing of web browsing system, called COLLABCLIO [9]. Chor did research for privacy preserving by keywords [1], and followed by Oleshchuk using pattern matching [11].

This project implements Privacy Preserving Pattern Matching that enables users perform pattern matching on remote databases located on insecure third place without revealing information about pattern and data from the database. Oleshchuk on his paper proposed a new method to protect privacy of user's queries and avoid disclosure of data contained in databases during searching and retrieval. The method works by searching directly in encrypted data without knowing a decryption key.

The paper presented three algorithms, which are Source Data Pre-processing Algorithm, Pattern Pre-processing Algorithm, and Matching Algorithm. We have tested the algorithms, and we will discuss more detail about the algorithms since they are much used in the project.

2. 1. 1 Source Data Pre-processing

Source Data Pre-processing algorithm transforms source data into data structure that allows pattern matching on encrypted data. It is used when a person (owner) wants to save data. When the person wants to save data in a third place, with this algorithm, he creates encryption of data ($E (T) = (E (t_1) E (t_2) \ldots E (t_l) = b_1 b_2 \ldots b_n$). He also creates hash function of the encrypted data for each encrypted data. For example, the encrypted data $E(t_1)$ has many hash functions that hashes all possibility of substrings t_1.

Finally, hash functions *h (E (u))* are stored in a hash table and used as reference to point out the data. Both the encrypted data and the hash table are located in the database. Figure 2.1 shows the algorithm.

We have tested the algorithm in our system, and it works well. Let us make an example to describe the process of the algorithm in our testing. We called the place where data is saved as administrator. But in our system, the encryption process before hashing is erased. The next section will discuss about the reason.

```
Source Pre-processing (T, h, E)
Input: Source data $T = t_1 t_2 \dots t_n$, hash function h and
        encryption function E where $\|t_i\| = l$ bits for i=1,2,..,n-1
        and $1 \le \|t_n\| \le l$
Output: Encrypted data $E(T) = b_1 b_2 \dots b_n$ and
        hash table $H_T$ $E(T) \leftarrow b_1 b_2 \dots b_n$ where $b_i \leftarrow E(t_i)$, i=1,2,..,n
        for each $i \in \{0,1,..,|H_T|\}$ do
        $H_T(i) \leftarrow \Phi$
        While $i \le n$ do
            For each u from $S_{1,l}(t_i)$ do
                $c \leftarrow h(E(u))$
                $H_T(c) \leftarrow H_T(c) \cup \{i\}$
            $i \leftarrow i+1$
        return E(T), $H_T$
        end;
```

Figure 2.1 Source Data Pre-processing Algorithm

This is the example text (*T*) that created by the owner.

Yusuf Kurnia Badriawan

T is then broken into small blocks such that $T = t_1 t_2 t_3 t_4 t_5$. For example

t_1 =*Yusuf*
t_2 = *Kurn*
t_3 =*ia Ba*
t_4 =*driaw*
t_5 =*an*

Each t_i then is encrypted, therefore

$$E(T) = E(t_1)\ E(t_2)\ E(t_3)\ E(t_4)\ E(t_5)$$

Each block has sub string. t_1 =*Yusuf* for example has sub string

$S_{11} = Y,\ S_{12} = Yu,\ S_{13} = Yus,\ S_{14} = Yusu,\ S_{15} = Yusuf$
$S_{24} = usuf\ S_{33} = suf\ S_{42} = uf\ S_{51} = f$

The All sub strings are then hashed.

$h_1 = h(Y), h_2 = h(Yu), h_3 = h(Yus), h_4 = h(Yusu), h_5 = h(Yusuf)$
$h_6 = h(usuf)\ h_7 = h(suf)\ h_8 = h(uf)\ h_9 = h(f)$

All the sub string are hashed and sent to the administrator. To make clear that these hashed are coming from block 1 (t_1), we should change the index. We arrange $h_{i\,j}$ such that *i* is the block number and j is the hashed number. Therefore,

$h_{1\,1} = h(Y), h_{1\,2} = h(Yu), h_{1\,3} = h(Yus), h_{1\,4} = h(Yusu),$
$h_{1\,5} = h(Yusuf)\ h_{1\,6} = h(usuf)\ h_{1\,7} = h(suf)\ h_{1\,8} = h(uf)$
$h_{1\,9} = h(f)$

The same process is done for the other blocks. The encrypted text (*E(T)*) and the hashed sub strings then is sent to the administrator. The hashed sub strings is saved in a table in the administrator. The table is such as a matrix that the rows refer the number of block, whereas the columns refer the hashed sub string. These are in the administrator:

E(T) :

E(Yusuf)	*E(Kurn)*	*E(ia Ba)*	*E(driaw)*	*E(an)*

$h(E(S_{i\,j}))$:

Blk no.						
1.	h_{11}	h_{12}	h_{13}	h_{14}	...	h_{1n}
2.	h_{21}	h_{22}	h_{23}	h_{24}	…	h_{2n}
3.	h_{31}	h_{32}	h_{33}	h_{34}	…	h_{3n}
					…	
n.	h_{n1}	h_{n2}	h_{n3}	h_{n4}	…	h_{nn}

2. 1. 2 Pattern Pre-processing

The algorithm is used when a person (user/client) wants to get data. He creates a query (*P*). The algorithm then sets all partitions of *P*, called $p_1\ p_2\ ...\ p_m$, and encrypted them to become $E(p_1)\ E(p_2)\ ...\ E(p_m)$. After that all of the encrypted partitions is hashed, and becomes $h(E(p_1))\ h(E(p_2))\ ...\ h(E(p_m))$. Formally,

$$Q_P = \{\ h(E(p_1))\ h(E(p_2))\ ...\ h(E(p_m))\ |\ \textit{for all}\ (p_1\ p_2\ ...\ p_m) \in Part_l\ (P)\ \}$$

For cryptographic matching the client sends Q_P instead of *E(P).* The algorithm for Pattern Pre-processing is presented on Figure 2.2.

```
Pattern Pre-processing (P, h, E)
Input: Pattern P, hash function h and encryption function E
Output: Set $Q_P = \{(c_1\ c_2 \ldots c_m)\}$ corresponding to pattern P
        where $c_i= h(E(p_i))$ for some $(p_1, p_2, .. , p_m)$ from Part (P)

        $Q_P \leftarrow \Phi$
        For each $p_1\ p_2 \ldots p_m$ from Part (P) do
                $Q_P \leftarrow Q_P \cup (h(E(p_1), h(E(p_2),\ldots, h(E(p_m))$
        return $Q_P$
        end;
```

Figure 2.2 Pattern Pre-processing Algorithm

In the test, we discovered that the encryption of partitions is no longer needed, and can be passed. Therefore the partitions of the query *(P)*, $p_1\ p_2 \ldots p_m$, are directly hashed, instead of encrypted before hashed. With this method (without encryption), it saves much time because encryption process is passed. However the query is still secure because it is hashed. This is the reason the hash table in the database also need to change without encryption.

The algorithm in the test works as follows. Let us continue with the example above, the query from the user is for example

Badriawan

We shift the query and make partitions of that. The partitions of the query becomes

$S_{15} = Badri,\ S_{64} = awan$
$S_{14} = Badr,\ S_{55} = iawan$
$S_{13} = Bad,\ S_{45} = riawa\ S_{91} = n$
$S_{12} = Ba,\ S_{35} = adriaw\ S_{82} = an$
$S_{11} = B,\ S_{25} = adria\ S_{73} = wan$

The all partitions become patterns from the user after being encrypted and hashed.

$c_1 = h(Badri),\ c_2 = h(awan)$
$c_3 = h(Badr),\ c_4 = h(iawan)$
$c_5 = h(Bad),\ c_6 = h(riawa),\ c_7 = h(n)$
$c_8 = h(Ba),\ c_9 = h(driaw),\ c_{10} = h(an)$
$c_{11} = h(B),\ c_{12} = h(adria),\ c_{13} = h(wan)$

The user sends these patterns to the administrator.

2. 1. 3 Matching Algorithm

The algorithm occurs in the database where data is saved. When the hash patterns from a client come, they are matched with hash table in the database. If the pattern matches, then algorithm returns the encrypted data. The matching algorithm is presented in Figure 2. 3.

```
Algorithm Matcher ($Q_P$, $E(T)$, $H_T$)
Input: Set $Q_P = \{(c_1 c_2 \dots c_m)$, encrypted data $E(T) = b_1 b_2 .. b_n$ and
          hash table $H_T$
Output: Blocks $(b_{k-m+1}, \dots, b_{k+1})$ where E(P) may occur

          $i \leftarrow 1$
          Next $\leftarrow H_T(c_i)$
          While $i \leq m$ and Res $\neq \Phi$ do
                    Res $\leftarrow H_T(c_i) \cap$ Next
                    Next $\leftarrow \Phi$
                    For each k from Res do
                              Next $\leftarrow$ Next U $\{k+1\}$
                    $i \leftarrow i+1$
          if Res $\neq \Phi$ then
                    Posible matching in blocks $b_{k-m+1}, \dots, b_{k+1}$ for
                    each k Є Res
                    Return $(b_{k-m+1}, \dots, b_{k+1})$ for each k Є Res
          else
                    return ′No Match′
          end;
```

Figure 2.3 Matching Algorithm

From the example above, we can see that

$c_8 = h(Ba)$ are matched with block 3.
$c_9 = h(driaw)$ are matched with block 4.
$c_{10} = h(an)$ are matched with block 5.

Therefore the administrator sends block 3, 4, 5 to the user, that is

E(ia Ba)	*E(driaw)*	*E(an)*

2. 1. 3 Protocol of Getting Data

The data from administrator is still encrypted, so the user cannot use it. The user must ask encryption to the owner. Oleshchuk on his paper developed a protocol to process of user from getting data from administrator until he can use the data. The protocol works as follows.

1. User generates set of Q_P based on pattern *P*, and hash function *h.*
2. User sends Q_P to the administrator as database service provider.
3. User receives set of encrypted blocks

$$E\ (T) = (E\ (t_{k-m+1})\ \dots\ E\ (t_{k+1})$$

4. User encrypts the blocks with own encryption function E_u and sends it to the owner to ask decryption. This is the blocks that are sent to the owner.

$$(E_u(E\ (T)) = (E_u(E\ (t_{k-m+1}))\dots\ E_u(E\ (t_{k+1}))\ )$$

5. The owner of decryption key (*D*) decrypts the block, and sends back the decryption to the user.

$$D(E_u(E\ (T)) = (D(E_u(E\ (t_{k-m+1})))\ldots D(E_u(E\ (t_{k+1})))\ )$$

6. User decrypts the blocks with his own private function D_u and reveals result

$$\begin{aligned} Du(D(E_u(E\ (T)))) &= D_u(D(E_u(E\ (t_{k-m+1}))))\ldots D_u(D(E_u(E\ (t_{k+1})))) \\ &= t_{k-m+1}\ \ldots\ t_{k+1} \end{aligned}$$

Since $Du(D(E_u(E\ (T)))) = D(D_u(E_u(E\ (T))))$

We implement the protocol using RSA encryption/decryption described in number 4 and 5 in the asking for decryption and it works well. We also investigate another method using secret encryption/decryption with key agreement first. Some author proved that using secret key is much more quickly than using public/private key [12, 15]. To generate the same secret key in the both side, we implement Key Agreement with Diffie-Hellman algorithm.

So, from the example, the text is becoming

ia Badriawan

2. 2 Threshold Cryptography

Threshold cryptography means that the secret is shared into some pieces and all the pieces then stored in different places. To reconstruct the secret, one must have some or all of them. Some authors have done research about this topic and many algorithms have been developed.

2. 2. 1 Concepts

Paper proposed by Shamir [7] might be clear to describe the concepts of threshold cryptography. His paper presented threshold cryptography with Lagrange Interpolation Polynomial scheme.

To share secret, first we choose a prime (*p*) as modulus, then generate an arbitrary polynomial of degree m-1 for m sharing. For example, let a message (*M*) be 11. The message will be saved into three different places, so that we only are able to reconstruct data if we have all of them. We generate quadratic equation (3-1=2).

$$F(x) = a\,x^2 + b\,x + M\ (mod\ p)$$

Here, a and b are chosen randomly, for example

$$F(x) = 7\,x^2 + 8\,x + 11$$

The shared data become:

$$\begin{aligned} F\ (1) &= 7 \quad + 8 \quad + 11 && = 0\ (mod\ 13) \\ F\ (2) &= 28 \quad +16 \quad + 11 && = 3\ (mod\ 13) \\ F\ (3) &= 63 \quad +24 \quad + 11 && = 7\ (mod\ 13) \end{aligned}$$

To reconstruct data, we must have all of the shared data.

$$0 = a\,1^2 + b\,1 + M \pmod{13}$$
$$3 = a\,2^2 + b\,2 + M \pmod{13}$$
$$7 = a\,3^2 + b\,3 + M \pmod{13}$$

The solution will be $a = 7$, $b = 8$, and $M = 11$. So M is recovered.

We shared a key in this project instead of the data (message), because to share a key is much simpler than to share data. Data can be very large such that it needs much place. If we must store data in some different place for threshold scheme, so it needs much more place. However to share key is relatively little, and fixed.

2. 2. 2 Shared Key

Threshold cryptography presented by Boneh and Malkin [10] is adopted in this project. They proposed a system that called *intrusion tolerant*. The system protects an application private key by sharing the key among a number of share servers. Figure 2.4 shows the system presented by Boneh.

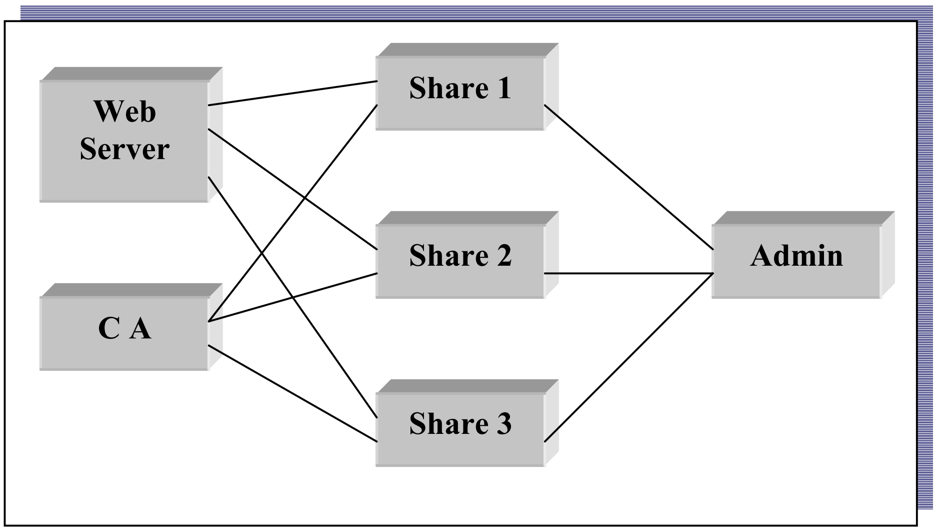


Figure 2.4 System components developed by Boneh

The system consists of three components; these are client that can be Web Server or CA, share servers, and Administrator. Share servers hold shares of the private key belonging to the clients obey serve. These shares reveal no information about the client's private key. It is desirable that an attacker not be able to compromise the share server. However, the intrusion tolerant ensures that even if a few of the share servers are penetrated and the secrets are exposed or corrupted, there is no compromise to overall system security. The attacker learns nothing from penetrating a few of the share server. The all share servers must be corrupted to get the real secret of the system.

A client is any application (Web Server, CA) that makes use of the client side Threshold library in the system. After authentication, the client interacts with the share servers to sign a message or decrypt a given ciphertext using the shared private key stored in the share servers.

Administrator is provided in the system to administer the all share server.

We adopt the system presented by Boneh, which the system consist of three components, Web Server (User), Share Server (Owner), and Administrator. The user in our implementation is a web server that supports JSP/servlets like Tomcat. The share servers are owners who hold the shared private key. Administrator is for saving data and manages data. Administrator can help the owner to generate the shared key.

As we know that a RSA private key consists of a modulus N and a secret exponent d, while public key consists of N and exponent e, where $e.d = 1 \mod (p-1)(q-1)$ and $N=p.q$. To encrypt a message (data) M, one computes $C = M^e \mod N$, and to decrypt a ciphertext C one computes $M = C^d \mod N$. The basic idea to share the private key is to split d into pieces, such that

$$d = d_1 + d_2 + \ldots$$

We then store the share d_i on the share server.

To decrypt a ciphertext each server computes

$$\begin{aligned} M_1 &= C^{d1} \mod N \\ M_2 &= C^{d2} \mod N \\ &\quad | \end{aligned}$$

The message can be recovered by computing

$$M = M_1 M_2 \ldots$$

Since,

$$\begin{aligned} M &= C^d \mod N \\ &= C^{d1 + d2 ..} \mod N \\ &= C^{d1} C^{d2} \ldots \mod N \\ &= M_1 M_2 \ldots \end{aligned}$$

2. 2. 3 Shared Key Generation

We will discuss about distributed for generation of the shared key. The simple way to generate public and (shared) private key is that there is no secret between the share servers (owners), but communication might be secure between them. It is centralized generation key. In this method, one of the share servers generates RSA key, public and private keys, and then distribute the public key to the others. Besides that, the share server also picks a random value d_1 less than d (exponent private key) such that $d = d_1 + d_0$. The share server holds d_1 as its shared private key. d_0 then is sent to the next share server. If there is more than two share servers, the second share server picks also a random value d_2 such that $d_0 = d_2 + d_{00}$. It holds d_2 as its shared private key, and sends d_{00}, and soon. Therefore, all of the share server can get the shared private key. The private key becomes $d_1, d_2, \ldots$ where $d = d_1 + d_2 \ldots$. Here, d is then discarded.We have implemented this method in this project, and it works well.

The other method is much more complicated, but it is much more secure also. Even each share server doesn’t know the secret value generated by the others. Some researches have been done to discover the effective way of the generation [2, 3, 8]. Some use third party for helping generation, but it is still secure, no reveal about secrets value. Based on this algorithm, we design that each owner reveals no information about the secret values, and each owner does not know about any other shared decryption key. The owners engage the protocol with help of Administrator.

As the RSA generation key algorithm that is doing the following steps: generate two random primes' *p* and *q*, compute *N,* select a random integer *e*, and finally compute *d,* we propose the following protocol to generate the shared keys:

1. Each owners pick random integer values p_1, q_1, and p_2, q_2.
2. **Compute N**. Using a private distributed computation the owners compute

$$N = (p_1+p_2)(q_1+q_2).$$

   N becomes public but this step reveals no information about the secret values p_1, q_1, p_2, q_2. Section 2.2.3.1 describes the step.
3. **Primality test**. The 2 owners engage in private distributed computation to test that *N* is product two primes. If test is fail, the protocol is restarted from step 1. Section 2.2.3.2 describes the step.
4. **Key generation**. Given a public encryption key *e*, the owners generate a shared decryption key d_1 and d_2 using a private distributed computation. Section 2.2.3.3 describes the step.

The next section describes about the generation of the shared keys with distributed computation.

2. 2. 3. 1 Distributed Computation of *N*

The same as the RSA key generation, we have to compute *N* in the first step. But with the distributed computation, each party are not allowed to reveal any secret values p_i and q_i. We develop the algorithm that is coming from Franklin and Boneh [2] and the others [3, 8]. Let *A* and *B* be the owners. The algorithm needs a helper (*H*) to compute *N*. The algorithm works as follow:

1. After *A* has picked random integer values p_1 and q_1, *A* picks a random value a_1 and a_2. He sets

$$A_1(x) = a_1 x + p_1$$
$$A_2(x) = a_2 x + q_1$$

   He computes

$$A_1(1) = a_1 + p_1 \qquad A_2(1) = a_2 + q_1$$
$$A_1(2) = 2a_1 + p_1 \qquad A_2(2) = 2\,a_2 + q_1$$
$$A_1(3) = 3a_1 + p_1 \qquad A_2(3) = 3\,a_2 + q_1$$

   He sends the values to *H.*
2. *B* picks a random integer values p_2 and q_2, and picks a random value b_1 and b_2. He sets

$$B_1(x) = b_1 x + p_2$$
$$B_2(x) = b_2 x + q_2$$

He computes

$$B_1(1) = b_1 + p_2 \qquad B_2(1) = b_2 + q_2$$
$$B_1(2) = 2b_1 + p_2 \qquad B_2(2) = 2\,b_2 + q_2$$
$$B_1(3) = 3b_1 + p_2 \qquad B_2(3) = 3\,b_2 + q_2$$

He sends the values to *H*

3. *H* computes

$$C(1)= (A_1(1)+ B_1(1))(A_2(1)+ B_2(1)) = (a_1 + p_1 + b_1 + p_2)(a_2 + q_1 + b_2 + q_2)$$
$$C(2)= (A_1(2)+ B_1(2)(A_2(2) + B_2(2)) = (2a_1 + p_1 + 2b_1 + p_2)(2a_2 + q_1+ 2b_2 + q_2)$$
$$C(3)= (A_1(3)+ B_1(3)(A_2(3) +B_2(3)) = (3a_1 + p_1+ 3b_1 + p_2)(3a_2 + q_1 + 3b_2 + q_2)$$

C becomes the quadrate equation with points at *C(1), C(2)* and *C(3).*

$$C(x)= a\,x^2+ b\,x + c$$
$$C(1)= a \quad + b \quad + c$$
$$C(2)= a\,4 + b\,2 + c$$
$$C(3)= a\,9 + b\,3 + c$$

H computes $C(0) = c = (p_1 + p_2)(q_1 + q_2) = N$ and sends back to *A* and *B*. Figure 2.5 shows the process in the sequence diagram.

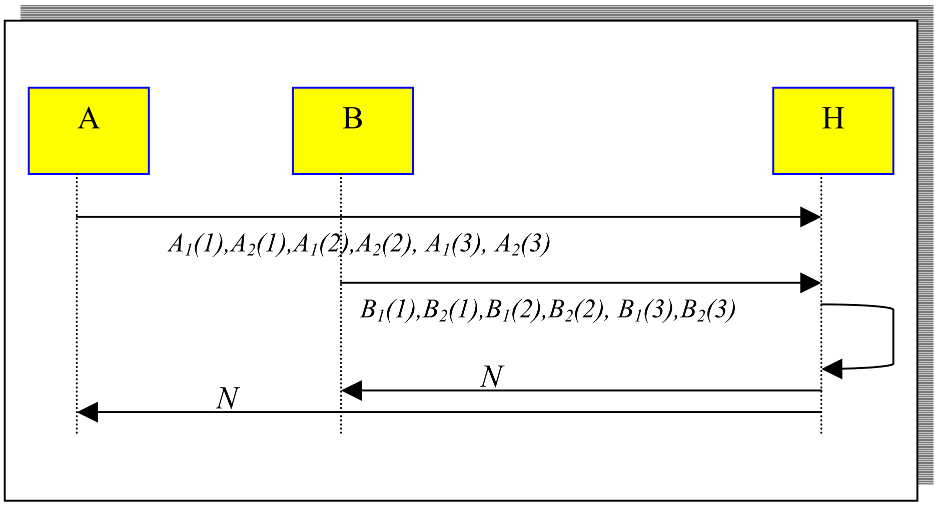

A
B
H
$A_1(1),A_2(1),A_1(2),A_2(2), A_1(3), A_2(3)$
$B_1(1),B_2(1),B_1(2),B_2(2), B_1(3),B_2(3)$
N
N

Figure 2.5 Distributed of Computation N

The values p_1, q_1, p_1 and q_1 is never revealed to the other parties. Even tough *H* receives the function values from *A* and *B*; *H* doesn't know the values of p_i and q_i.

2. 2. 3. 2 Distributed Primality Test

Since *N* is product of the two primes $p=p_1+p_2$ and $q=q_1+q_2$, but each party knows only p_1, q_1 or p_2, q_2 and doesn't know the total values *p* and *q*, so we need to test that (p_1+p_2) and (q_1+q_2) are primes. On the other hand, we test that *N* is the product of the two prime numbers, but in this test each party doesn't reveal any secret values. The test is based on the basic number theory and known as Fermat test. If N is product of two primes p and q, then

$$g^{(p-1)(q-1)} = 1 \bmod N \quad where \ \ gcd(g,N) = 1$$

Since $p = (p_1 + p_2)$ and $q =(q_1 + q_2),$ then

$$g^{(p1+p2-1)(q1+q2-1)} = g^{((p1+q2)(q1+q2)-p1-p2-q1-q2+1)} = g^{(N - q1-p1-p2 -q2+1)}$$

The algorithm works as follows:

1. A picks a random value g where $gcd(g,N) = 1$. A then computes $V_a = g^{(N-p1-q1+1)}$. He sends g to B and V_a to H.
2. B computes $V_b = g^{(-p2-q2)}$ and sends V_b to H.
3. H checks whether $V_a\, V_b = 1 \bmod N$. If so, then N is accepted as the product of two primes. Otherwise, the protocol backs to determine the new p_i and q_i.

Figure 2.6 shows the Distributed Primality Test in the sequence diagram.

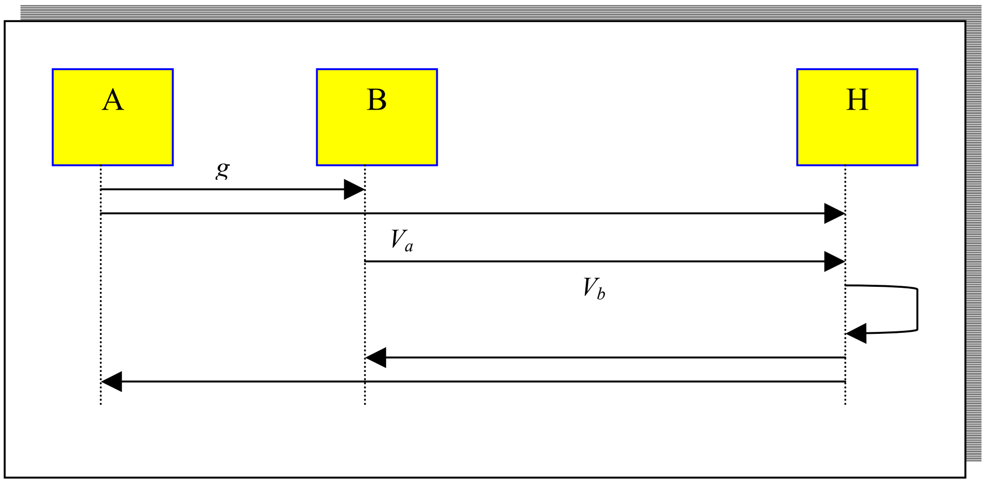


Figure 2.6 Distributed Primality Test

The secret values are not revealed. Even tough H receives V_a and V_b; H doesn't know the values of p_i and q_i, because he doesn't know g.

2. 2. 3. 3 Distributed Key Generation

As the conservative RSA, after succeeding to compute N, we finally compute (shares) d. First we choose the encryption key (e) randomly that are relatively prime with $\varphi(N)=(p-1)(q-1)$. To compute shares of d the parties must invert e modulo $\varphi(N)$ where $\varphi(N) = \varphi = (p-1)(q-1)$. We compute the inverse of e using following steps:

1. Computes $\varsigma = \varphi^{-1} \bmod e$
2. Set $T = -\varsigma\, \varphi + 1$
3. Set $d = T/e$, we can verify that $d = e^{-1} (mod\ \varphi)$ since $d.e = T \equiv 1 \bmod \varphi$

Since $\varphi = \varphi_1 + \varphi_2$, where $\varphi_1 = N - p_1 - q_1 + 1$ and $\varphi_2 = - p_2 - q_2$, A and B computes $d=d_1+d_2$ as follows:

1. Either *A* or *B* picks a random value *e* and send to the other party.
2. A sends φ_1 and *B* sends φ_2 to *H*.
3. *H* computes ς, and sends back the value to *A* and *B*.
4. *A* and *B* separately compute

   $d_1 =(-\varsigma\ \varphi_1 + 1) / e$
   $d_2 =(-\varsigma\ \varphi_2) / e$

   They hold the values as the private key.

The diagram of the Distributed Key Generation can be seen in the Figure 2.7.

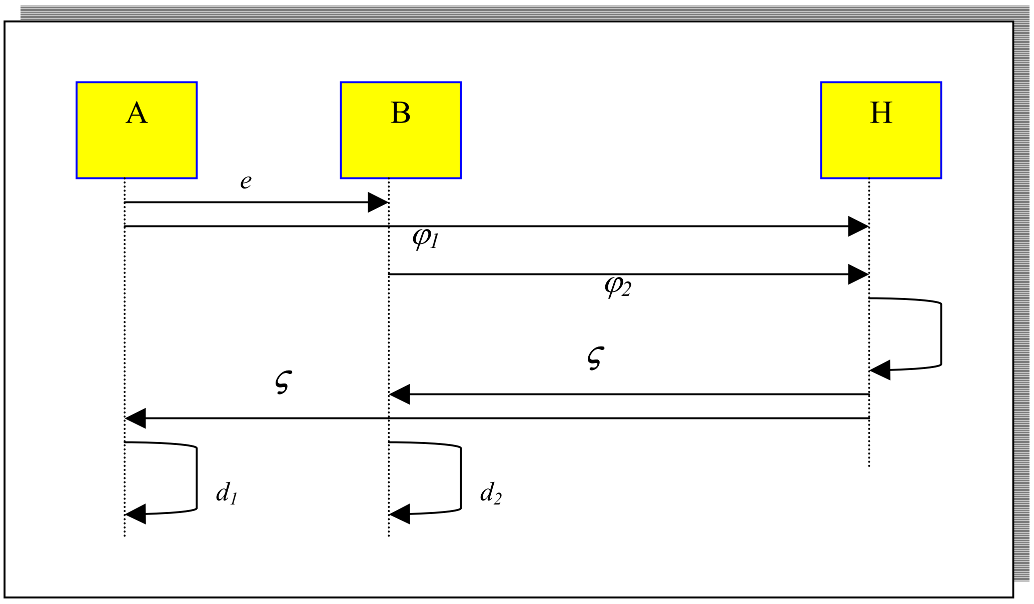


Figure 2.7 Distributed Key Generation

Once again, the secret values are never revealed. Even though *H* receives φ_1 and φ_2; *H* may know only $(p_i + q_i)$, but doesn't know the values of p_i and q_i.

Chapter 3 Cryptography In Java

In this section, we discuss about the techniques aspects about cryptography in Java Programming. We think that it is important since we use Java for the implementation in this project. In Java, The Java Cryptography Architecture (JCA) and The Java Cryptography Extension (JCE) were designed to provide an implementation-independent API for cryptographic function. The JCA is part of the Java 2 run-time environment, while the JCE is an extension to the JCA.

The JCA is packaged in `java.security` package. The Java Security in JDK contains a subset of cryptography functionality, including APIs for *digital signatures* and *message digests*. In addition, there are abstract interfaces for *key* management and *certificate* management.

On the other hand, the JDK does not include the JCE. We have to install before we can use the JCE. The JDK ships with only certain cryptographic functions enabled, namely those defined by the JCA. The JCE contains others subset of cryptography functionality, such as: *Cipher*, *Key Agreement, Key Generator*, and *Secret Key*. After installation, the JCE stays in the *javax.crypto* package and its sub-packages.

We use both the JCA and the JCE for this project. Since the JCA is a part of the Java, we do not need to download the package separately. But we have to download and install self the JCE. We choose the JCE from Bouncy Castle for this project.

3. 1 Java Cryptography Architecture (JCA)

The Java Cryptography Architecture (JCA) is composed of a number of classes in the *java.security* package and its sub-packages. These classes provide generic APIs for functions like digital signatures and message digests. The important classes include:

- Message Digest
- Signature
- Key Pair Generator
- Key Factory
- Certificate Factory
- Key Store
- Algorithm Parameters
- Algorithm Parameter Generator
- Secure Random

3. 2 Java Cryptography Extension (JCE)

The JCE is very similar in design to the JCA. The JCE is classes in the *javax.crypto* package and its sub-packages. It includes the following classes and interfaces, among others:

- Cipher
- Key Agreement
- Key Generator
- Mac
- Secret Key
- Secret Key Factory

3. 2. 1 Installing JCE

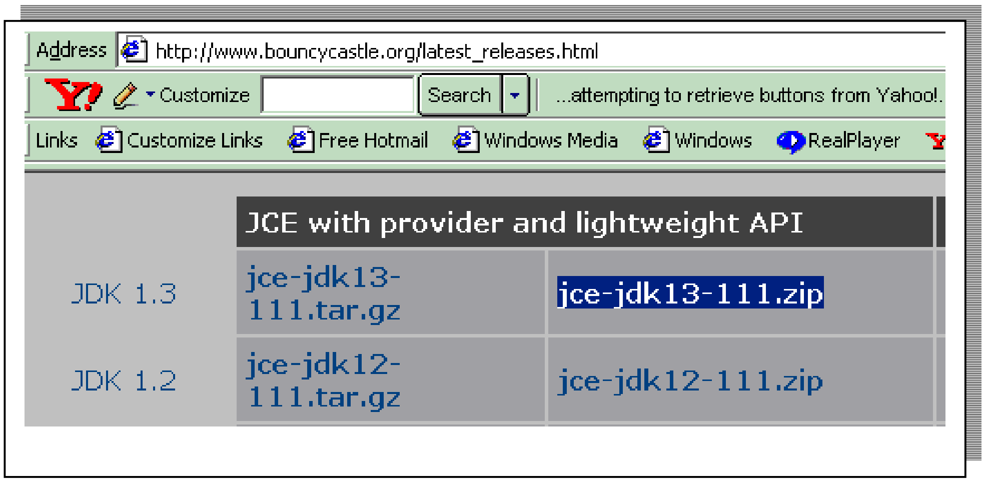
We need to install the JCE first before we use it. Garms and Somerfield have described how to install the JCE in their book [13]. There are any of a number of possible implementations of the JCE, many of which come with a provider containing implementations of various algorithms. We use the JCE from *Bouncy Castle* that is recommended by Garms [13].

We do the following steps in order to install the JCE:

1. Download the provider
2. Copy the JAR files into an appropriate location
3. Configure the java.security file

3. 2. 1. 1 Downloading the provider

We download the provider from *Bouncy Castle* that is the most complete and freely available JCE provider. Bouncy Castle can be downloaded from www.bouncycastle.org. We pick up the *JCE with Provider and Lightweight API*, jce-jdk13-111.zip.



http://www.bouncycastle.org/latest_releases.html

	JCE with provider and lightweight API	
JDK 1.3	jce-jdk13-111.tar.gz	jce-jdk13-111.zip
JDK 1.2	jce-jdk12-111.tar.gz	jce-jdk12-111.zip

Figure 3.1 Bouncy Castle Home Page

After downloading the file (jce-jdk13-111.zip), we extract it into a special place.

3. 2. 1. 2 Copying the JAR files

The bouncy Castle provider doesn't come with JAR files, but they easily can be built from the *jce-jdk13-111.zip* file that has been downloaded. We extract the file, CD into the directory classes, and make the JAR file from the necessary files. To make the JAR file, we write the following command:

```
C:\ jar cvf bouncycastle.jar javax org
```

A new jar file called *bouncycastle.jar* is created. In order to have the JCE classes available to all Java applications, including tools like javac and key-tool, we install the JAR files as extensions. The extensions are placed inside the Java Run-time Environment's *lib\ext* directories.

In Windows, Java is typically installed into two locations. One is for development and includes all the tools of the JDK. It is usually in *C:\jdk1.3* or the like. The other is merely a run-time environment, and typically in *C:\Program Files\JavaSoft\JRE\1.3*. Each one has a *lib\ext* directory. In our environment, we use Jbuilder5. The JDK itself is in *D:\ jbuilder5\jdk1.3,* and the run-time environment is in *C:\Program Files\JavaSoft\JRE\1.3*. We need to copy the *bouncycastle.jar* into the *lib\ext*, which is in:

D:\ jbuilder5\jdk1.3\jre\lib\ext

and

C:\Program Files\JavaSoft\JRE\1.3\lib\ext

3. 2. 1. 3 Configuring the java.security file

Parallel to the *lib\ext* subdirectory is a subdirectory named *lib\security*. There is a file *java.security* in that subdirectory. The *java.security* file defines what cryptographic providers are enabled. If we open it (for example with notepad), we will see one or more lines like the following:

```
#
# List of providers and their preference orders (see above):
#
security.provider.1=sun.security.provider.Sun
security.provider.2=com.sun.rsajca.Provider
```

It indicates to the VM that there are two cryptographic service providers. It defines their precedence, and class names should be used to access. When asked for an implementation of a cryptographic algorithm, the VM will query each of the listed providers in order, looking for such an implementation. It will use the first one it finds, based on the precedence.

We edit the list to add our new provider by adding a line like the following:

```
security.provider.3=org.bouncycastle.jce.provider.BouncyCastleProvider
```

Finally, here is the full list of providers for our VM, which uses the built-in JDK providers and Bouncy Castle:

```
#
# List of providers and their preference orders (see above):
#
security.provider.1=sun.security.provider.Sun
security.provider.2=com.sun.rsajca.Provider
security.provider.3=org.bouncycastle.jce.provider.BouncyCastleProvider
```

We update the file, and on our system it is in:

D:\ jbuilder5\jdk1.3\lib\security\ java.security

And

C:\Program Files\JavaSoft\JRE\1.3\lib\ security\java.security

Since we use Jbuilder (5) for the program, we have to specify the JCE in the Jbuilder. It is done from *Project Properties* menu, *Path*, and *Required libraries*. Click *Add…* and *New…*, Write JCE in the text field, and click *Add…* Specify the subdirectory:

D:\ jbuilder5\jdk1.3\jre\lib\ext\bouncycastle.jar

3. 2. 2 Testing the installation of the JCE

After finishing the installation, we have to check whether the installation has succeeded. To check it, we try running a simple program that attempt to use some of the classes in the JCE. The test attempts to generate a secret key for the DES algorithm, using *javax.crypto* classes. Figure 3.2 is list of simple test program shows that the installation works well.

```
package des01;
import java.security.*;
import javax.crypto.*;

public class DESclass {

  static String stringToEncrypt = "Hallo Sayang";

 public static void main(String[] args) throws Exception{
    String text ="Hallo Sayang";
    System.out.println("Testing JCE install.... ");
    KeyGenerator keygenerator = KeyGenerator.getInstance("DESede");
    SecretKey secretkey = keygenerator.generateKey();
    System.out.println("OK ...");

    System.out.println("Attempting to get a Cipher and encrypt");
    Cipher cipher = Cipher.getInstance("DESede/ECB/PKCS5Padding");
    cipher.init(Cipher.ENCRYPT_MODE, secretkey);
    byte[] ciphertext = cipher.doFinal(stringToEncrypt.getBytes("UTF8"));
    System.out.println("OK ...");

    System.out.println("Test completed succesfully ...");
  }
}
```

Figure 3.2 Test Program

After compiling and running the program, the output should be seen as the following:

```
Testing JCE install....
OK ...
Attempting to get a Cipher and encrypt
OK ...
Test completed succesfully ...
```

Chapter 4 System Scenario

4. 1 Project Description

The project is to implement an application when we have a database that is saved in third place. If not all person is permitted to get the data we have to protect it. Only one having permission can use the data.

One classical way to protect data is that we encrypt the data with a RSA private key, and then we can decrypt it with its coupled public key. We then distribute the public key to one who is permitted to use the data. In this way, the process of public key distribution becomes central issues, besides the protection of the key and the data self. We know some topics about Digital Signature, Digital Certificate, and so on, that now are the ways for the distribution of key.

In this project, we develop a system that protects the data in third place without key distribution. We encrypt data with RSA public key, and save the data into a third place called Administrator, which will manage it. The data can be decrypted with its coupled private key. We as owner hold both the public key and the private key, and we decrypt self the data for user who has permission.

The system works as follow. A person called User wants to get data we have. Data stores in a third place called Administrator. The user first sends a query to the administrator. To protect the query from eavesdropping, the query is hashed before sent. After the administrator has received the hashed query, it looks into its hash table to match if it matches with one row in the table. If the query is match with the table, then the administrator sends the data related to the query to the user.

But note that the data is still encrypted. The user cannot use the data until it is decrypted. Because the user doesn’t have the private key, s/he has to ask decryption to the data owner. The user then sends the encrypted data to the user for asking decryption. Suppose that the user has permission to use the data. The owner decrypts the data and then sends back to the user. Here, communication between the user and the owner has to be secure, especially when the owner sends back the data to the user. In this step, the data has been decrypted. Therefore they must set once more encryption to the data for the communication. Figure 4.1 below is the Use Case diagram that shows the activity of the system.

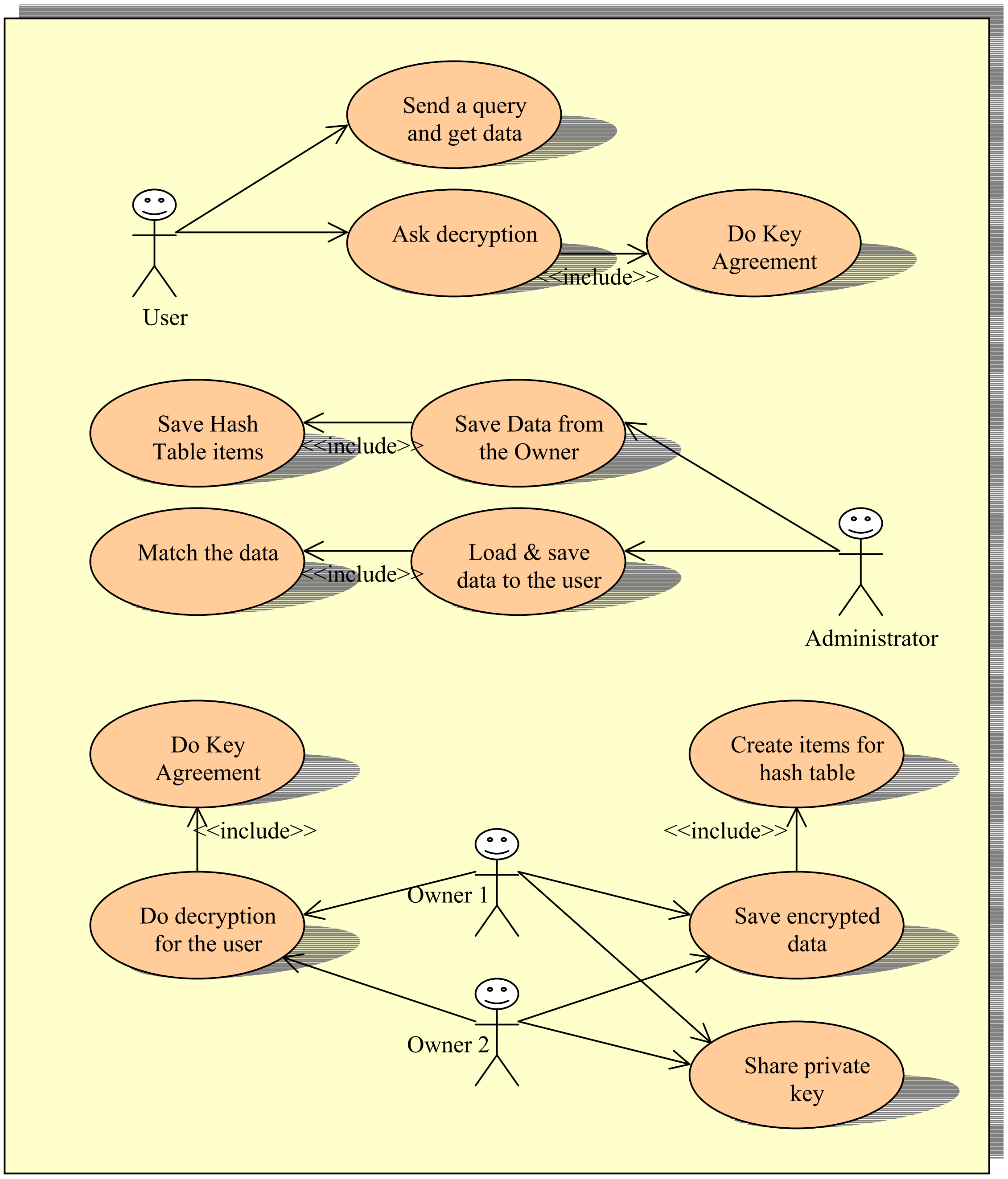


Figure 4.1 Use Case Diagram of System

4. 2 System Components

The system consists of three components, that are

1. **User**, who is a person that wants to get the text.
2. **Owner**. The owner is a person that owns the data. It can be more than one person. In case more than one owner, they save the public key and share the private key in their computers. If a user wants to read data, the owner helps to decrypt the text without revealing the private key.

3. **Administrator**. Administrator is a place (server) where the data are saved. The owner produces data. The data is written and saved in a third place. One who wants to read the data must have permission to the owner, and therefore it is saved with encryption. Every one probably can download data, but no one can read it without the owner's permission.

Figure 4.2 shows the system components. One more component is the database system that is not showed in the figure. It can be in the administrator or in another place, but there is administrator that controls the database system. Therefore we do not show the database component in the figure.

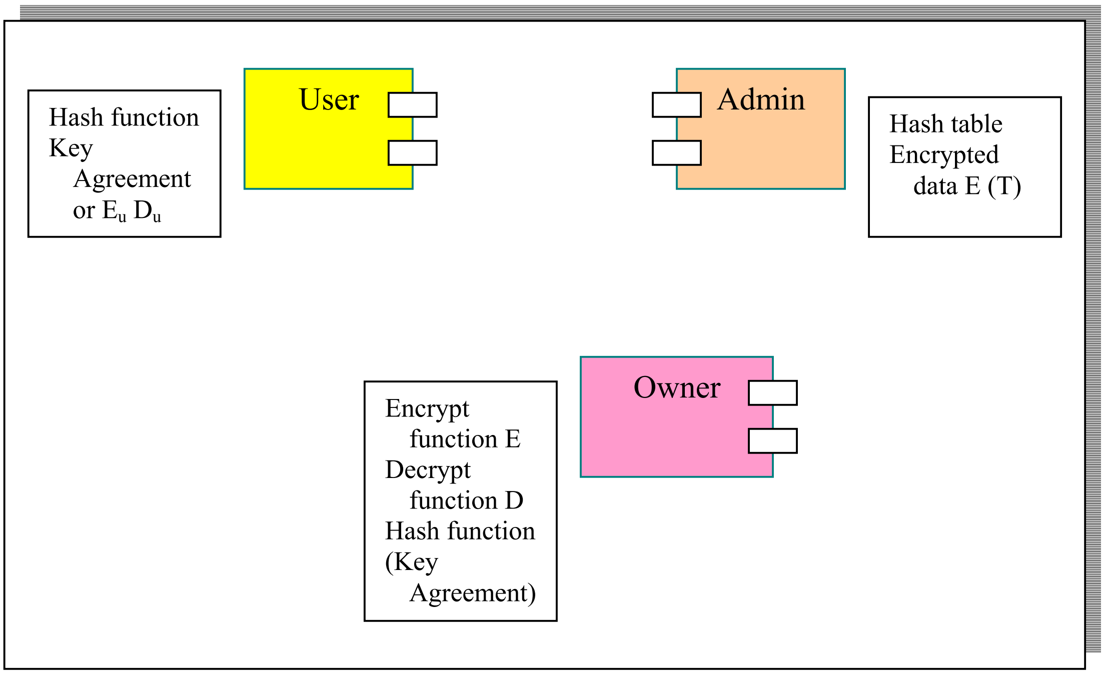


Figure 4.2 System components

The user has Hash function. The hash function is for creating a query. To ask decryrption to the owner, the user can use either key agreement or user's RSA key. The owner has Encrypt function E, Decrypt function D, Hash function. E is used for encrypting the data, D is for decrypting the data from the user, and the hash function is for hashing the items of the data. When the owner will save data, besides encrypting the data, he creates hash of the data to locate in the hash table in the administrator. The hash table will point out which data that related with the encrypted data. The administrator has Hash table and Encrypted data E (T). The administrator has responsibility to manage all data; matches query from a user with his table, and send the data to the user.

We implement two cases in this project. These are system with one owner and system with two owners. In the one owner, the system works as we have described. In the two owners, the private key has to be shared between the owners. We implement two share keys that we have discussed in the last section.

4. 3 Case 1: System with one owner

The system is developed to secure data in the third place. After creating a new secure data (*T*), the owner saves it into the database via administrator. But before putting *T* to the administrator, the owner makes crypto transformation of *T* in order to protect the data and still be able to perform pattern matching. Algorithms and protocol proposed by Oleshchuk are used in the system [11]. Interaction between the parties (the owner, user and administrator) is done in the following process:

1. Process of encryption and saving done by the owner. The owner must do some works to do the process. This is the interaction only between the owner and the administrator, and doesn't have relationship with the user. We describe the process in section 4.3.1.
2. The user sends a query to get information. Actually, the process of getting information begins from this step. Section 4.3.2 describes about the process.
3. The administrator does the matching process based on the query from the user. Section 4.3.3 describes about this step. After matching the administrator sends set of encrypted blocks to the user.
4. The user decrypts the blocks in cooperation with the owner. It is described in the section 4.3.4.

4. 3. 1 Saving Data by the Owner

There is the owner that creates the data. He must do some works in order to protect the data. The owner generates a RSA public key and its private key. The owner then encrypts data with the public key. Besides encrypting the data, the owner hashes all partitions of the data to let the user match the information about the data. Both the encrypted data and the hash data are sent to the administrator. However the public key is not revealed, even to the permitted user. The owner holds both the public and private keys.

Source Data Pre-processing Algorithm from Oleshchuk [11] is used in this process. But note that we pass the encryption for the hash table from the algorithm. Section 2 has described about the reason. Figure 4.3 shows the saving data by the owner.

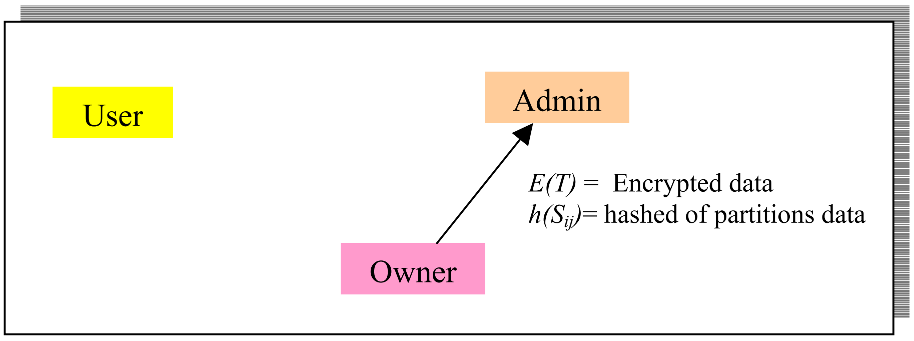


Figure 4.3 Saving Data

4. 3. 2 Getting Data by the User

When a user wants to get the data, he sends a query first to the administrator. But the data is encrypted in the administrator. The user must do some works before sending the query. The user first split the query into partitions. The user then hashes the all partitions of the query. Based on the pattern pre-processing algorithm presented by Oleschuk[2], this is the process done by the user:

1. The user creates a query.
2. The query is broken into partitions; the user combines all possibility of the query.
3. The all partitions then are hashed.
4. The user sends the all encrypted hashed query. We call the hashed queries that are sent as patterns.

Since we don't use encryption for the hash table, we have to pass also the encryption of the query from the algorithm presented by Oleshchuk in this process, such that the patterns can be matched with the hash table in the administrator. Figure 4.4 shows process of getting data by the user.

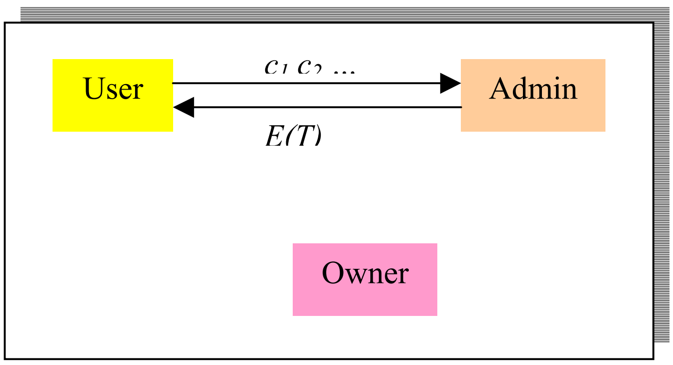


Figure 4.4 Getting Data by the User

4. 3. 3 Matching Process by the Administrator

All data and any other information are saved in the administrator. The administrator has responsibility to send the data to the user. Before sending the data, he must check which data that is needed by the user. So he matches the query from the user with his table. Oleschuk has presented matching algorithm for this goal [2]. Based on the algorithm, after receiving the patterns from the user, the administrator does the following steps:

1. The administrator first matches each of the patterns to the hash table.
2. If it matches, then the block number is listed. Therefore some block number will be listed based on the pattern matching.
3. The administrator then sends back the blocks to the user based on the lists.

4. 3. 4 Process of getting data

The data that is sent by the administrator to the user is encrypted data; the user must ask the owner to decrypt it. The user then sends this (encrypted) data to the owner. To protect the data from eavesdropping we implement an encryption with a RSA public key (E_u) by the user before sending, and decryption process after the user get the decrypted data.

We investigate also to use a Key Agreement between the user and the owner. After the key agreement is done, both the user and the owner send the data with the same secret key, not with the RSA key. It is proved that using secret key is much faster than using public/private key [12, 15].

After receiving the encrypted data from the user, suppose that the user has permission, the owner then decrypts the data, and sends back to the user. The user then can access the data.

So, in this system all users can download data from administrator, but it cannot be used until be decrypted by the owner. The owner will only decrypt data for users that have permission. Figure 4.5 shows the communication between the user and the owner in the decryption process.

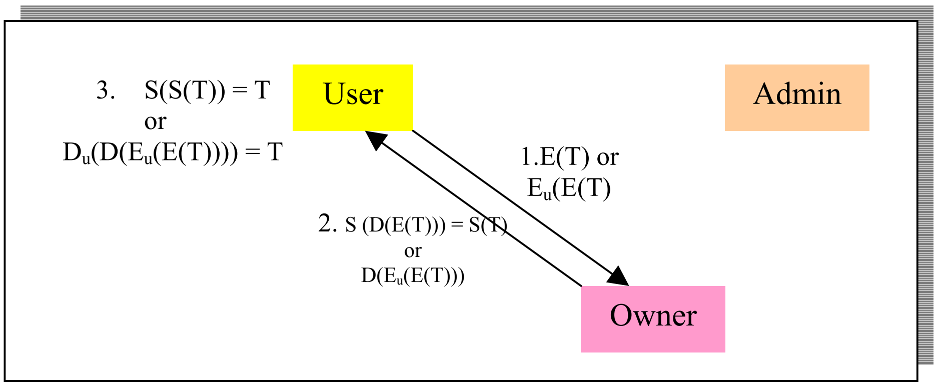

3. $S(S(T)) = T$ or $D_u(D(E_u(E(T)))) = T$
User
Admin
1.E(T) or $E_u(E(T)$
2. $S(D(E(T))) = S(T)$ or $D(E_u(E(T)))$
Owner

Figure 4.5 Decryption Process

4. 4 Case 2: System with more than one owner

Further scenario is that if there is more than one owner. This is similar to system with one owner. The difference is that to read the data, the user must have permission from the two owners. Figure 4.6 shows about the system.

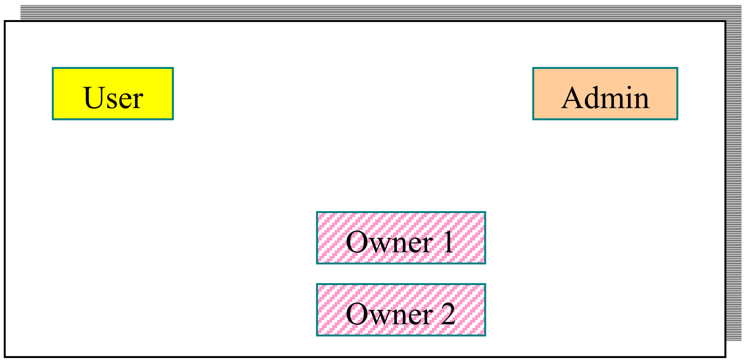


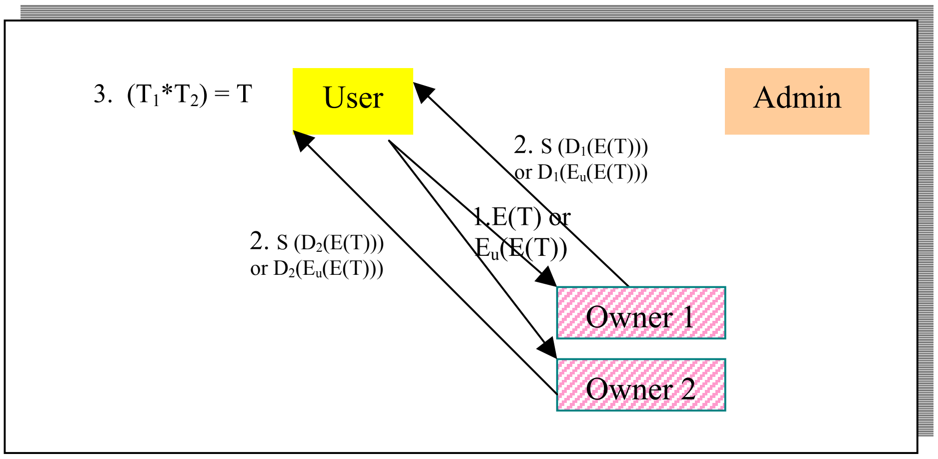
Figure 4.6 System with two owners

We developed an application presented by Boneh [6]. Wu, Malkin and Boneh have presented an intrusion tolerance system that includes two or more servers that share the private keys. The system protects an application's private key by sharing the key among a number of share servers. An attacker who breaks into a small number of share servers cannot expose the private key.

In our system, the private key is shared among a number of share servers that are the owners' servers. Owner 1 holds the private key d_1 and Owner 2 holds d_2. The user that wants to read the data must send the (encrypted) data to the two owners. Owner 1 decrypts the data with his shared key d_1 and Owner 2 decrypts the data with d_2. They then send back to the user. The user can read the data after he multiplies the datas from the owners. Figure 4.7 shows the process of decryption by the two owners.

The solution to read the data becomes

$D = D_1{*}D_2$ because $c^{d1+d2} = c^{d1} * c^{d2}$.



3. $(T_1{*}T_2) = T$
User
Admin
2. S $(D_1(E(T)))$ or $D_1(E_u(E(T)))$
1.E(T) or $E_u(E(T))$
2. S $(D_2(E(T)))$ or $D_2(E_u(E(T)))$
Owner 1
Owner 2

Figure 4.7 Data Decryption of the two owners

Chapter 5 Implementation

As we have discussed above, the system consists of three components; User, Owner, and Administrator, plus Database that has closed connection with the administrator. This is three tiers architecture, where User is web server, Administrator is server tier, and Database is as database tier. However, we add Owner that is available between User and Administrator. The owner is as server for the user because the owner serves the user to decrypt data, and is client for the administrator to save and load data. Figure 5.1 shows the system in this project.

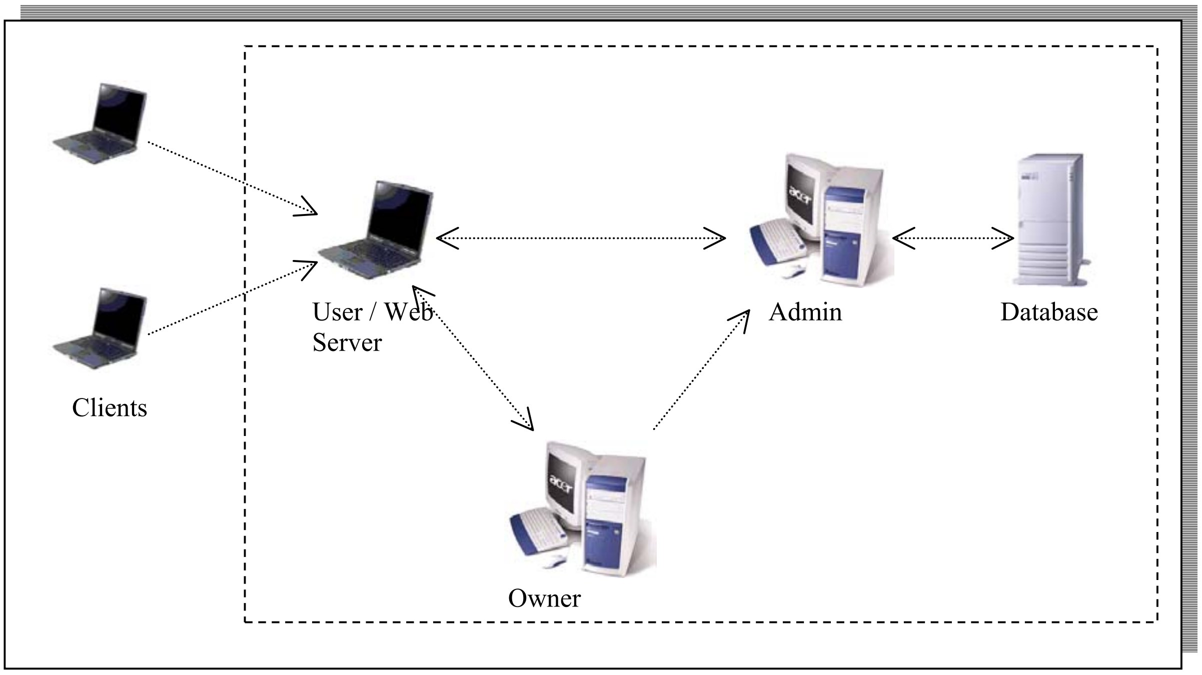


Figure 5.1 System Implementation

Communication between clients and User is out of the project. The communication can be secure using SSL. SSL, or Secure Socket Layer, is a technology which allows web browsers and web servers to communicate over a secured connection. The data being sent is encrypted by one side, transmitted, and then decrypted by the other side before processing. This is a two-way process, meaning that both the server AND the browser encrypt all traffic before sending out data. Another important aspect of the SSL protocol is Authentication. During the initial attempt to communicate with a web server over a secure connection, that server will present the web browser with a set of credentials, in the form of a "Certificate", as proof the site is who and what it claims to be. In certain cases, the server may also request a Certificate from your web browser, asking for proof that *the client* is who the really client claim to be. This is known as "Client Authentication," although in practice this is used more for business-to-business (B2B) transactions than with individual users. Most SSL-enabled web servers do not request Client Authentication.

Many web server vendors provide the facility of SSL. Tomcat, which is used in this project, has also the SSL for making protection to the clients. We can follow the manual instruction to setup this facility. So, we concentrate on the protection between User, Owner, and Administrator.

In this project we implement an application where a user **searches** information in the database. Suppose that a client wants to search information about article in the database. The client writes request about article via web server (HTML pages). In this project, the server hashes it and sends to the administrator to protect the query such we have described above. After that the process is the same as the section before. **The result** is that the user can read information via Web server.

Each system components will be described in the next section. We use Java Programming for the application. In our experience, Java is good for network programming. It can be access via web server with Servlets or JSP. We found also that Java supports security and cryptography.

The implementation is tested in one computer. We use three different ports for each component, such that it behove like different stations. If the system is implemented in the three different stations, it can work well.

5. 1 Java Socket Programming

We use Socket from java to support communications between all of the components. Socket let each components receive and send messages over **TCP/IP network**. Java Socket is part of the Java 2 run-time environment, and packaged in `java.net` package.There are three main socket classes in Java; ServerSocket, Socket, and DatagramSocket. The two former are used for connection-oriented data, so that they support TCP protocol. The Datagramsocket class does connection-less networking, UDP protocol. Since data in this project is encrypted and need to be complete for decryption, we use only ServerSocket and Socket. The Socket class is used to implement both clients and servers, while the ServerSocket is used to implement servers.

Socket programming provides a low level mechanism by which we can connect two stations for the exchange of data. One of those is considered to be the client and the other is considered to be the server. The server uses the ServerSocket class and waits connection from client. The client initiates connection with the Socket class.

To apply a station as a server, it has to decide which port will be use, and instantiates a ServerSocket object tied to this port. Then it will enter an infinite loop and invoke *accept* () method on that object. The number of port will be monitored by the ServerSocket when *accept* () method is invoked. The *accept ()* method will block consuming few if any resources, until a connection request is received from a client. When a connection request is received, the communication between the server and the client can be done using the port that is used by the server. Figure 5.2 shows a little listing program of the ServerSocket.

```
try {
        ServerSocket serversocket = new ServerSocket(port);
        while (true) {
                serversocket.acept();
                :
                :
                serversocket.close();
        }
}
```

Figure 5.2 ServerSocket

A client initiates the server has to specify the server address (IP address), and the port on the server that is connecting to. Then it establishes a connection with the specified port on the specified server by instantiating a new object of type Socket. Once the object exists, it is possible to use it to communicate with the server on the specified port using the protocol prescribed for the service being delivered on that port, so transferring data between the client and the server is allowed. Figure 5.3 is the presume listing of Socket, where a client initiates a connection with a specific server. Note that the port must be the same with the server port.

```
try {
        Socket socket = new Socket(serveraddress, port);
                :
                :
        socket.close();
}
```

Figure 5.3 Client Socket

5. 2 User

5. 2. 1 Apache Tomcat on the User

A client connects to the user from everywhere in the world. We use a Web Server for the user, and we use the **Servlets technology** from Java for the applications. Servlets are Java programs based on Common Gateway Interface (CGI) that run on a Web Server, acting as a middle layer between a request coming from a Web browser or other HTTP client and databases or applications on the HTTP server.

We choose **Apache Tomcat** v 4.0 as the server. Tomcat is the official reference implementation of the Java Servlets (and Java Server Pages, JSP) specifications. The 4.0 release implements the Servlet 2.3 and JSP 1.2 specifications. Tomcat v 4.0 are available at:

Binary: http://jakarta.apache.org/builds/jakarta-tomcat-4.0/nightly/
Source: http://jakarta.apache.org/builds/jakarta-tomcat-4.0/nightly/src/

Tomcat can be downloaded for free, and used as a stand-alone Web server, or used as an extension to other Web server (such as the Apache Web server). By default, Tomcat uses port **8080.** The client can browse the server with **HTML pages**, using Internet Explorer, Netscape Communicator, or other Internet browsers. After the client has come to the server, the servlets implement communication between the user and the administrator, or/and between the user and the owner.

Tomcat is used by the user as Web server to clients. From the user to the administrator and the owner, we use Java Socket Programming, where the user is act as client that initiates connection to the administrator or the owner as servers. The user has two works to get data from the administrator and the owner. These are

- Sending query to the administrator and receives encrypted data
- Asking decryption to the owner

5. 2. 2 The user Sends query and receives encrypted data

The previous section has discussed about the mechanism of the processes. To send a query, in this implementation, if the query has some words, it is broken into words. Each word then is hashed using a hash algorithm. We use **MD5 algorithm** for hashing in this implementation. So, we implement various sizes of blocks to be hashed depending on the size of the each word. For example, the query is *Yusuf Kurnia*. It will be splitt into two words; *Yusuf* and *Kurnia*. Each *Yusuf* and *Kurnia* then is hashed, becomes *h(Yusuf)* and *h(Kurnia).* This is different from the paper blocks presented by Oleshchuk [11], which is uses fixed size. We have discussed details in the Pattern Pre-processing section previously. The hashed query then is sent to the administrator. There is the administrator that will match the query in its hash table, and send back the matched data to the user. Figure 5.4 shows the flowchart diagram of the process sending query and getting data to/from the administrator.

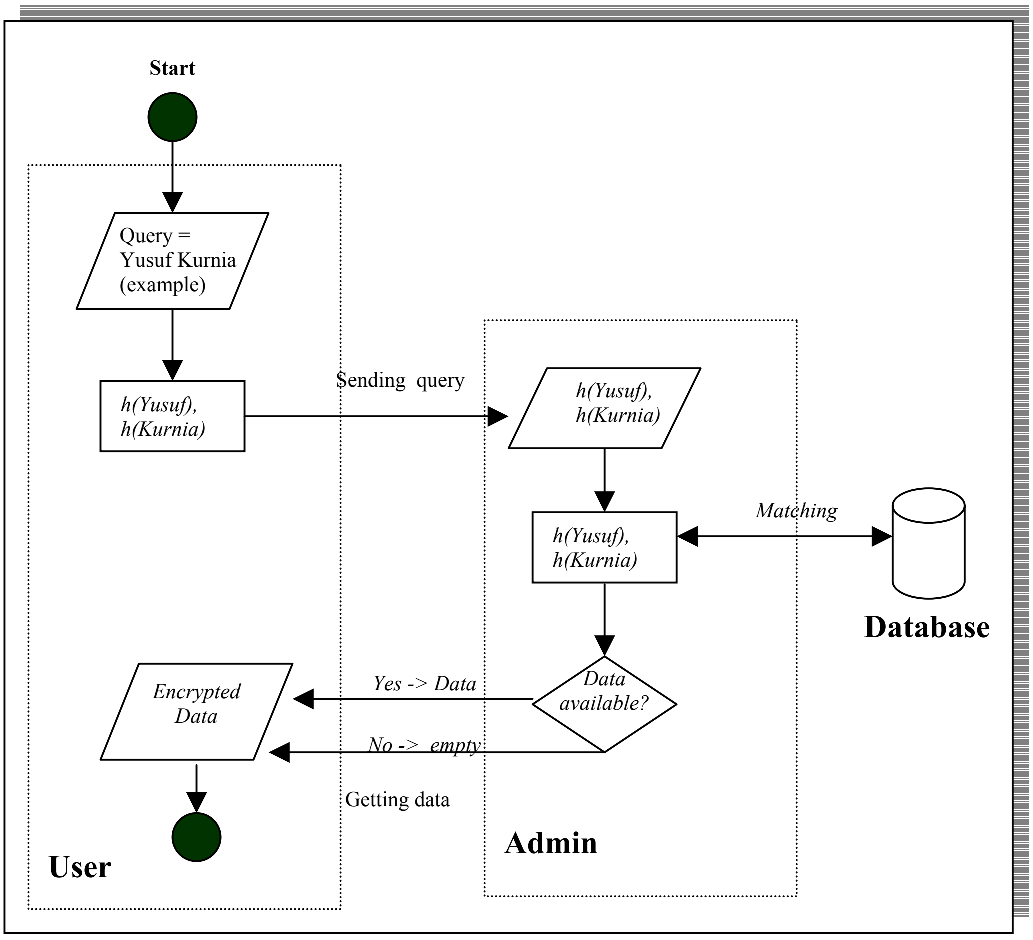


Figure 5.4 Flowchart diagrams of sending query and getting data

5. 2. 2 Key Agreement

To ask decryption, the user must send the encrypted data to the owner, because the owner does not distribute the decryption key. The user sending the encrypted data, the owner decrypts the data, and sends back the decrypted data, such that the user can use the data. Note that the data from the owner is now unencrypted, so that all can use that. This is insecure; we must protect it from un-permitted users. We apply a key agreement between two parties, the user and the owner, before the data has been sent. After doing the agreement, each part has the same integer value. Based on the value, both of them will generate the same secret key.

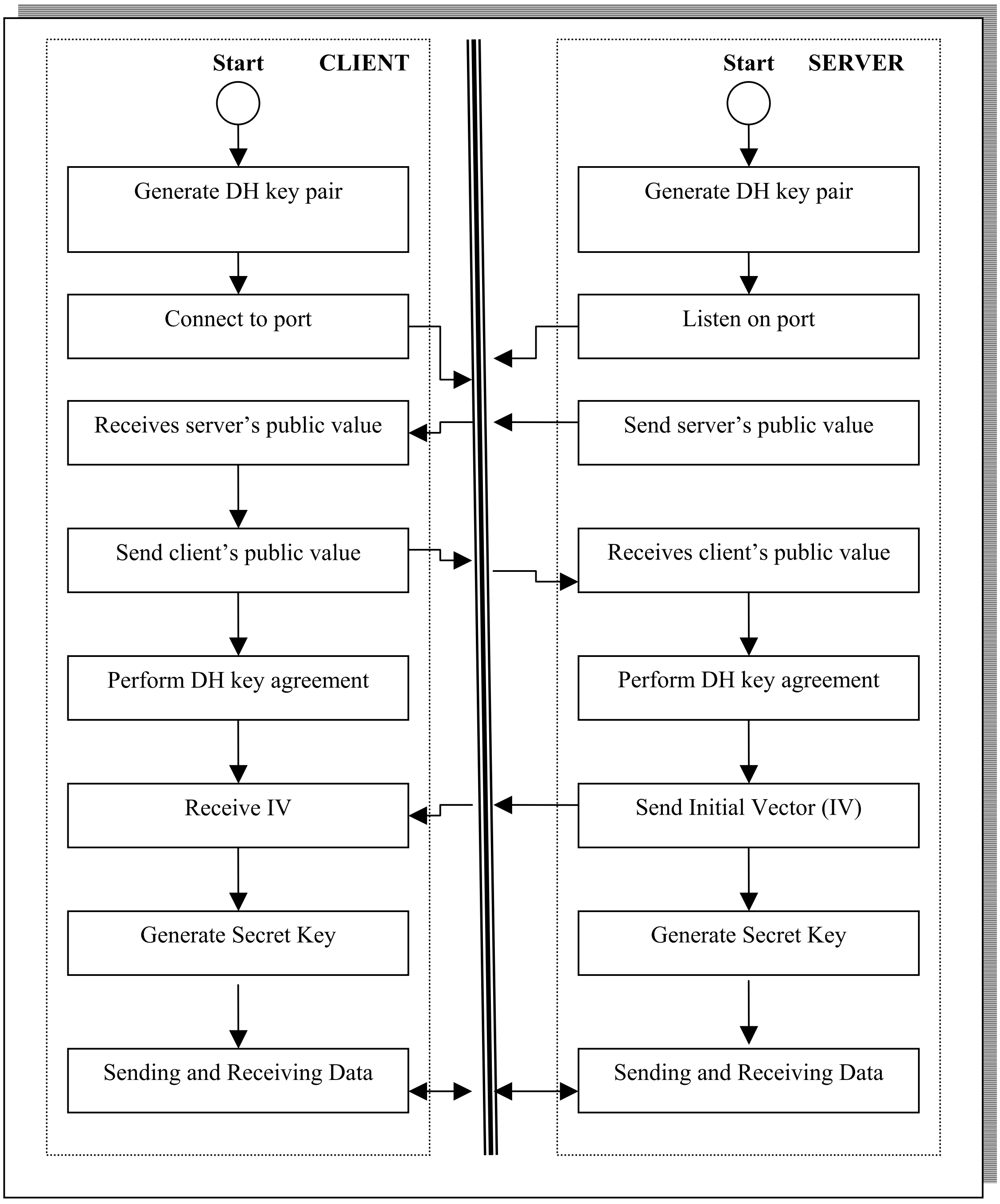


Figure 5.5 Key Agreement between the user and the owner

Finally, they can send and receive the data after the data has been encrypted with the key. On the opposite side, the data will be decrypted with the same key. So, the decrypted data from the owner now is protected because the owner encrypts the data with the secret key. The key agreement and decryption processes are shown in Figure 5.5. We use **Diffie-Helman** algorithm to do the key agreement in this project. The client is the user and the owner acts as server.

Beside the key agreement method, we also implement using RSA key by the user, which is presented by Oleshchuk [11]. On the paper, the encrypted data must be encrypted again with a

RSA encryption key. After the owner has decrypted the data, it just sends back to the user, because the data is still encrypted with the RSA key by the user. After the data has come to the user, the user will decrypt it with his private key.

5. 3 Administrator

5. 3. 1 Java Socket on the Administrator

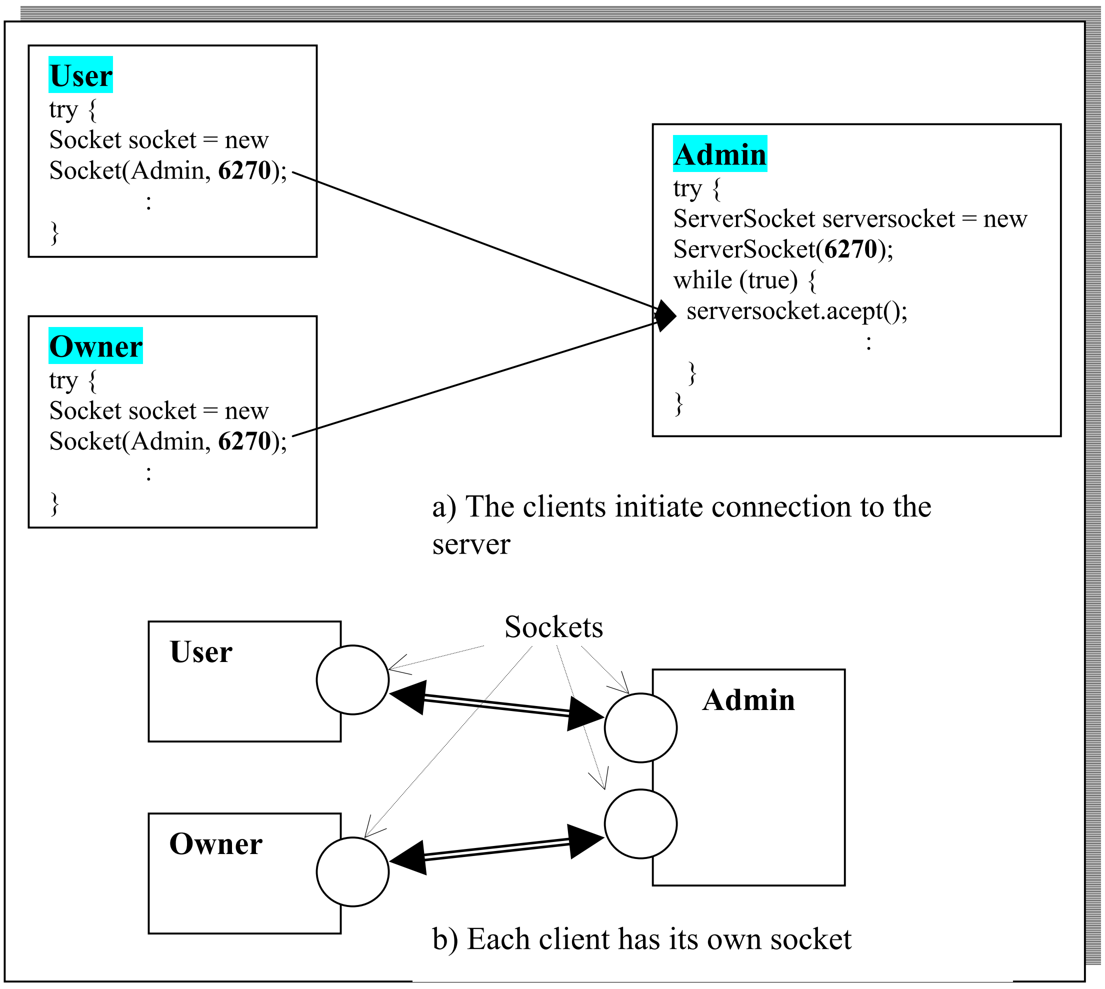
The administrator is the **Java application** that uses Java Socket Programming. We build the administration server with Socket on the port **6270** in this project. The administrator always acts as server that has tasks as follows:

- Implement request from the user
- Implement request from the owner
- Save and load the data to/from the database

The implementation from the user is that the user as a client wants to get the data from the administrator. When the user request data, the administrator matches it and sends the encrypted data from the database to the user. The data sending by the user is secure because it has been hashed, whereas the data sending by the administrator to the user is secure too, because it is encrypted data. We have discussed about the user in the previous section.

The implementation from the owner happens when the owner either saves data or loads the data. First, the owner as a client initiates connection to the administrator, and sending information what will the owner do. If the owner will save data, the administrator only receives data sending by the owner, and saves it into the database. The administrator has responsibility to manage the database. When the owner wants to checks data, the administrator sends the data that the owner will. Note that the data is encrypted, the administrator cannot read/analyse it because there is only the owner that can decrypted the data.

Note that although the user and the owner as clients of the administrator, and they use the same port to the administrator, between the user and the owner cannot communicate each other through the administrator. When a client establishes a connection with a server, a new object of type Socket is created. The object includes a channel that is used for communication between the client and the server. In this case, when the user connects to the administrator, it will create a socket object. When the owner connects to the administrator, it will also create an object of type socket. But the object is another object. So, the socket object from the user and the socket object from the owner are different. Therefore, the user and the owner cannot communicate through the administrator. They can communicate with the administrator without crashing to the other. We can see from figure 5.6 that illustrates about the concept.



```
User
try {
Socket socket = new
Socket(Admin, 6270);
            :
}
```

```
Owner
try {
Socket socket = new
Socket(Admin, 6270);
            :
}
```

```
Admin
try {
ServerSocket serversocket = new
ServerSocket(6270);
while (true) {
  serversocket.acept();
                      :
  }
}
```

a) The clients initiate connection to the server

b) Each client has its own socket

Figure 5.6 Socket objects for each component

5. 3. 2 Database

The database is a place that is used to save the data from the owner. It is in the administrator side and managed by the administrator. There is only the administrator that can access the database directly. All clients and the owner access the database through the administrator.

We suppose that the database is in the administrator location, in the local net of the administration, even in the administrator computer. Therefore the communication doesn't need cryptography. Also, the data that is saved in the database has been encrypted.

Communication between the administrator and the database happens when the administrator saves or load the data based on the request from the user or the owner. We use the Access Database from Microsoft Office, and use **SQL (Sequence Query Language)** to access the database. Other database platforms can be used also, such as MySql, Oracle, and soon. JDBC API from Java provides a standard library for accessing relational database. Using the JDBC API, we can access a wide variety of different SQL database with exactly the same Java syntax.

There are seven standard steps in querying databases:

1. Load the JDBC Driver
2. Define the connection URL

3. Establish the connection
4. Create a statement object
5. Execute a query or update
6. Process the result
7. Close the connection

The driver manager is the piece of software that knows how to talk to the actual database server. To load the driver, we need to load the appropriate class using instruction *Class.forName* , and a static block in the class itself automatically makes a driver instance and registers it with the JDBC driver manager. After loading the driver, the location of the database server must be specified. URLs referring to databases use the *jdbc:protocol* and have the server host, port, and database name (or reference) embedded within the URL. After that we need to make connection to pass the URL, the database username and password. Then a statement object is used to send queries and commands to the database. After executing the query and process the results, the connection has to be closed. Figure 5.7 shows the steps in Java.

```
    try {
//      Class.forName("org.gjt.mm.mysql.Driver"); // Step 1
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); // step1
    } catch (Exception E) {
        E.printStackTrace();
    } // end catch
    try {
      con = DriverManager.getConnection("jdbc:odbc:Database1");//,"root",""); // step 2 and 3
      stmt = con.createStatement();  // step 4
      rs = stmt.executeQuery("SQL Query'"); // step 5
          while (rs.next()) {  // step 6
                :
                :
          } // while
      } // end while
      stmt.close();  // step7
      con.close();
    }
```

Figure 5.7 the steps in querying database

5. 4 Owner

The owner implements client's request to decrypt data that has been got from the administrator. When a client requests a decryption, the owner decrypts the data and sends back to the client. There is only the owner that has the key to decrypt the data.

The owner is built by the **Java applications,** and has responsibility as follows:

- Encrypt the data and sends to the administrator
- Decrypt the data from the user
- Hold both the public key and the private key.

Since the owner does encryption and decryption, the key (RSA public and private key) is in the owner. We build the owner application with the Java Socket programming and use port **6275** in this project.

Communication between the owner and the user happens when the user has gotten the data from the administrator and wants to decrypt it. The communication is principally the same as between the user and the Administrator. It is based on the TCP/IP protocol with Java Socket. The communication uses either user's RSA encryption or a key agreement to protect the data after being decrypted by the owner. We use Diffie-Hellman algorithm for the key-agreement, and each side use a secret key after agreement for communication. Figure 5.8 shows about communication between the owner and the user in the sequence diagram. If communication uses user's RSA key, we don't need agreement first, but data must be encrypted.

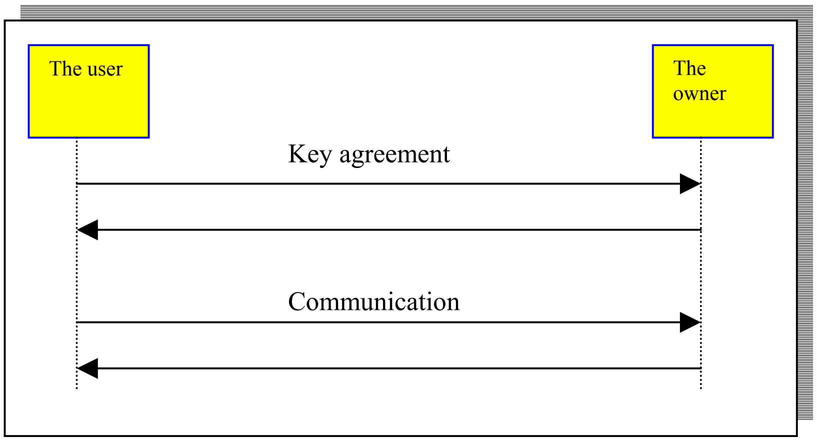


Figure 5.8 Communication between the owner and the user in sequence diagram

Communication between the owner and the Administrator happens when the owner saves the data to the database or loads data from the database. The owner must do that through the administrator. It is also based on the TCP/IP protocol. The data that is sent is secure since the owner encrypts it before sends it to the administrator. Figure 5.9 shows the communication when the owner saves data. The same way occurs when the owner loads data.

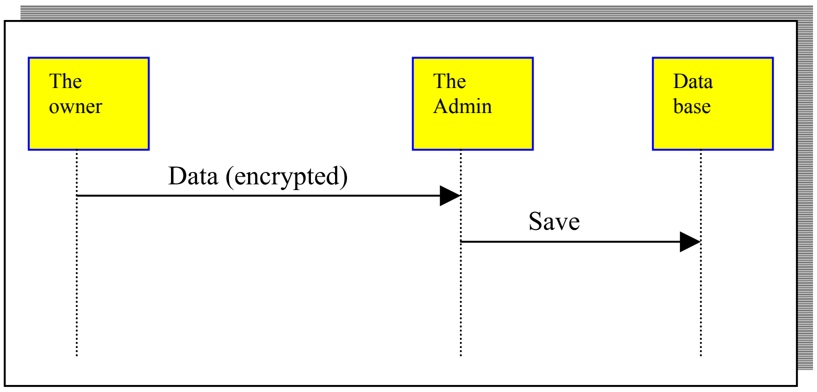


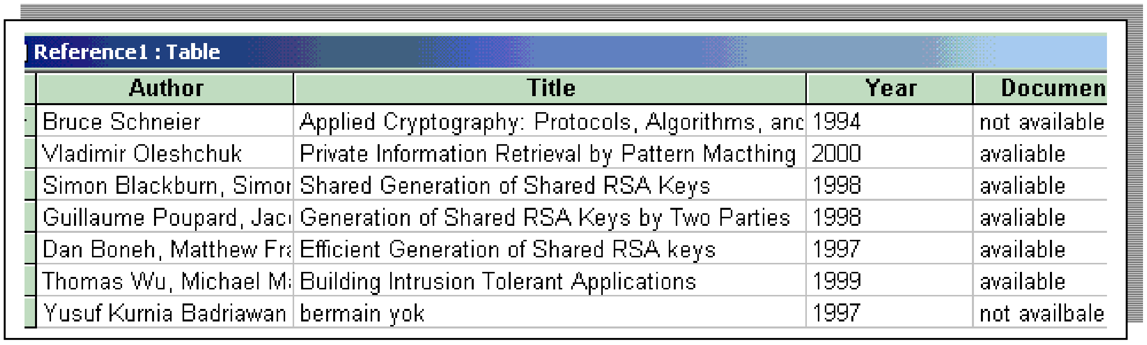
Figure 5.9 Communication between the owner and the administrator

In case there are two owners in the system, the second owner uses port **6276**. The system on the second owner is precisely the same as the first owner, unless the port. They use the same public key in order to encrypt data, and share the private key. The first owner holds d_1 and the second owner holds d_2, such that $d = d_1 + d_2$. When the user has discovered that data is owned by two owners, the user then sends it to the owners to ask decryption. After getting data, the user multiplies the data in order to get the real data.

Chapter 6 Result

6. 1 Database Implementation

Figure 6.1 shows the un-encrypted data table in the database. We do not use the table in the implementation. This is used to test only that the system works well.

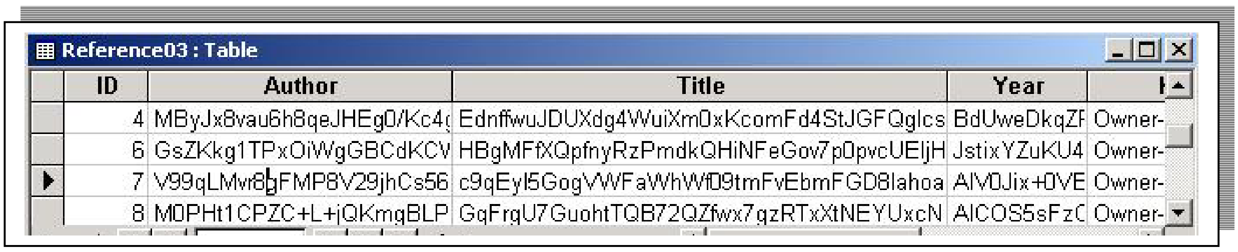


Reference1 : Table

Author	Title	Year	Documen
Bruce Schneier	Applied Cryptography: Protocols, Algorithms, anc	1994	not available
Vladimir Oleshchuk	Private Information Retrieval by Pattern Macthing	2000	avaliable
Simon Blackburn, Simor	Shared Generation of Shared RSA Keys	1998	avaliable
Guillaume Poupard, Jac	Generation of Shared RSA Keys by Two Parties	1998	avaliable
Dan Boneh, Matthew Fr	Efficient Generation of Shared RSA keys	1997	available
Thomas Wu, Michael M	Building Intrusion Tolerant Applications	1999	available
Yusuf Kurnia Badriawan	bermain yok	1997	not availbale

Figure 6.1 Un-encrypted data

Figure 6.2 is the encrypted data in the database, which is the encryption of the table from Figure 6.1. Each column is encrypted with RSA encryption, such that no one can read it without decryption. This is the table that we use in the application. Note that we do not break each column item into blocks to encrypt the data. Each column is one block that becomes encrypted with a public RSA key.



Reference03 : Table

ID	Author	Title	Year	I
4	MByJx8vau6h8qeJHEg0/Kc4	EdnffwuJDUXdg4WuiXm0xKcomFd4StJGFQglcs	BdUweDkqZF	Owner-
6	GsZKkg1TPxOiWgGBCdKCV	HBgMFfXQpfnyRzPmdkQHiNFeGov7p0pvcUEljH	JstixYZuKU4	Owner-
7	V99qLMvr8gFMP8V29jhCs56	c9qEyl5GogVWFaWhWf09tmFvEbmFGD8lahoa	AlV0Jix+0VE	Owner-
8	M0PHt1CPZC+L+jQKmqBLP	GqFrqU7GuohtTQB72QZfwx7qzRTxXtNEYUxcN	AICOS5sFzC	Owner-

Figure 6.2 Encrypted data

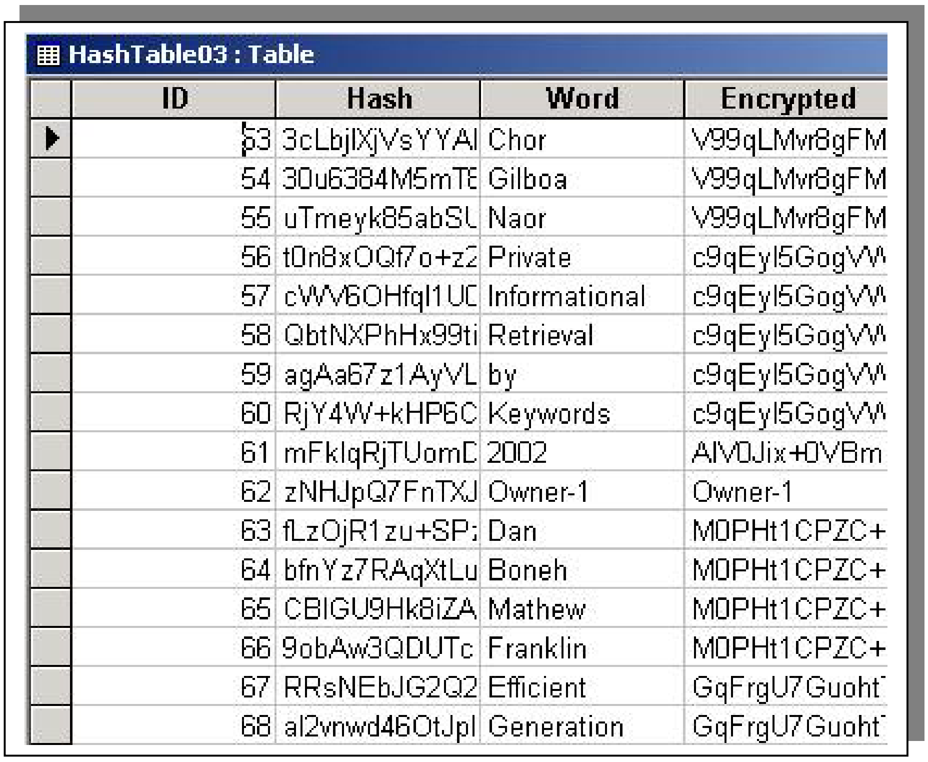
Column Author in the *Reference03* table is the encryption of the Author column in the *Reference1* table, column Title in the *Reference03* table is the encryption of column Title in the *Reference1* table, and so on.

Because the *Reference03* table is encrypted, we cannot read each item partly. For example, we cannot take on the Author column and row 1st only "V99qLM", because it cannot be decrypted. We have to take all string in the item to be able to decrypt it.

This is very difficult to read or to estimate about the items because of the encryption; even it cannot be taken partly. The user definitely cannot send the query that contains the all encrypted string. The user probably wants to know about some information, so he/she only writes some words and then send it.

In order to the case, the owner makes a table that transform the word from the user to an item. With security reason, the user must hashes first before send it, which we have discussed before. The table therefore must include the hash and the encrypted item. Below is the hash table that transform the hashed words to the encrypted items. The column **Word** is not used in the practice, it is only used to make easier to read what the data is.

Figure 6.3 shows the hash table in this project. As we discussed above, the administrator uses the hash table to match the query from the user, in order the user wants get the data.


HashTable03 : Table

ID	Hash	Word	Encrypted
53	3cLbjIXjVsYYAI	Chor	V99qLMvr8gFM
54	30u6384M5mTE	Gilboa	V99qLMvr8gFM
55	uTmeyk85abSU	Naor	V99qLMvr8gFM
56	t0n8xOQf7o+z2	Private	c9qEyI5GogVW
57	cWV6OHfqI1UC	Informational	c9qEyI5GogVW
58	QbtNXPhHx99ti	Retrieval	c9qEyI5GogVW
59	agAa67z1AyVL	by	c9qEyI5GogVW
60	RjY4W+kHP6C	Keywords	c9qEyI5GogVW
61	mFkIqRjTUomD	2002	AIV0Jix+0VBm
62	zNHJpQ7FnTXJ	Owner-1	Owner-1
63	fLzOjR1zu+SP;	Dan	M0PHt1CPZC+
64	bfnYz7RAqXtLu	Boneh	M0PHt1CPZC+
65	CBIGU9Hk8iZA	Mathew	M0PHt1CPZC+
66	9obAw3QDUTc	Franklin	M0PHt1CPZC+
67	RRsNEbJG2Q2	Efficient	GqFrgU7Guoht
68	aI2vnwd46OtJpl	Generation	GqFrgU7Guoht

Figure 6.3 Hash Table

6. 2 Case 1: Implementation with One Owner

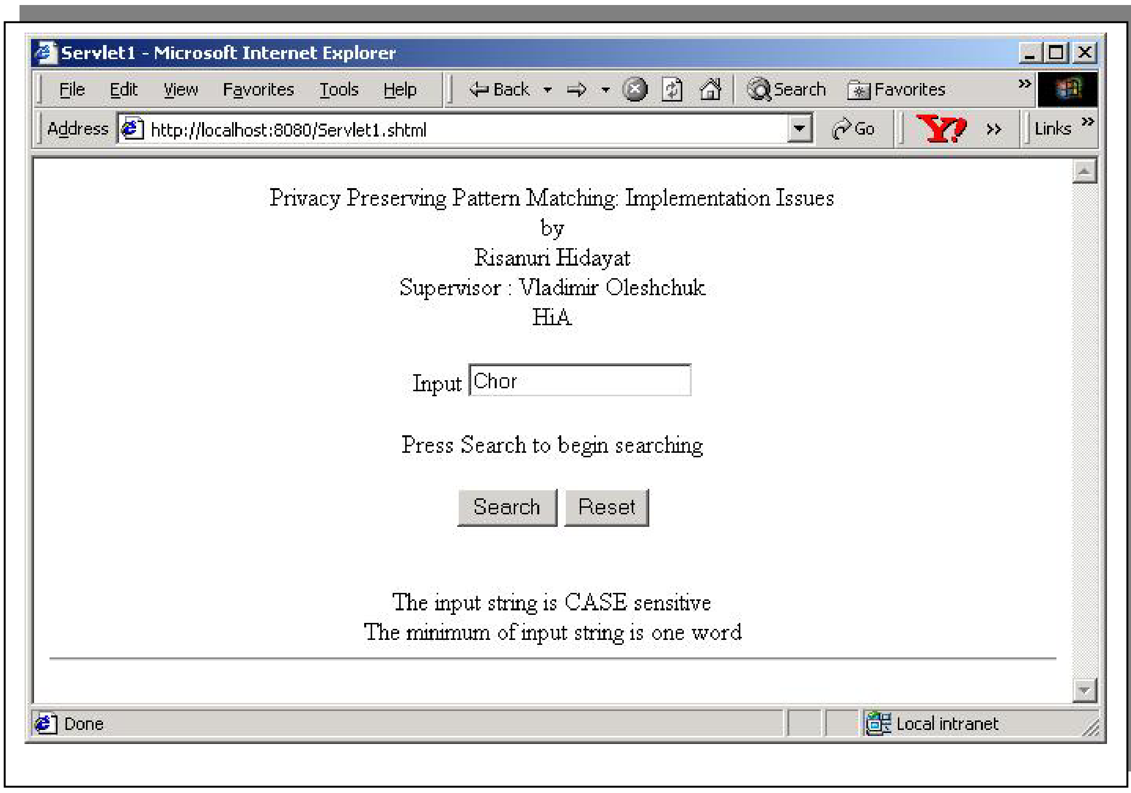
A client contacts the administrator through the Web Server. As we discussed, the web server acts as the user in the system. The client writes a word for searching; the user then hashes it and sends to the administrator. Figure 6.4 shows the web page for clients in this project. When the client writes an input string as a query, for example:

Chor

The user sends the hash of the query (*h (Chor)=3cLbjIXjVsYYANapqgozTA==)* to the administrator. The administrator then matches the query with its hash table. Since it is match with one of the item (see figure 6.3), the administrator uses the column (item) **Encrypted** in the hash table to look into the encrypted data in the database.

We use command prompt to monitor the interaction in the implementation. Figure 6.5 monitors on the user side. This is the implementation with user'r RSA encryption/decryption. After decryption with the user decryption key, the data is:

Chor, Gilboa, Naor


Servlet1 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/Servlet1.shtml
Go Links
Search Reset
Done
Local intranet

Privacy Preserving Pattern Matching: Implementation Issues
by
Risanuri Hidayat
Supervisor : Vladimir Oleshchuk
HiA

Input Chor

Press Search to begin searching

The input string is CASE sensitive
The minimum of input string is one word

Figure 6.4 the web page in the user side

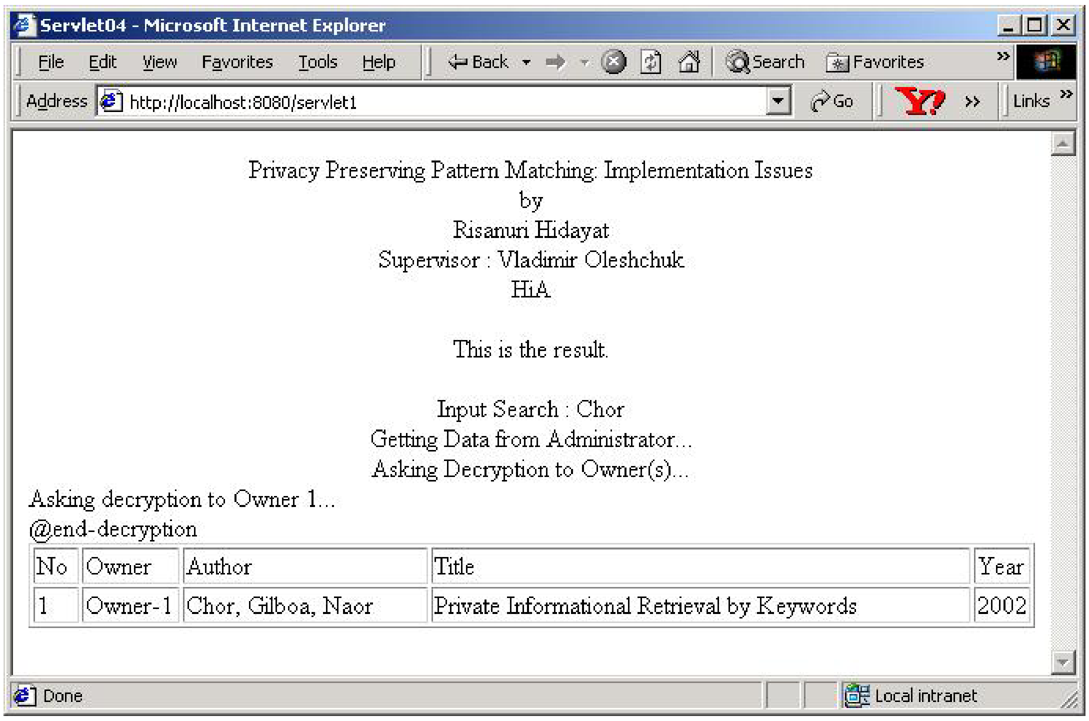
```
To Admin : 3cLbjIXjVsYYANapqgozTA==

From Admin :
V99qLMvr8gFMP8V29jhCs56shFhIP4/6k3I4fPNsZwB/irqMUZhgZ5ewRCcsZpgGRn7OMgjE4NQ3eRcSe
6uxNMMDUwIXvUyrfbbdkHDRUe4Cnv2BXKoWj1vfRQq5OsbYdU7NdDWvPDvwZJFBR3S0SDPwxSf
NcvHzNDjJ0uGC0cE=

Encryption...
Done Encryption...

To Owner :
AJ1ggRLk/ZzLh0Fn+i47N7t0p9+rC/lOSRSlKOLVNA7iJBfjDQ0sG/BjH+yzSgKatnDHyFAP31ZO
Ou+224i7Sp+8Q1PulZ4WZ5bpjA22cCUkKg+j/7R+4iXwBzmvYXaFLc8goav9EiB2bsB1Nwe5aAKv
0EmaJssmqYUVyfjmAE3l

From Owner :
AJeyv6aQwF2wAoJNLRTynur20pK3xYM60fc2zJviRlAt8y55XH3OyCi1wt3wWf8nxgM9HC/9NXeOOIZ2
l3feluXxYA53ysqi31xTf6TM4jIXC/FPO+RTOCfBwSzIELWgclPh+hFAd0GgaW2sYl2cbKD4Nb2nf3Rw4I
mKrc1c3FEj

Decryption...
Done Decryption...
```

Figure 6.5 Monitoring on the user side with User’s RSA key

Figure 6.6 shows the monitoring from the owner side. This is the first data. For the next data, the same processes are done. The final results can be seen from the html page as Figure 6.7.

```
From User :
AJ1ggRLk/ZzLh0Fn+i47N7t0p9+rC/lOSRSlKOLVNA7iJBfjDQ0sG/BjH+yzSgKatnDHyFAP31ZOOu+224i
7Sp+8Q1PulZ4WZ5bpjA22cCUkKg+j/7R+4iXwBzmvYXaFLc8goav9EiB2bsB1Nwe5aAKv0EmaJssmqYU
VyfjmAE3l

Decryption...
Done Decryption...

To User :
AJeyv6aQwF2wAoJNLRTynur20pK3xYM60fc2zJviRlAt8y55XH3OyCi1wt3wWf8nxgM9HC/9NXeO
OIZ2l3feluXxYA53ysqi31xTf6TM4jIXC/FPO+RTOCfBwSzIELWgclPh+hFAd0GgaW2sYl2cbKD4
Nb2nf3Rw4ImKrc1c3Fej
```

Figure 6.6 Monitoring on the owner side with User's RSA key



Servlet04 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/servlet1
Go Links
Done
Local intranet

Privacy Preserving Pattern Matching: Implementation Issues
by
Risanuri Hidayat
Supervisor : Vladimir Oleshchuk
HiA

This is the result.

Input Search : Chor
Getting Data from Administrator...
Asking Decryption to Owner(s)...
Asking decryption to Owner 1...
@end-decryption

No	Owner	Author	Title	Year
1	Owner-1	Chor, Gilboa, Naor	Private Informational Retrieval by Keywords	2002

Figure 6.7 The Final result in the user

From the figures, the data to the owner are the different from the data received from the administrator. The reason is the user encrypts with its RSA encryption key the data before send to the owner. The owner decrypts the data without knows what is the data about because it is still encrypted with the user's key. So the data sent back to the user is still encrypted, and the user decrypts the data with its RSA decryption key key before the data can be accessed.

This method ensures privacy for both data in the database and the user, since no one even the owner knows about data accessed by the user. But in this case, the user's key to decrypt the data must have the same modulus with the key from the owner. To generate the user'r key, the user must know about *modulus (N)* and *phi (p-1)(q-1)*. Therefore it will reduce the security level of the RSA keys, but it ensures high privacy about the data and the user.

When we implement with key agreement between the user and the owner, the user builds a key agreement with the owner after receiving data from the administrator. The process of the agreement is shown in Figure 6.8.

The owner(Server)	*The user (Client)*
Sending server public key ... *Done Sending server public key ...* *Receiving Client's Public key ...* *Done Receiving Client's Public key ...* *Performing Agreement ...* *Done Performing Agreement ...*	*Receiving server public key ...* *Done Receiving server public key ...* *Sending Client's Public key ...* *Done Sending Client's Public key ...* *Performing Agreement ...* *Done Performing Agreement ...*

Figure 6.8 Key Agreement Process

After that, we can see from the command prompt we monitor that the user sends and receives messages. Figure 6.9 shows the process from the user side for the first data. After decryption with the secret key, the data is:

Chor, Gilboa, Naor

```
To Admin : 3cLbjIXjVsYYANapqgozTA==

From Admin :
V99qLMvr8gFMP8V29jhCs56shFhIP4/6k3I4fPNsZwB/irqMUZhgZ5ewRCcsZpgGRn7OMgjE4NQ3eRcSe
6uxNMMDUwIXvUyrfbbdkHDRUe4Cnv2BXKoWj1vfRQq5OsbYdU7NdDWvPDvwZJFBR3S0SDPwxSf
NcvHzNDjJ0uGC0cE=

To Owner :
V99qLMvr8gFMP8V29jhCs56shFhIP4/6k3I4fPNsZwB/irqMUZhgZ5ewRCcsZpgGRn7OMgjE4NQ3eRcSe
6uxNMMDUwIXvUyrfbbdkHDRUe4Cnv2BXKoWj1vfRQq5OsbYdU7NdDWvPDvwZJFBR3S0SDPwxSf
NcvHzNDjJ0uGC0cE=

From Owner :
 iXJkPD7Ujn/CvgookkDb6DkuBt7nu+qY
```

Figure 6.9 Monitoring on the user side with key agreement

Figure 6.10 is the monitoring on the owner side. The final result is the same as using user's RSA encryption/decryption, and presented in the Figure 6.7.

```
From User :
V99qLMvr8gFMP8V29jhCs56shFhIP4/6k3I4fPNsZwB/irqMUZhgZ5ewRCcsZpgGRn7OMgjE4NQ3eRcSe
6uxNMMDUwIXvUyrfbbdkHDRUe4Cnv2BXKoWj1vfRQq5OsbYdU7NdDWvPDvwZJFBR3S0SDPwxSf
NcvHzNDjJ0uGC0cE=

Decryption...
Done Encryption...

To User :
iXJkPD7Ujn/CvgookkDb6DkuBt7nu+qY
```

Figure 6.10 Monitoring on the owner side with key agreement

We can see from the monitors that on the user side, the data to the owner are the same as the data received from the administrator. The reason is that the data is not encrypted because it has been encrypted since from the administrator. But after the owner decrypts the data and sends to the user, it must be encrypted to secure the data. Therefore the user must decrypt it with secret key from key agreement before the data can be accessed.

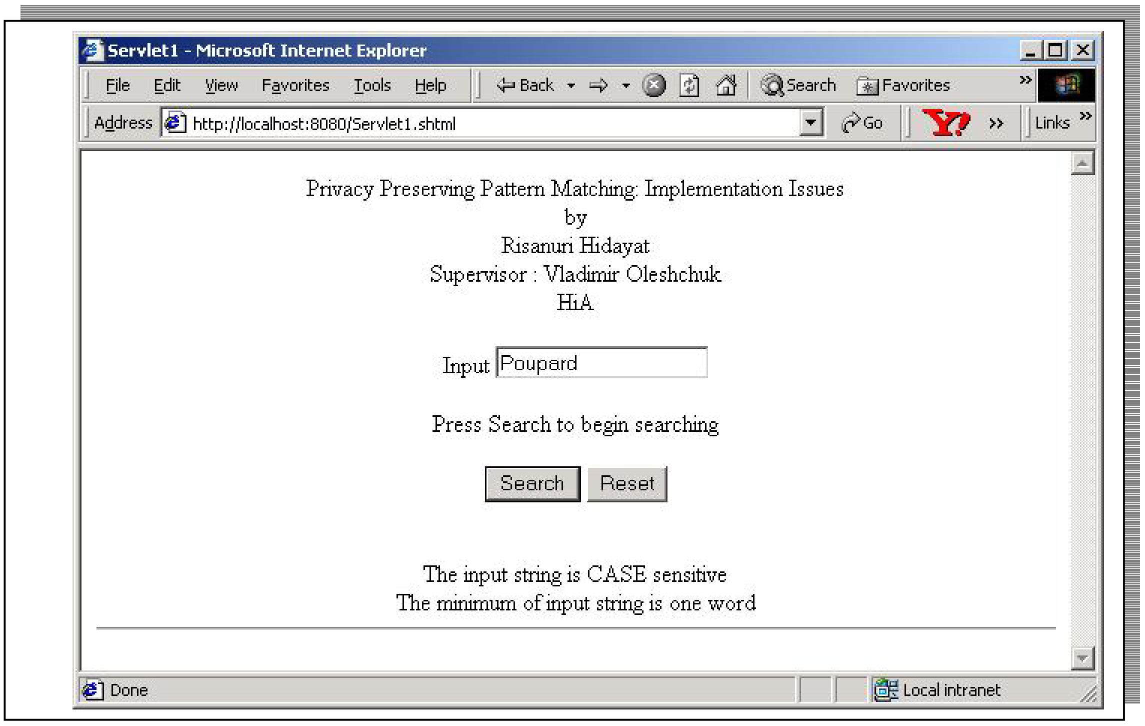
This method ensures privacy only for the data in the database, but not for the user, because after the owner decrypts the data, the owner knows what the user will access. But the method secures the RSA encryption and decryption keys, because the keys is never revealed even to the permitted user.

6. 3 Case 2: Implementation with Two Owners

Implementation with two owners is principally the same as in the one owner implementation, unless the user has to contact to the both owners. From html page as Figure 6.11, for example, the input string as a query is:

Poupard

The same process occurs to the administrator as in the one owner implementation. Figure 6.12 shows the monitoring on the user side when we implement user's RSA encryption/decryption. All monitorings are presented only for the first data.


Servlet1 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/Servlet1.shtml
Go Links
Search Reset
Done
Local intranet

Privacy Preserving Pattern Matching: Implementation Issues
by
Risanuri Hidayat
Supervisor : Vladimir Oleshchuk
HiA

Input Poupard

Press Search to begin searching

The input string is CASE sensitive
The minimum of input string is one word

Figure 6.11 the web page on the user side

```
To Admin : A6xJsSBZ/+o3lrYR24x/bw==

With Owner-1 ***
From Admin :
Zr2R3YG59+gDGczybRx0JQyJwsaOWA+zVBCruNPN7WbVULXfum4mH71lF3eYkzhcGIoRrE3pdBF8J2
xaqt36dgoZ2rhNZi1+NBeO2IZlc6wiVuvhelTe4F8NjHNLBBPRjHaNVhgSrg5c7nC858m7bhpmI8bT8oFV0
+9vZ3HOcY0=

Encryption...
Done Encryption...

To Owner :
YoQa0/ig6aMWs+O/p/sQ3L7yS6qHvk2tdBde3KKX7yJpWlzUmLSjK5T/oSdgDPXj8xcAg+/j7/v4
FpmtjO7Il58/+sJ2xaXPBYji6q63Po7Z59wK+FF8Dcuqnufm0f6k2fXv8hCE4jhEj49fNogUZXAL
ccts5tyeMxC0FAdm89w=

From Owner :
bHJY01K+jLGemjt79qr+C68r2DVNUSpVyaA500xni4xlXJcIAO7092U/Y3BW1Oc14BjYw7ivsyZmUugo
X2h0mgmIE8eYAmaM9a+xqkhWDPceF+llz4m2Q09KDxicT9/WSwt+2Pxh4SLxduSFOwJfHAQ7ydHSePs
ipCegkuyxTvo=

Decryption...
Done Encryption...

HBXLDQnuA3sycl18NfeaUjruMu9v8UiB5+MIRNN02bBGKO49boGPsLDM41gMDHThj021G6+SywaQ
zHL+stAdPhn9OV8pJbo5tgZBU7hxUO53PJyX9z4y5wyQp15gTrkADCsDGxk225EcPZLgnCnB6/Ly
7zb6ecY+wN6qz3OSK50=

With Owner-2 ***
From Admin :
Zr2R3YG59+gDGczybRx0JQyJwsaOWA+zVBCruNPN7WbVULXfum4mH71lF3eYkzhcGIoRrE3pdBF8J2
xaqt36dgoZ2rhNZi1+NBeO2IZlc6wiVuvhelTe4F8NjHNLBBPRjHaNVhgSrg5c7nC858m7bhpmI8bT8oFV0
+9vZ3HOcY0=

Encryption...
Done Encryption...

To Owner :
YoQa0/ig6aMWs+O/p/sQ3L7yS6qHvk2tdBde3KKX7yJpWlzUmLSjK5T/oSdgDPXj8xcAg+/j7/v4
FpmtjO7Il58/+sJ2xaXPBYji6q63Po7Z59wK+FF8Dcuqnufm0f6k2fXv8hCE4jhEj49fNogUZXAL
ccts5tyeMxC0FAdm89w=

From Owner :
TXdVEcHkqr5adp5AsIkwCli1rrSlFUIZIVj9IV3C5H/qgPuVTS0bzTYr66Zzcy8S+/5Et1M7XE1rL3BIaxD6i
HsimwRfjJ/zqGzY1LN6m46ljRqRPGJWNEMZIhM0eCpu7zEJb2q1q/vi5mpOcug5YIlRzqQ7S1P2K7tCG+
gIlNs=

Decryption...
Done Decryption...

NFWB8T1cb+OqzzXNXTJMUeEjGp/laSrC0zyt/LDpODvkSjqiqR7u4RVtBQs6ll9QdwSWQkvgy1pa
```

Figure 6.12 Monitoring on the user side with User’s RSA key

```
From User :
YoQa0/ig6aMWs+O/p/sQ3L7yS6qHvk2tdBde3KKX7yJpWlzUmLSjK5T/oSdgDPXj8xcAg+/j7/v4FpmtjO7I
l58/+sJ2xaXPBYji6q63Po7Z59wK+FF8Dcuqnufm0f6k2fXv8hCE4jhEj49fNogUZXALccts5tyeMxC0FAdm
89w=

Decryption...
Done Encryption...

To User :
bHJY01K+jLGemjt79qr+C68r2DVNUSpVyaA500xni4xlXJcIAO7092U/Y3BW1Oc14BjYw7ivsyZm
```

Figure 6.13 Monitoring on the owner 1 side with User's RSA key

```
From User :
YoQa0/ig6aMWs+O/p/sQ3L7yS6qHvk2tdBde3KKX7yJpWlzUmLSjK5T/oSdgDPXj8xcAg+/j7/v4FpmtjO7I
l58/+sJ2xaXPBYji6q63Po7Z59wK+FF8Dcuqnufm0f6k2fXv8hCE4jhEj49fNogUZXALccts5tyeMxC0FAdm
89w=

Decryption...
Done Encryption...

To User :
TXdVEcHkqr5adp5AsIkwCli1rrSlFUIZIVj9IV3C5H/qgPuVTS0bzTYr66Zzcy8S+/5Et1M7XE1r
```

Figure 6.14 Monitoring on the owner 2 side with User's RSA key

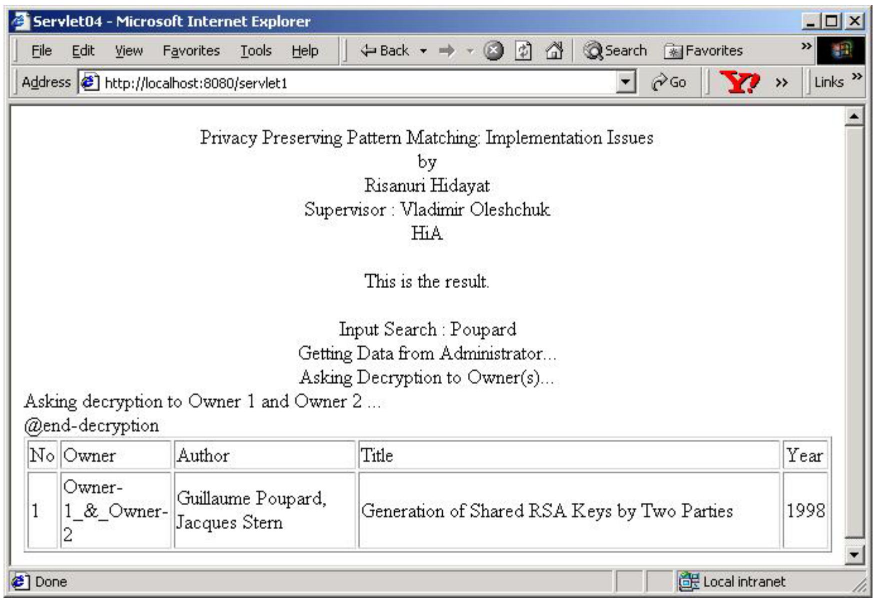
Figure 6.13 shows monitoring on the first owner's side, and Figure 6.14 shows monitoring on the second owner's side. Multiplication modulus *N* (encryption/decryption modulus) between the message from the owner 1 and the message from the owner 2 will result the real data. The multiplication occurs in the user, and Figure 6.15 shows the listing program of the multiplication. Here, *detexbyte1* is data from owner-1 and *detextbyte2* is from owner-2. Bold style is the multiplication program as we have discussed in section 2.2.2 and 4.4.

Note that the multiplication needs the same modulus (*N*). Since it happens in the user side, the modulus must be distributed to the user.

```
:
BigInteger detextbig1 = new BigInteger(detextbyte1);
BigInteger detextbig2 = new BigInteger(detextbyte2);
BigInteger n = this.getrsakeyimpl("publicmodulus1.txt");
BigInteger newbig = detextbig1.multiply(detextbig2).mod(n);
detext = new String(newbig.toByteArray());
:
```

Figure 6.15 A part of the listing programs of multiplication data

Finally, Figure 6.16 is the result on the user.


Servlet04 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/servlet1
Go Links
Done
Local intranet

Privacy Preserving Pattern Matching: Implementation Issues
by
Risanuri Hidayat
Supervisor : Vladimir Oleshchuk
HiA

This is the result.

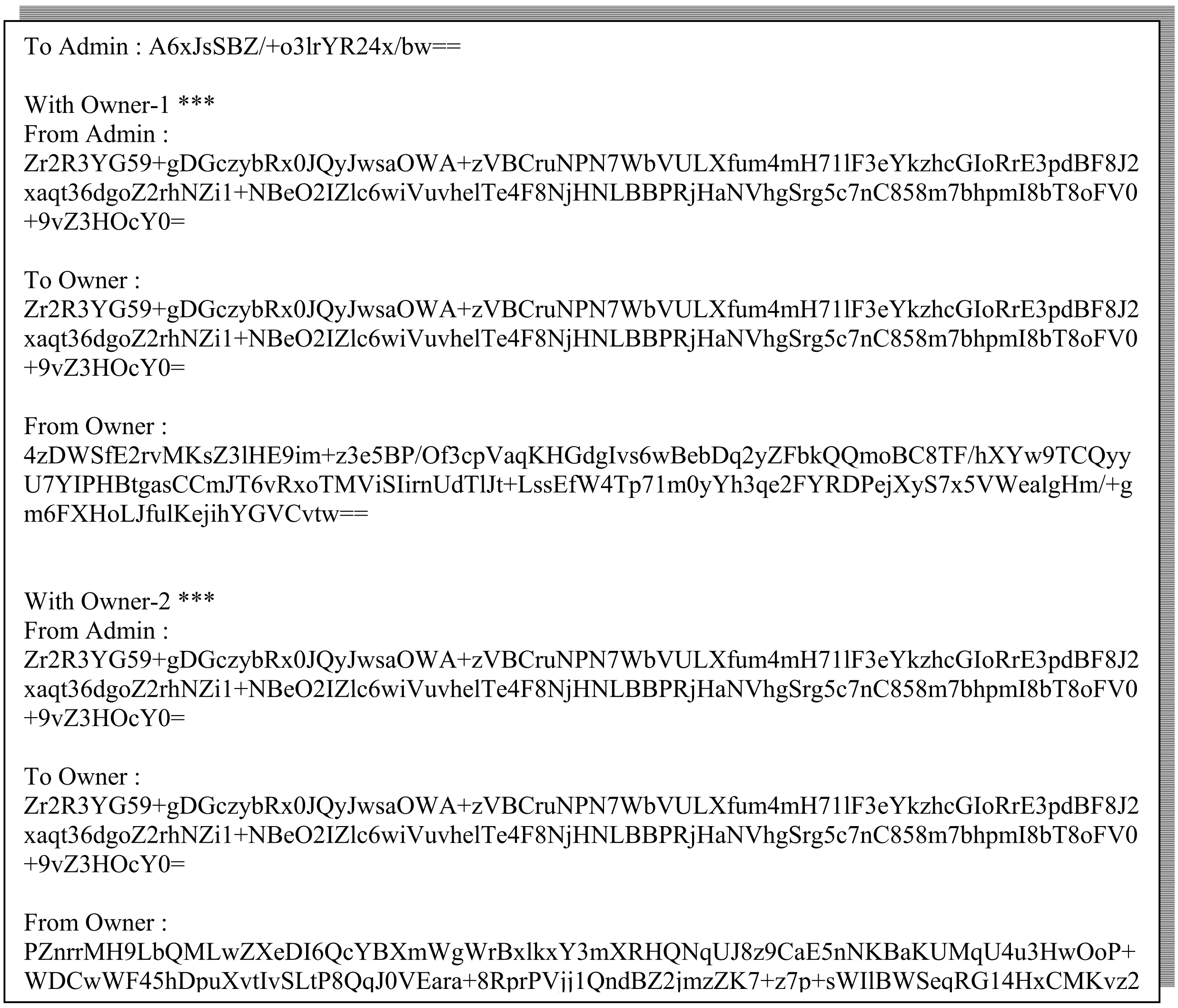
Input Search : Poupard
Getting Data from Administrator...
Asking Decryption to Owner(s)...
Asking decryption to Owner 1 and Owner 2 ...
@end-decryption

No	Owner	Author	Title	Year
1	Owner-1_&_Owner-2	Guillaume Poupard, Jacques Stern	Generation of Shared RSA Keys by Two Parties	1998

Figure 6.16 The Final result in the user

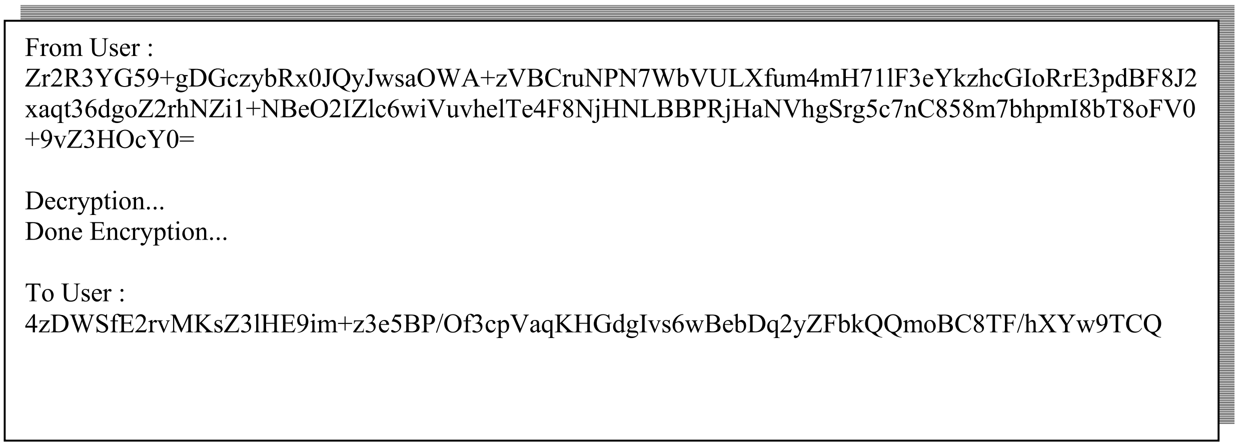
Implementation with Key Agreement for two owners is principally the same as for one owner. Note that the user does the key agreement with each owner, so the keys are different. Figures 6.17, 6.18, 6.19 respectively show the monitor from the user, owner-1, and owner-2 sides for the first data with input string:

Poupard



```
To Admin : A6xJsSBZ/+o3lrYR24x/bw==

With Owner-1 ***
From Admin :
Zr2R3YG59+gDGczybRx0JQyJwsaOWA+zVBCruNPN7WbVULXfum4mH71lF3eYkzhcGIoRrE3pdBF8J2
xaqt36dgoZ2rhNZi1+NBeO2IZlc6wiVuvhelTe4F8NjHNLBBPRjHaNVhgSrg5c7nC858m7bhpmI8bT8oFV0
+9vZ3HOcY0=

To Owner :
Zr2R3YG59+gDGczybRx0JQyJwsaOWA+zVBCruNPN7WbVULXfum4mH71lF3eYkzhcGIoRrE3pdBF8J2
xaqt36dgoZ2rhNZi1+NBeO2IZlc6wiVuvhelTe4F8NjHNLBBPRjHaNVhgSrg5c7nC858m7bhpmI8bT8oFV0
+9vZ3HOcY0=

From Owner :
4zDWSfE2rvMKsZ3lHE9im+z3e5BP/Of3cpVaqKHGdgIvs6wBebDq2yZFbkQQmoBC8TF/hXYw9TCQyy
U7YIPHBtgasCCmJT6vRxoTMViSIirnUdTlJt+LssEfW4Tp71m0yYh3qe2FYRDPejXyS7x5VWealgHm/+g
m6FXHoLJfulKejihYGVCvtw==


With Owner-2 ***
From Admin :
Zr2R3YG59+gDGczybRx0JQyJwsaOWA+zVBCruNPN7WbVULXfum4mH71lF3eYkzhcGIoRrE3pdBF8J2
xaqt36dgoZ2rhNZi1+NBeO2IZlc6wiVuvhelTe4F8NjHNLBBPRjHaNVhgSrg5c7nC858m7bhpmI8bT8oFV0
+9vZ3HOcY0=

To Owner :
Zr2R3YG59+gDGczybRx0JQyJwsaOWA+zVBCruNPN7WbVULXfum4mH71lF3eYkzhcGIoRrE3pdBF8J2
xaqt36dgoZ2rhNZi1+NBeO2IZlc6wiVuvhelTe4F8NjHNLBBPRjHaNVhgSrg5c7nC858m7bhpmI8bT8oFV0
+9vZ3HOcY0=

From Owner :
PZnrrMH9LbQMLwZXeDI6QcYBXmWgWrBxlkxY3mXRHQNqUJ8z9CaE5nNKBaKUMqU4u3HwOoP+
WDCwWF45hDpuXvtIvSLtP8QqJ0VEara+8RprPVjj1QndBZ2jmzZK7+z7p+sWIlBWSeqRG14HxCMKvz2
```

Figure 6.17 Monitoring on the user side with key agreement



```
From User :
Zr2R3YG59+gDGczybRx0JQyJwsaOWA+zVBCruNPN7WbVULXfum4mH71lF3eYkzhcGIoRrE3pdBF8J2
xaqt36dgoZ2rhNZi1+NBeO2IZlc6wiVuvhelTe4F8NjHNLBBPRjHaNVhgSrg5c7nC858m7bhpmI8bT8oFV0
+9vZ3HOcY0=

Decryption...
Done Encryption...

To User :
4zDWSfE2rvMKsZ3lHE9im+z3e5BP/Of3cpVaqKHGdgIvs6wBebDq2yZFbkQQmoBC8TF/hXYw9TCQ
```

Figure 6.18 Monitoring on the owner 1 side with key agreement

```
From User :
Zr2R3YG59+gDGczybRx0JQyJwsaOWA+zVBCruNPN7WbVULXfum4mH71lF3eYkzhcGIoRrE3pdBF8J2
xaqt36dgoZ2rhNZi1+NBeO2IZlc6wiVuvhelTe4F8NjHNLBBPRjHaNVhgSrg5c7nC858m7bhpmI8bT8oFV0
+9vZ3HOcY0=

Decryption...
Done Encryption...

To User :
PZnrrMH9LbQMLwZXeDI6QcYBXmWgWrBxlkxY3mXRHQNqUJ8z9CaE5nNKBaKUMqU4u3HwOoP+
WDCwWF45hDpuXvtIvSLtP8QqJ0VEara+8RprPVjj1QndBZ2jmzZK7+z7p+sWIlBWSeqRG14HxCMKvz2
DK0zLcHwkvk6pPJRect2BUoJCyMlBUA==
```

Figure 6.19 Monitoring on the owner 1 side with key agreement

Chapter 7 Conclusion and Future Work

7. 1 Conclusion

Recently the Internet has been in tremendous growth. The Internet technology is now becoming the world's largest public electronic place. Besides the potential growing of the Internet, it has the potential to erode personal privacies. The data owner might be not realizing that the data on the Internet are monitored or logged by some unseen third party.

The project is the implementation of the privacy preserving. Data with high privacy must be encrypted when connected to the Internet to prevent them from accessing data by unpermitted person. The implementation gives possibility to place data on insecure third-part side with pattern matching support without revealing either data or pattern. The system let the data owner hold both public and private keys. The data owner decrypts data for a permitted user, but without knowing what kind of data from database he is asked to decrypt.

Hash table is used to transform from input pattern to the encrypted data. The table is useful when one wants to search information about the data. It will allow matching of input pattern on encrypted data without decrypting them at all.

The implementation with user's RSA encryption/decryption when the user asks decryption to the owner ensures the privacy for the user and the data, because even the owner doesn't know the data accesed. But it reduces the scurity of the key since the user must use the same modulus and phi as the key for the data. The implementation with key agreement doesn't ensure the user from the owner because the owner knows what data the user accessed, but in this case the secret value of the RSA key is never distributed.

The system has possibility to share RSA key into some pieces when data belongs to more than one person. Some methods have been proposed in the threshold cryptography research. The implementation divides RSA private key using addition ($d=d_1+d_2$), such that the original data becomes multiplication of the partial data. But it needs distribution of modulus (N) because the multiplication occurs on the user side. Since modulus is a part of public key, it becomes no problem to distribute the modulus.

7. 2 Future Work

Until now, we have implemented the system with data belongs to two owners using classical generation key, where one of the owner generates a RSA key and then divides it into parts, holds one of the part, and distributes the other. In the future, we plan to investigate the generation of key with distributed computation of the key where each part doesn't know the secret values holds by other part.

References

Articles

1. Chor, N. Gilboa, M. Naor, *Private Informational Retrieval by Keywords*, Tech. Rep. TR CS097. Dept. Computer Science. Technion, Israel.
2. Dan Boneh and Matthew Franklin, Efficient Generation of Shared RSA keys, Comp. Science Dept., Stanford University, http://www.stanford.edu/~dabo/ITTC, 1997.
3. Guillaume Poupard and Jacques Stern, *Generation of Shared RSA Keys by Two Parties*, ASIACRYPT: International Conference on the Theory and Application of Cryptology, 1998.
4. Hu Aiqing Wang, Mathew KO Lee, Chen Wang, *Consumer Privacy Concerns about Internet Marketing*, Communication of the ACM, Vol. 41, No. 3, March, 1998.
5. Ian Goldberg, David Wagner, Eric Brewer, *Privacy Enhancing Technologies for Internet*, University of California, Berkeley, IEEE, 1997.
6. Paola Banesi, *Truste; an online privacy seal program*, Communication of the ACM, Vol. 42, No. 2, February, 1999.
7. Shamir, *How to Share a Secret,* Communication of the ACM, v.24, n. 11, 1979.
8. Simon Blackburn, Simon Blake-Wilson, Mike Burmester and Steven Galbraith, *Shared Generation of Shared RSA Keys*, Technical reports CORR 89-19, Dept. of C&O, University of Waterloo, Canada, 1998.
9. Tessa Lau, Oren Etzioni, Daniel S. Weld, *Privacy Interfaces for Information Management*, Communication of the ACM, Vol. 42, No. 10, October, 1999.
10. Thomas Wu, Michael Malkin and Dan Boneh, *Building Intrusion Tolerant Applications*, Comp. Science Dept., Stanford University, http://www.stanford.edu/~dabo/ITTC, 1999.
11. Vladimir Oleshchuk, *Private Information Retrieval by Pattern Macthing,* Proceeding of the Second International Workshop, Information Integration and Web-based Application & Services, 2000.

Books

12. Bruce Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, John Wiley& Sons Inc, 1994.
13. Jess Garms and Daniel Somerfield, *Professional Java Security*, Wrox Press, 2001
14. Marty Hall, *Core Servlets and JavaServer Pages*, Prentice Hall & Sun Mycrosystems, 2000.
15. Mel HX and Baker Doris, *Cryptography Decrypted*, Addison-Wesley, 2001

Internets

16. Legion of the Bouncy Castle, http://www.bouncycastle.org/
17. The source for Java Technology, http://java.sun.com/
18. The Jakarta project, http://jakarta.apache.org

Appendix

Administrator/Data Base Manager

```
package admin06;

/**
 * Title:        Administrator
 * Description:
 * Copyright:    Copyright (c) 2002
 * Company:      HiA
 * @author Risanuri HIdayat
 * @version 1.0
 */

import java.math.*;
import java.io.*;
import java.util.Enumeration;
import java.sql.*;
import java.net.*;
import java.security.*;
import javax.crypto.*;
import java.security.spec.*;
import javax.crypto.spec.*;
import javax.crypto.interfaces.*;
import sun.misc.*;


class Administrator06 extends Thread {
  int port = 6270;
  ServerSocket serversocket;
  Socket socket;
  BufferedReader in;
  PrintWriter out;
  String[] input = {"",""};
    AdminFrame06 adminFrame06 = new AdminFrame06(this);

  public Administrator06() throws IOException, Exception{
    this.start();
  }
  public void run() { //throws IOException { //, Exception {
    try {
      serversocket = new ServerSocket(port);
      System.out.println("Administrator Server starting on port "+port);
    }
    catch (Exception e) {
      System.out.println("Error creating server socket ...."+e.toString());
      System.exit(1);
    }
    while (true) {
      try {
        socket = serversocket.accept();
        in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
        out = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()),true);

        input[0] = this.receiveMessage();
        input[1] = this.receiveMessage();
        adminFrame06.list1.add(input[0]+"("+socket.getInetAddress()+") ->"+input[1]);
        if (input[0].equalsIgnoreCase("@user")) {
          this.UserCommunication(input[1]);
        }
        if (input[0].equalsIgnoreCase("@owner-read")) {
          this.OwnerGetData(input[1]);
        }
        if (input[0].equalsIgnoreCase("@owner-save")) {
          this.OwnerSavingToAdmin(input[1]);
        }
        socket.close();
        System.out.println("Socket closed");
        } catch (Exception e) {
          System.out.println("Error...."+ e);
        }
    }
  } // constructor

  public String ReceiveLongMessage() {
    String text = "";
    String str1 = "";
    try {
      String str = this.receiveMessage();
      str1 = str1 + str;
      while (!((str==null)||(str.equalsIgnoreCase("@end")))) {
        str = this.receiveMessage();
        if (!((str==null)||(str.equalsIgnoreCase("@end")))) { str1 = str1 + str; }
      }
      text = str1;
    }catch (Exception exp) {}

    return text;
  }

  public void OwnerSavingToAdmin(String text) {
    Connection con = null;
    Statement stmt;
    ResultSet rs;
    String wordtext = text;
    String str1 = "";
    String word = "";
    String hashtext = "";
    String entext = "";

    try {
//         Class.forName("org.gjt.mm.mysql.Driver");
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    } catch (Exception E) { E.printStackTrace(); } // end catch

    try {
      con = DriverManager.getConnection("jdbc:odbc:Database1");//,"root","");
      stmt = con.createStatement();

      System.out.println(text);
      String authenctext = this.ReceiveLongMessage();
      System.out.println("author :"+authenctext);

      text = this.receiveMessage();
      adminFrame06.list1.add(input[0]+"("+socket.getInetAddress()+") ->"+text);
      System.out.println(text);
      String titleenctext = this.ReceiveLongMessage();
      System.out.println("title :"+titleenctext);

      text = this.receiveMessage();
      adminFrame06.list1.add(input[0]+"("+socket.getInetAddress()+") ->"+text);
      System.out.println(text);
      String yearenctext = this.ReceiveLongMessage();
      System.out.println("year :"+yearenctext);

      text = this.receiveMessage();
      adminFrame06.list1.add(input[0]+"("+socket.getInetAddress()+") ->"+text);
      System.out.println(text);
      String keytext = this.ReceiveLongMessage();
      System.out.println("doc :"+keytext);

      stmt.executeUpdate("INSERT INTO Reference03 (Author,Title,Year,Key) VALUES('"+ authenctext +"','"+ titleenctext
+"','"+ yearenctext +"','"+ keytext +"');");

      text = this.receiveMessage();
      while (!(text.equalsIgnoreCase("@end"))) {
        if (text.equalsIgnoreCase("@author-hash")) {
          word = this.receiveMessage();
          hashtext = this.receiveMessage();
          entext = authenctext;
        }
        if (text.equalsIgnoreCase("@title-hash")) {
          word = this.receiveMessage();
          hashtext = this.receiveMessage();
          entext = titleenctext;
        }
        if (text.equalsIgnoreCase("@year-hash")) {
          word = this.receiveMessage();
          hashtext = this.receiveMessage();
          entext = yearenctext;
        }
        if (text.equalsIgnoreCase("@doc-hash")) {
          word = this.receiveMessage();
          hashtext = this.receiveMessage();
          entext = keytext;
        }
        if (!(text.equalsIgnoreCase(""))){
          adminFrame06.list1.add(input[0]+"("+socket.getInetAddress()+") ->"+text);
          System.out.println("word: "+word+" ** hash: "+hashtext+" ** encrypted: "+text);

          stmt.executeUpdate("INSERT INTO HashTable03 (Hash,Word,Encrypted) VALUES('"+ hashtext +"','"+ word
+"','"+ entext +"');");
        }
        text = this.receiveMessage();
      } // while

      stmt.close();
      con.close();
      System.out.println("Saving Succeed...");
    } catch (Exception e) {
      System.out.println("Error...."+ e);
    }

  } // OwnerSavingToAdmin

  public void OwnerGetData(String input) throws Exception {
    Connection con = null;
    Connection con2 = null;
    Statement stmt,stmt2;
    ResultSet rs,rs2;
    String text = "";

    try {
```

```
//      Class.forName("org.gjt.mm.mysql.Driver");
       Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
     } catch (Exception E) {
         E.printStackTrace();
     } // end catch
     try {
       con = DriverManager.getConnection("jdbc:odbc:Database1");//,"root","");
       stmt = con.createStatement();
       rs = stmt.executeQuery("SELECT Encrypted FROM HashTable03 WHERE Hash like '"+input+"'");
       while (rs.next()) {
           text = rs.getString("Encrypted");
           stmt2 = con.createStatement();
           rs2 = stmt2.executeQuery("SELECT * FROM Reference03 WHERE Author like '"+text+"' OR Title like '"+text+"' OR
Year like '"+text+"' OR Key like '"+text+"'");
           while (rs2.next()) {
               this.sendMessage(rs2.getString("Key"));
               this.sendMessage(rs2.getString("Author"));
               this.sendMessage(rs2.getString("Title"));
               this.sendMessage(rs2.getString("Year"));
           } // while
       } // end while

       if (text.equalsIgnoreCase("")) {
           text = "@not-found";
           this.sendMessage(text);
       }
       this.sendMessage("@end");
       stmt.close();
       con.close();
     }
     catch(SQLException ex) {
         System.err.println("SQLException : " + ex.getMessage());
         System.out.println("DATA BASE TIDAK DITEMUKAN...");
         text = "@not-found";
         this.sendMessage(text);
       } // end cacth
       this.sendMessage("@end");
   } // Owner Communication

   public void UserCommunication(String intext) throws Exception {
     Connection con = null;
     Connection con2 = null;
     Statement stmt,stmt2;
     ResultSet rs,rs2;
     String text = "";
     String output = "";
     String output2 = "";

       try {
//         Class.forName("org.gjt.mm.mysql.Driver");
         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
       } catch (Exception E) {
           E.printStackTrace();
       } // end catch
       try {
           con = DriverManager.getConnection("jdbc:odbc:Database1");//,"root","");
           stmt = con.createStatement();
           while (!(intext.equalsIgnoreCase("@end-from-user"))) {
             rs = stmt.executeQuery("SELECT Encrypted FROM HashTable03 WHERE Hash like '"+intext+"'");
             while (rs.next()) {
               text = rs.getString("Encrypted");
```

```
              stmt2 = con.createStatement();
              rs2 = stmt2.executeQuery("SELECT * FROM Reference03 WHERE Author like '"+text+"' OR Title like '"+text+"' OR
Year like '"+text+"' OR Key like '"+text+"'");
              while (rs2.next()) {
                 this.sendMessage(rs2.getString("Key"));
                 this.sendMessage(rs2.getString("Author"));
                 this.sendMessage(rs2.getString("Title"));
                 this.sendMessage(rs2.getString("Year"));
              } // while
            } // end while
            intext = "";
            intext = this.receiveMessage();
            adminFrame06.list1.add(input[0]+"("+socket.getInetAddress()+") ->"+intext);
          } //while
          stmt.close();
          con.close();
       }
       catch(SQLException ex) {
          System.err.println("SQLException : " + ex.getMessage());
          System.out.println("DATA BASE TIDAK DITEMUKAN...");
          text = "@not-found";
          this.sendMessage(text);
       } // end cacth
       this.sendMessage("@end-decryption");

    } // UserCommunication

    public void sendMessage(String string) {
       out.println(string);
    }

    public String receiveMessage() throws Exception {
       return (in.readLine());
    }

    public static void main(String[] args) throws IOException, Exception {
      Administrator06 administrator1 = new Administrator06();
   }
}
```

```
package admin06;

import java.awt.*;
import java.awt.event.*;

/**
 * Title:        Administrator
 * Description:
 * Copyright:    Copyright (c) 2002
 * Company:      HiA
 * @author Risanuri HIdayat
 * @version 1.0
 */

public class AdminFrame06 extends Frame {
  List list1 = new List();
  Button DOWNbutton = new Button();

  Administrator06 administrator;

  public AdminFrame06(Administrator06 admin) {
    this.administrator = admin;
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
//  public static void main(String[] args) {
//    AdminFrame03 adminFrame03 = new AdminFrame03();
//  }
  private void jbInit() throws Exception {
    this.setSize(400,300);
    this.setTitle("Administrator");
    DOWNbutton.setLabel("DOWN");
    DOWNbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        DOWNbutton_actionPerformed(e);
      }
    });
    this.addWindowListener(new java.awt.event.WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        this_windowClosing(e);
      }
    });
    this.add(list1, BorderLayout.CENTER);
    this.add(DOWNbutton, BorderLayout.SOUTH);
    this.setVisible(true);
  }

  void DOWNbutton_actionPerformed(ActionEvent e) {
      System.exit(0);
  }

  void this_windowClosing(WindowEvent e) {
      System.exit(0);
  }
}
```

Owner 1 (Key Agreement Implementation)
```
package owner107;

import java.io.*;
import java.net.*;
import java.math.*;
import java.security.*;
import java.security.spec.*;
import javax.crypto.*;
import javax.crypto.spec.*;
import javax.crypto.interfaces.*;
import sun.misc.*;
import java.util.*;
import risanuri01.*;

class AKAServer01 extends Thread {
  private static final byte SKIP_1024_MODULUS[] = {
   (byte)0xF4, (byte)0x88, (byte)0xFD, (byte)0x58,
   (byte)0x4E, (byte)0x49, (byte)0xDB, (byte)0xCD,
   (byte)0x20, (byte)0xB4, (byte)0x9D, (byte)0xE4,
   (byte)0x91, (byte)0x07, (byte)0x36, (byte)0x6B,
   (byte)0x33, (byte)0x6C, (byte)0x38, (byte)0x0D,
   (byte)0x45, (byte)0x1D, (byte)0x0F, (byte)0x7C,
   (byte)0x88, (byte)0xB3, (byte)0x1C, (byte)0x7C,
   (byte)0x5B, (byte)0x2D, (byte)0x8E, (byte)0xF6,
   (byte)0xF3, (byte)0xC9, (byte)0x23, (byte)0xC0,
   (byte)0x43, (byte)0xF0, (byte)0xA5, (byte)0x5B,
   (byte)0x18, (byte)0x8D, (byte)0x8E, (byte)0xBB,
   (byte)0x55, (byte)0x8C, (byte)0xB8, (byte)0x5D,
   (byte)0x38, (byte)0xD3, (byte)0x34, (byte)0xFD,
   (byte)0x7C, (byte)0x17, (byte)0x57, (byte)0x43,
   (byte)0xA3, (byte)0x1D, (byte)0x18, (byte)0x6C,
   (byte)0xDE, (byte)0x33, (byte)0x21, (byte)0x2C,
   (byte)0xB5, (byte)0x2A, (byte)0xFF, (byte)0x3C,
   (byte)0xE1, (byte)0xB1, (byte)0x29, (byte)0x40,
   (byte)0x18, (byte)0x11, (byte)0x8D, (byte)0x7C,
   (byte)0x84, (byte)0xA7, (byte)0x0A, (byte)0x72,
   (byte)0xD6, (byte)0x86, (byte)0xC4, (byte)0x03,
   (byte)0x19, (byte)0xC8, (byte)0x07, (byte)0x29,
   (byte)0x7A, (byte)0xCA, (byte)0x95, (byte)0x0C,
   (byte)0xD9, (byte)0x96, (byte)0x9F, (byte)0xAB,
   (byte)0xD0, (byte)0x0A, (byte)0x50, (byte)0x9B,
   (byte)0x02, (byte)0x46, (byte)0xD3, (byte)0x08,
   (byte)0x3D, (byte)0x66, (byte)0xA4, (byte)0x5D,
   (byte)0x41, (byte)0x9F, (byte)0x9C, (byte)0x7C,
   (byte)0xBD, (byte)0x89, (byte)0x4B, (byte)0x22,
   (byte)0x19, (byte)0x26, (byte)0xBA, (byte)0xAB,
   (byte)0xA2, (byte)0x5E, (byte)0xC3, (byte)0x55,
   (byte)0xE9, (byte)0x2F, (byte)0x78, (byte)0xC7
   };
  private static final BigInteger MODULUS = new
BigInteger(1,SKIP_1024_MODULUS);
  private static final BigInteger BASE = BigInteger.valueOf(2);
  private static final DHParameterSpec PARAMETER_SPEC = new
DHParameterSpec(MODULUS,BASE);

   String input = null;
   String output = null;
   String adminhost = "ikt02-16";
   SecretKey secretkey;
   IvParameterSpec spec;
  Cipher cipher;
  ServerSocket serversocket;
  Socket socket;
  int port = 6275;
  BufferedReader in;
  PrintWriter out;
  DataInputStream dis;
  DataOutputStream dos;
  BASE64Decoder deBASE64tobyte = new BASE64Decoder();
  BASE64Encoder enBASE64toString = new BASE64Encoder();
  boolean KAdone = false;
  OwnerFrame107 ownerFrame107 = new OwnerFrame107(this);
  ReadFrame107 readFrame107 = new ReadFrame107(this);
  SendFrame107 sendFrame107 = new SendFrame107(this);

  public AKAServer01()  throws Exception {
      this.start();
  }
  public void run() { //throws IOException { //, Exception {
      String str1 = "";
      try {
          serversocket = new ServerSocket(port);
          System.out.println("Owner-1 .. Listening on port "+port);
//          socket = serversocket.accept();
      } catch (Exception e) {
          System.out.println("Error creating server socket
...."+e.toString());
          System.exit(1);
      }

      while (true) {
          try {
              socket = serversocket.accept();
              dis = new DataInputStream(socket.getInputStream());
              dos = new DataOutputStream(socket.getOutputStream());
              in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
              out = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()),true);
              input = in.readLine();
              ownerFrame107.list1.add("@user ("+socket.getInetAddress()+")-
-> "+input);
              if (input.equalsIgnoreCase("@owner-2")) {
                  input = this.ReceiveLongMessage(in);
                      byte[] inputbyte =
deBASE64tobyte.decodeBuffer(input);
                      System.out.println(input);
                      BigInteger rsabig = this.RSADecryption(new
BigInteger(inputbyte),"nd1.txt","d1.txt");
                      byte[] rsabyte = rsabig.toByteArray();
                      output = new String(rsabyte);
                      System.out.println("To User :"+output);
                      this.SendingLongMessage(output, out);
              }
              if (input.equalsIgnoreCase("@owner-1_&_owner-2")) {
                  input = this.ReceiveLongMessage(in);
                      byte[] inputbyte =
deBASE64tobyte.decodeBuffer(input);
                      System.out.println(input);
                      BigInteger rsabig = this.RSADecryption(new
BigInteger(inputbyte),"nd1.txt","d1a.txt");
```

```
                    byte[] rsabyte = rsabig.toByteArray();
                    output = enBASE64toString.encodeBuffer(rsabyte);
                    System.out.println("To Owner-1 :"+output);
                    this.SendingLongMessage(output, out);
            }
            if (input.equalsIgnoreCase("@DoKeyAgreement")) {
//                  ownerFrame107.list1.add("@user
("+socket.getInetAddress()+")--> "+input);
                this.KeyAgreement();
            }
            if (KAdone) {
                input = null;
                input = this.ReceiveLongMessage(in);
                String ownerinput = input;
                while (!(input.equalsIgnoreCase("@end-decryption"))) {
                    ownerFrame107.list1.add("@user
("+socket.getInetAddress()+")--> encrypted message");
                    System.out.println("From User : "+input);
                    System.out.println();
                    if ((input.equalsIgnoreCase("Owner 1") ) ||
                        (input.equalsIgnoreCase("Owner-1") ) ||
                        (input.equalsIgnoreCase("Owner-1_&_Owner-2") )) {
                        output = input;
                        ownerinput = input;
                        System.out.println("To User :"+output);
                        this.SendingLongMessage(output, out);
                    } else {
//                          byte[] inputbyte =
deBASE64tobyte.decodeBuffer(input);
//                          byte[] ciphertext =
this.SecretDecryption(inputbyte);
                        byte[] ciphertext =
deBASE64tobyte.decodeBuffer(input);
                        BigInteger rsabig = null;
                        if ((ownerinput.equalsIgnoreCase("Owner 1") ) ||
                            (ownerinput.equalsIgnoreCase("Owner-1") )) {
                            rsabig = this.RSADecryption(new
BigInteger(ciphertext),"nd1.txt","d1.txt");
                        }
                        if (ownerinput.equalsIgnoreCase("Owner-1_&_Owner-
2") ) {
                            rsabig = this.RSADecryption(new
BigInteger(ciphertext),"nd1.txt","d1a.txt");
                        }
                        byte[] rsabyte = rsabig.toByteArray();
                        byte[] outputbyte =
this.SecretEncryption(rsabyte);
                        output =
enBASE64toString.encodeBuffer(outputbyte);
                        System.out.println("To User :"+output);
                        this.SendingLongMessage(output, out);
                    }
                    input = this.ReceiveLongMessage(in);
                } // while
                System.out.println("From User : "+input);
            } else {
                System.out.println("Not KAdone");
            }
            socket.close();
            System.out.println("Socket close");
            input = null;
```

```
                output = null;
            } catch (Exception e) {
                System.out.println("Error...."+ e);
            }
        } // while

    } // constructor

    public void KeyAgreement() throws Exception {
        System.out.println("Generating Diffie-Hellman Key Pair ...");
        KeyPairGenerator keypairgenerator = 
  KeyPairGenerator.getInstance("DH");
        keypairgenerator.initialize(PARAMETER_SPEC);
        KeyPair keypair = keypairgenerator.genKeyPair();
        byte[] keybytes = keypair.getPublic().getEncoded();

        System.out.println("Sending server public key ...");
        dos.writeInt(keybytes.length);
        dos.write(keybytes);
        System.out.println("Done Sending server public key ...");

        System.out.println("Receiving Client's Public key ...");
        try {
          keybytes = new byte[dis.readInt()];
          dis.readFully(keybytes);
          System.out.println("Done Receiving Client's Public key ...");
        }catch (IOException ioe) {}

        KeyFactory keyfactory = KeyFactory.getInstance("DH");
        X509EncodedKeySpec x509spec = new X509EncodedKeySpec(keybytes);
        PublicKey clientpublickey = keyfactory.generatePublic(x509spec);

        System.out.println("Performing Agreement ...");
        KeyAgreement keyagreement = KeyAgreement.getInstance("DH");
        keyagreement.init(keypair.getPrivate());
        keyagreement.doPhase(clientpublickey,true);

        byte[] iv = new byte[8];
        SecureRandom securerandom = new SecureRandom();
        securerandom.nextBytes(iv);
        dos.write(iv);
        spec = new IvParameterSpec(iv); // this is the spec

        byte[] sessionkeybytes = keyagreement.generateSecret();
        SecretKeyFactory secretkeyfactory = 
  SecretKeyFactory.getInstance("DESede");
        DESedeKeySpec desedekeyspec = new DESedeKeySpec(sessionkeybytes);
        secretkey = secretkeyfactory.generateSecret(desedekeyspec); // this 
  is the secret key
        System.out.println("Done Performing Agreement ...");
  //      dos.close();
        KAdone = true;
  //      socket.close();
  //      System.out.println("Socket close");

    } // keyagreement

    public String ReceiveLongMessage(BufferedReader readin) {
        String text = "";
        String str1 = "";
        try {
            String str = readin.readLine();
            str1 = str1 + str;
            while (!((str==null)||(str.equalsIgnoreCase("@end")))) {
                str = readin.readLine();
                if (!((str==null)||(str.equalsIgnoreCase("@end")))) { str1 =
  str1 + str; }
            }
            text = str1;
        }catch (Exception exp) {}

        return text;
    } // ReceiveLongMessage

    public SecretKey getDESedekey(String filename) throws IOException {
      FileInputStream fis = new FileInputStream(filename);
      byte[] keybyte = new byte[fis.available()];
      fis.read(keybyte);
      SecretKeySpec mykey = new SecretKeySpec(keybyte,"DESede");
      fis.close();

      return mykey;
    }

    public byte[] SecretDecryption(byte[] inputbyte) throws Exception,
  IOException {
        SecretKey key;
        if (KAdone) { key = this.getSecretKey(); }
        else { key = this.getDESedekey("DESede.txt"); }
        cipher = Cipher.getInstance("DESede/ECB/PKCS5Padding");
        cipher.init(Cipher.DECRYPT_MODE, key);
        byte[] ciphertext = cipher.doFinal(inputbyte);

        return ciphertext;
    }

    public byte[] SecretEncryption(byte[] inputbyte) throws Exception,
  IOException {
        SecretKey key;
        if (KAdone) { key = this.getSecretKey(); }
        else { key = this.getDESedekey("DESede.txt"); }
  //      SecretKey key = this.getDESedekey("DESede.txt");
        cipher = Cipher.getInstance("DESede/ECB/PKCS5Padding");
        cipher.init(Cipher.ENCRYPT_MODE, key);
        byte[] ciphertext = cipher.doFinal(inputbyte);

        return ciphertext;
    }

    public String getdigest(String pass) throws Exception {
        String text = pass;
        System.out.println("Hashing...");
        MessageDigest md = MessageDigest.getInstance("MD5");
        byte[] strbyte = deBASE64tobyte.decodeBuffer(text);
  //      md.update(text.getBytes());
        md.update(strbyte);
        byte[] digest = md.digest();
        md.reset();
  //      String text1 = new String(digest);
        String text1 = enBASE64toString.encodeBuffer(digest);
        System.out.println("Done Hashing...");
```

```
//      return digest;
      return text1;
  }// getdigest

  public BigInteger getrsakeyimpl(String filename) throws Exception {
      FileInputStream fis = new FileInputStream(filename);
      byte[] keybytes = new byte[fis.available()];
      fis.read(keybytes);
      fis.close();
      BigInteger big= new BigInteger(keybytes);

      return big;
  }// getrsakeyimpl

  public RSAPublicKeyImpl getPublicKeyImpl(String mod, String expo) throws
Exception {
      BigInteger ne = this.getrsakeyimpl(mod);
      BigInteger e = this.getrsakeyimpl(expo);
      BigInteger phi = BigInteger.ONE;
      RSAPublicKeyImpl rsapublickey = new RSAPublicKeyImpl(ne, e, phi);

      return rsapublickey;
  }// getpublickey

  public RSAPrivateKeyImpl getPrivateKeyImpl(String mod, String expo)
throws Exception {
      BigInteger nd = this.getrsakeyimpl(mod);
      BigInteger d = this.getrsakeyimpl(expo);
      BigInteger phi = BigInteger.ONE;
      RSAPrivateKeyImpl rsaprivatekey = new RSAPrivateKeyImpl(nd, d, phi);

      return rsaprivatekey;
  }// getprivatekey

  public BigInteger RSADecryption(BigInteger textbig, String mod, String
expo) throws Exception {
      RSA rsa = new RSA();
      BigInteger decrypted = null;
      System.out.println("\nDecryption...");
      try {
          RSAPrivateKeyImpl rsaprivatekey =
this.getPrivateKeyImpl(mod,expo);
          decrypted = rsa.rsadp(rsaprivatekey,textbig);
          System.out.println("Done Encryption...");
      } catch (BadPaddingException bpe) {
          bpe.printStackTrace();
          throw new InvalidKeyException("Missing Crypto Algorithm");
      }
      return decrypted;
  } // RSADecryption

  public BigInteger RSAEncryption(BigInteger textbig, String mod, String
expo) throws Exception {
      RSA rsa = new RSA();
      BigInteger ciphertext = null;
      System.out.println("\nEncryption...");
      try {
          RSAPublicKeyImpl rsapublickey = this.getPublicKeyImpl(mod,expo);
          ciphertext = rsa.rsaep(rsapublickey,textbig);
          System.out.println("Done Encryption...");
      } catch (BadPaddingException bpe) {
          bpe.printStackTrace();
          throw new InvalidKeyException("Missing Crypto Algorithm");
      }
    return ciphertext;
  } // RSAEncryption

  public void SendingLongMessage(String text, PrintWriter admout) {
      try {
          int len = text.length();
          while (len>50) {
              String stext = text.substring(0,50);
              admout.println(stext);
              System.out.println(stext);
              text = text.substring(50);
              len = text.length();
          } //while
          admout.println(text);
          admout.println("@end");
          System.out.println(text);
      } catch (Exception exp) {}

  } // SendingLongMessage

  public void SendingToAdmin(String categori, String text, PrintWriter
admout) {
      BigInteger textbig;
      BigInteger textbigencrypted;
      byte[] textbyteencrypted;
      String texttosend;

      try {
          if (!(categori.equalsIgnoreCase("@doc-save"))) {
              textbig = new BigInteger(text.getBytes());
              textbigencrypted =
this.RSAEncryption(textbig,"n1.txt","e1.txt");
              textbyteencrypted = textbigencrypted.toByteArray();
              texttosend =
enBASE64toString.encodeBuffer(textbyteencrypted);
          } else {
              texttosend = text;
          }
          admout.println(categori);
          this.SendingLongMessage(texttosend, admout);
      } catch (Exception exp) {}
  }

  public void SendingHashToAdmin(String categori, String text, PrintWriter
admout) {
      try {
          StringTokenizer st = new StringTokenizer(text);
          while (st.hasMoreTokens()) {
              String stext = st.nextToken();
              int len = stext.length();
              if ((stext.substring(len-1).equalsIgnoreCase(","))||
                  (stext.substring(len-1).equalsIgnoreCase("."))) {
                  stext = stext.substring(0,len-1);
              }
              System.out.println(stext);
              if (!(stext.equalsIgnoreCase("and"))) {
                  String hashtext = this.getdigest(stext);
                  admout.println(categori);
```

```
                    admout.println(stext);
                    admout.println(hashtext);
                }
            } // while
        } catch (Exception exp) {}
    } /// SendingHashToAdmin

    public void OwnerSavingToAdmin(String text) {
        String hashtext = "";
        try {

            Socket admsocket = new Socket(adminhost, 6270);
            BufferedReader admin = new BufferedReader(new
InputStreamReader(admsocket.getInputStream()));
            PrintWriter admout = new PrintWriter(new
OutputStreamWriter(admsocket.getOutputStream()),true);

            admout.println(text);
            System.out.println(text);

            System.out.println("@author-save");
            String author = this.sendFrame107.AuthortextField.getText();
            this.SendingToAdmin("@author-save", author, admout);

            System.out.println("@title-save");
            String title = this.sendFrame107.TitletextField.getText();
            this.SendingToAdmin("@title-save",title, admout);

            System.out.println("@year-save");
            String year = this.sendFrame107.YeartextField.getText();
            this.SendingToAdmin("@year-save",year, admout);

            System.out.println("@doc-save");
            String doc = this.sendFrame107.Docchoice.getSelectedItem();
            this.SendingToAdmin("@doc-save",doc, admout);

            System.out.println("@author-hash");
            this.SendingHashToAdmin("@author-hash", author, admout);

            System.out.println("@title-hash");
            String keyword = this.sendFrame107.WordtextField.getText();
            this.SendingHashToAdmin("@title-hash", keyword, admout);

            System.out.println("@year-hash");
            this.SendingHashToAdmin("@year-hash", year, admout);

            System.out.println("@doc-hash");
            this.SendingHashToAdmin("@doc-hash", doc, admout);

            admout.println("@end");

            admsocket.close();
        } catch (Exception exp) {}

    } // OwnerSavingToAdmin

    public void OwnerReadFromAdmin(String input) throws Exception {
//      System.out.println("Beginning Communication ...");
        String text = "";
        String hashtext = "";
        Socket admsocket = new Socket(adminhost, 6270);
```

```
      BufferedReader admin = new BufferedReader(new
InputStreamReader(admsocket.getInputStream()));
      PrintWriter admout = new PrintWriter(new
OutputStreamWriter(admsocket.getOutputStream()),true);
      admout.println("@owner-read");
      hashtext = this.getdigest(input);
      admout.println(hashtext);
      text = admin.readLine();
      String ownertext = text;
      while (!(text.equalsIgnoreCase("@end"))) {
          if (!(text.equalsIgnoreCase("@not-found") )) {
              System.out.println(text);
              if ((text.equalsIgnoreCase("Owner 1") ) ||
                  (text.equalsIgnoreCase("Owner 2") ) ||
                  (text.equalsIgnoreCase("Owner-1") ) ||
                  (text.equalsIgnoreCase("Owner-2") ) ||
                  (text.equalsIgnoreCase("Owner-1_&_Owner-2") )) {
                  ownertext = text;
                  text = text;
              } else {
                  if ((ownertext.equalsIgnoreCase("Owner 1") ) ||
                      (ownertext.equalsIgnoreCase("Owner-1") )) {
                      byte[] textbyte = deBASE64tobyte.decodeBuffer(text);
                      BigInteger detextbig = this.RSADecryption(new
BigInteger(textbyte),"nd1.txt","d1.txt");
                      byte[] detextbyte = detextbig.toByteArray();
                      text = new String(detextbyte);
                  }
                  if ((ownertext.equalsIgnoreCase("Owner 2") ) ||
                      (ownertext.equalsIgnoreCase("Owner-2") )) {
                      Socket socket = new Socket("ikt02-16", 6276);
                      BufferedReader infromowner1 = new BufferedReader(new
InputStreamReader(socket.getInputStream())); // from owner
                      PrintWriter outtoowner1 = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()),true); // to owner
                      outtoowner1.println("@owner-1");
                      this.SendingLongMessage(text,outtoowner1);
                      text = this.ReceiveLongMessage(infromowner1);
                  }
                  if (ownertext.equalsIgnoreCase("Owner-1_&_Owner-2") ) {
                      byte[] textbyte = deBASE64tobyte.decodeBuffer(text);
                      BigInteger detextbig1 = this.RSADecryption(new
BigInteger(textbyte),"nd1.txt","d1a.txt");

                      Socket socket = new Socket("ikt02-16", 6276);
                      BufferedReader infromowner2 = new BufferedReader(new
InputStreamReader(socket.getInputStream())); // from owner
                      PrintWriter outtoowner2 = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()),true); // to owner
                      outtoowner2.println("@Owner-1_&_Owner-2");
                      System.out.println("To Owner 2   : "+text);
                      this.SendingLongMessage(text,outtoowner2);
                      String text2 = this.ReceiveLongMessage(infromowner2);
                      System.out.println("From Owner 2 : "+text2);
                      byte[] textbyte2 =
deBASE64tobyte.decodeBuffer(text2);

                      BigInteger detextbig2 = new BigInteger(textbyte2);
                      BigInteger n = this.getrsakeyimpl("nd1.txt");
                      BigInteger newbig =
detextbig1.multiply(detextbig2).mod(n);

                      text = new String(newbig.toByteArray());
                  }
              }
          }
          ownerFrame107.list1.add("@admin ("+admsocket.getInetAddress()+")
--> "+text);
          text = admin.readLine();
      } // while

  } // OwnerGetData

  public SecretKey getSecretKey() {
      return secretkey;
  }
  public IvParameterSpec getspec() {
      return spec;
  }
  public void sendMessage(String stringtosend) {
      out.println(stringtosend);
  }

  public String receiveMessage() throws Exception {
      return (in.readLine());
  }

} // Class AKAServer02

public class Owner107 {
  SecretKey secretkey;
  IvParameterSpec spec;
  String input = null;

  public Owner107() throws IOException, Exception {
      AKAServer01 akaserver = new AKAServer01();

  }
  public static void main(String[] args) throws IOException, Exception {
    Owner107 owner1071 = new Owner107();
  }
}
```

```
package owner107;

import java.awt.*;
import com.borland.jbcl.layout.*;
import java.awt.event.*;

/**
 * Title:
 * Description:
 * Copyright:    Copyright (c) 2002
 * Company:
 * @author
 * @version 1.0
 */

public class OwnerFrame107 extends Frame {
  Panel panel1 = new Panel();
  List list1 = new List();

  AKAServer01 akaserver01;
  Button SENDbutton = new Button();
  FlowLayout flowLayout1 = new FlowLayout();
  Button READbutton = new Button();
  Button DOWNbutton = new Button();
  Button CLEARbutton = new Button();

  public OwnerFrame107(AKAServer01 aka) {
  this.akaserver01 = aka;
    try {
      jbInit();
      this.setVisible(true);
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
//  public static void main(String[] args) {
//    OwnerFrame04 ownerFrame04 = new OwnerFrame04();
//  }
  private void jbInit() throws Exception {
    this.setSize(400,300);
    this.setTitle("Owner-1");
    this.addWindowListener(new java.awt.event.WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        this_windowClosing(e);
      }
    });
    panel1.setLayout(flowLayout1);
    SENDbutton.setLabel("SEND");
    SENDbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        SENDbutton_actionPerformed(e);
      }
    });
    READbutton.setLabel("READ");
    READbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        READbutton_actionPerformed(e);
      }
    });
    DOWNbutton.setLabel("DOWN");
    DOWNbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        DOWNbutton_actionPerformed(e);
      }
    });
    CLEARbutton.setLabel("CLEAR");
    CLEARbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        CLEARbutton_actionPerformed(e);
      }
    });
    this.add(list1, BorderLayout.CENTER);
    this.add(panel1, BorderLayout.SOUTH);
    panel1.add(CLEARbutton, null);
    panel1.add(SENDbutton, null);
    panel1.add(READbutton, null);
    panel1.add(DOWNbutton, null);
  }

  void DOWNbutton_actionPerformed(ActionEvent e) {
      System.exit(0);
  }

  void SENDbutton_actionPerformed(ActionEvent e) {
      akaserver01.sendFrame107.setVisible(true);
  }

  void READbutton_actionPerformed(ActionEvent e) {
      akaserver01.readFrame107.setVisible(true);
  }

  void CLEARbutton_actionPerformed(ActionEvent e) {
    this.list1.removeAll();
  }

  void this_windowClosing(WindowEvent e) {
      System.exit(0);
  }
}
```

```
package owner107;

import java.awt.*;
import com.borland.jbcl.layout.*;
import java.awt.event.*;

/**
 * Title:
 * Description:
 * Copyright:    Copyright (c) 2002
 * Company:
 * @author
 * @version 1.0
 */

public class ReadFrame107 extends Frame {
  AKAServer01 akaserver01;

  Label searchlabel = new Label();
  TextField textField = new TextField();
  VerticalFlowLayout verticalFlowLayout1 = new VerticalFlowLayout();
  Button Searchbutton = new Button();
  Label label1 = new Label();
  VerticalFlowLayout verticalFlowLayout2 = new VerticalFlowLayout();
  VerticalFlowLayout verticalFlowLayout3 = new VerticalFlowLayout();

  public ReadFrame107(AKAServer01 aka) {
    this.akaserver01 = aka;
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
//  public static void main(String[] args) {
//    ReadFrame04 readFrame04 = new ReadFrame04();
//  }
  private void jbInit() throws Exception {
  this.setSize(350, 175);
    this.setTitle("Read Frame Owner-1");
    this.addWindowListener(new java.awt.event.WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        this_windowClosing(e);
      }
    });
    Searchbutton.setLabel("BEGIN SEARCH");
    Searchbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        Searchbutton_actionPerformed(e);
      }
    });
    this.setLayout(verticalFlowLayout3);
    searchlabel.setText("Search Word");
    textField.setText("");
    label1.setText("You have to fill the textfield");
    this.add(searchlabel, null);
    this.add(Searchbutton, null);
    this.add(label1, null);
    this.add(textField, null);
  }
  void this_windowClosing(WindowEvent e) {
      this.setVisible(false);
  }

  void Searchbutton_actionPerformed(ActionEvent e) {
      String text = this.textField.getText();
      if (text.length()==0) {
          label1.setText("YOU HAVE TO FILL THE TEXTFIELD");
      } else {
      try {
          akaserver01.OwnerReadFromAdmin(text);
          label1.setText("You have to fill the textfield");
      } catch (Exception exc) {}
      }
  }
}
```

```
package owner107;

import java.awt.*;
import com.borland.jbcl.layout.*;
import java.awt.event.*;
import javax.swing.*;

/**
 * Title:
 * Description:
 * Copyright:    Copyright (c) 2002
 * Company:
 * @author
 * @version 1.0
 */

public class SendFrame107 extends Frame {
  AKAServer01 akaserver01;
  VerticalFlowLayout verticalFlowLayout1 = new VerticalFlowLayout();
  Label Wordlabel = new Label();
  TextField WordtextField = new TextField();
  Button SENDbutton = new Button();
  Label Authorlabel = new Label();
  TextField AuthortextField = new TextField();
  Label Titlelabel = new Label();
  TextField TitletextField = new TextField();
  Label Yearlabel = new Label();
  TextField YeartextField = new TextField();
  Label Documentlabel = new Label();
  Choice Docchoice = new Choice();
  Panel panel1 = new Panel();
  FlowLayout flowLayout1 = new FlowLayout();
  Button Cancellbutton = new Button();
  Button kEYWORDbutton = new Button();
  VerticalFlowLayout verticalFlowLayout2 = new VerticalFlowLayout();
  VerticalFlowLayout verticalFlowLayout3 = new VerticalFlowLayout();

  public SendFrame107(AKAServer01 aka) {
    this.akaserver01 = aka;
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
//  public static void main(String[] args) {
//    SendFrame04 sendFrame04 = new SendFrame04();
//  }
  private void jbInit() throws Exception {
    this.setSize(new Dimension(350, 355));
    this.setTitle("Send Frame Owner-1");
    this.addWindowListener(new java.awt.event.WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        this_windowClosing(e);
      }
    });
    Wordlabel.setFont(new java.awt.Font("Dialog", 1, 16));
    Wordlabel.setText("Key word for the title");
    this.setLayout(verticalFlowLayout3);
    SENDbutton.setLabel("SEND");
    SENDbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        SENDbutton_actionPerformed(e);
      }
    });
    WordtextField.setText("");
    Authorlabel.setText("Author");
    Titlelabel.setText("Title");
    Yearlabel.setText("Year");
    YeartextField.setText("2002");
    Documentlabel.setText("Owner");
    Docchoice.addItemListener(new java.awt.event.ItemListener() {
      public void itemStateChanged(ItemEvent e) {
        Docchoice_itemStateChanged(e);
      }
    });
    panel1.setLayout(flowLayout1);
    Cancellbutton.setLabel("CANCELL");
    Cancellbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        Cancellbutton_actionPerformed(e);
      }
    });
    kEYWORDbutton.setLabel("generate key word");
    kEYWORDbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        kEYWORDbutton_actionPerformed(e);
      }
    });
    Docchoice.add("Owner-1");
    Docchoice.add("Owner-1_&_Owner-2");
    panel1.add(kEYWORDbutton, null);
    panel1.add(SENDbutton, null);
    panel1.add(Cancellbutton, null);
    this.add(Authorlabel, null);
    this.add(AuthortextField, null);
    this.add(Titlelabel, null);
    this.add(TitletextField, null);
    this.add(Yearlabel, null);
    this.add(YeartextField, null);
    this.add(Documentlabel, null);
    this.add(Docchoice, null);
    this.add(Wordlabel, null);
    this.add(WordtextField, null);
    this.add(panel1, null);
  }

  void this_windowClosing(WindowEvent e) {
      this.setVisible(false);
  }

  void SENDbutton_actionPerformed(ActionEvent e) {
      if (!(WordtextField.getText().equalsIgnoreCase("")) ||
          !(AuthortextField.getText().equalsIgnoreCase("")) ||
          !(YeartextField.getText().equalsIgnoreCase("")) ) {
          akaserver01.OwnerSavingToAdmin("@owner-save");
      }
  }

  void Docchoice_itemStateChanged(ItemEvent e) {
      System.out.println(Docchoice.getSelectedItem());
  }

  void Cancellbutton_actionPerformed(ActionEvent e) {
      this.setVisible(false);
  }

  void kEYWORDbutton_actionPerformed(ActionEvent e) {
      this.WordtextField.setText(TitletextField.getText());
  }

}
```

Owner 2 (Key Agreement Implementation)

```
package owner207;

import java.io.*;
import java.net.*;
import java.math.*;
import java.security.*;
import java.security.spec.*;
import javax.crypto.*;
import javax.crypto.spec.*;
import javax.crypto.interfaces.*;
import sun.misc.*;
import java.util.*;
import risanuri01.*;

class AKAServer01 extends Thread {
  private static final byte SKIP_1024_MODULUS[] = {
   (byte)0xF4, (byte)0x88, (byte)0xFD, (byte)0x58,
   (byte)0x4E, (byte)0x49, (byte)0xDB, (byte)0xCD,
   (byte)0x20, (byte)0xB4, (byte)0x9D, (byte)0xE4,
   (byte)0x91, (byte)0x07, (byte)0x36, (byte)0x6B,
   (byte)0x33, (byte)0x6C, (byte)0x38, (byte)0x0D,
   (byte)0x45, (byte)0x1D, (byte)0x0F, (byte)0x7C,
   (byte)0x88, (byte)0xB3, (byte)0x1C, (byte)0x7C,
   (byte)0x5B, (byte)0x2D, (byte)0x8E, (byte)0xF6,
   (byte)0xF3, (byte)0xC9, (byte)0x23, (byte)0xC0,
   (byte)0x43, (byte)0xF0, (byte)0xA5, (byte)0x5B,
   (byte)0x18, (byte)0x8D, (byte)0x8E, (byte)0xBB,
   (byte)0x55, (byte)0x8C, (byte)0xB8, (byte)0x5D,
   (byte)0x38, (byte)0xD3, (byte)0x34, (byte)0xFD,
   (byte)0x7C, (byte)0x17, (byte)0x57, (byte)0x43,
   (byte)0xA3, (byte)0x1D, (byte)0x18, (byte)0x6C,
   (byte)0xDE, (byte)0x33, (byte)0x21, (byte)0x2C,
   (byte)0xB5, (byte)0x2A, (byte)0xFF, (byte)0x3C,
   (byte)0xE1, (byte)0xB1, (byte)0x29, (byte)0x40,
   (byte)0x18, (byte)0x11, (byte)0x8D, (byte)0x7C,
   (byte)0x84, (byte)0xA7, (byte)0x0A, (byte)0x72,
   (byte)0xD6, (byte)0x86, (byte)0xC4, (byte)0x03,
   (byte)0x19, (byte)0xC8, (byte)0x07, (byte)0x29,
   (byte)0x7A, (byte)0xCA, (byte)0x95, (byte)0x0C,
   (byte)0xD9, (byte)0x96, (byte)0x9F, (byte)0xAB,
   (byte)0xD0, (byte)0x0A, (byte)0x50, (byte)0x9B,
   (byte)0x02, (byte)0x46, (byte)0xD3, (byte)0x08,
   (byte)0x3D, (byte)0x66, (byte)0xA4, (byte)0x5D,
   (byte)0x41, (byte)0x9F, (byte)0x9C, (byte)0x7C,
   (byte)0xBD, (byte)0x89, (byte)0x4B, (byte)0x22,
   (byte)0x19, (byte)0x26, (byte)0xBA, (byte)0xAB,
   (byte)0xA2, (byte)0x5E, (byte)0xC3, (byte)0x55,
   (byte)0xE9, (byte)0x2F, (byte)0x78, (byte)0xC7
  };
  private static final BigInteger MODULUS = new
BigInteger(1,SKIP_1024_MODULUS);
  private static final BigInteger BASE = BigInteger.valueOf(2);
  private static final DHParameterSpec PARAMETER_SPEC = new
DHParameterSpec(MODULUS,BASE);

  String input = null;
  String output = null;
  String adminhost = "ikt02-16";
  SecretKey secretkey;
  IvParameterSpec spec;
  Cipher cipher;
  ServerSocket serversocket;
  Socket socket;
  int port = 6276;
  BufferedReader in;
  PrintWriter out;
  DataInputStream dis;
  DataOutputStream dos;
  BASE64Decoder deBASE64tobyte = new BASE64Decoder();
  BASE64Encoder enBASE64toString = new BASE64Encoder();
  boolean KAdone = false;
  OwnerFrame207 ownerFrame207 = new OwnerFrame207(this);
  ReadFrame207 readFrame207 = new ReadFrame207(this);
  SendFrame207 sendFrame207 = new SendFrame207(this);

  public AKAServer01()  throws Exception {
      this.start();
  }
  public void run() { //throws IOException { //, Exception {
      String str1 = "";
      try {
          serversocket = new ServerSocket(port);
          System.out.println("Owner-2 .. Listening on port "+port);
//          socket = serversocket.accept();
      } catch (Exception e) {
          System.out.println("Error creating server socket
...."+e.toString());
          System.exit(1);
      }

      while (true) {
          try {
              socket = serversocket.accept();
              dis = new DataInputStream(socket.getInputStream());
              dos = new DataOutputStream(socket.getOutputStream());
              in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
              out = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()),true);
              input = in.readLine();
              ownerFrame207.list1.add("@user ("+socket.getInetAddress()+")-
-> "+input);
              if (input.equalsIgnoreCase("@owner-1")) {
                  input = this.ReceiveLongMessage(in);
                      byte[] inputbyte =
deBASE64tobyte.decodeBuffer(input);
                      System.out.println(input);
                      BigInteger rsabig = this.RSADecryption(new
BigInteger(inputbyte),"nd2.txt","d2.txt");
                      byte[] rsabyte = rsabig.toByteArray();
                      output = new String(rsabyte);
                      System.out.println("To owner-1 :"+output);
                      this.SendingLongMessage(output, out);
              }
              if (input.equalsIgnoreCase("@owner-1_&_owner-2")) {
                  input = this.ReceiveLongMessage(in);
                      byte[] inputbyte =
deBASE64tobyte.decodeBuffer(input);
                      System.out.println(input);
                      BigInteger rsabig = this.RSADecryption(new
BigInteger(inputbyte),"nd1.txt","d1b.txt");
```

```
                        byte[] rsabyte = rsabig.toByteArray();
                        output = enBASE64toString.encodeBuffer(rsabyte);
                        System.out.println("To Owner-1 :"+output);
                        this.SendingLongMessage(output, out);
                }
                if (input.equalsIgnoreCase("@DoKeyAgreement")) {
//                    ownerFrame207.list1.add("@user
("+socket.getInetAddress()+")--> "+input);
                    this.KeyAgreement();
                }
                if (KAdone) {
                    input = null;
                    input = this.ReceiveLongMessage(in);
                    String ownerinput = input;
                    while (!(input.equalsIgnoreCase("@end-decryption"))) {
                        ownerFrame207.list1.add("@user
("+socket.getInetAddress()+")--> encrypted message");
                        System.out.println("From User : "+input);
                        System.out.println();
                        if ((input.equalsIgnoreCase("Owner 2") ) ||
                            (input.equalsIgnoreCase("Owner-2") ) ||
                            (input.equalsIgnoreCase("Owner-1_&_Owner-2") )) {
                            output = input;
                            ownerinput = input;
                            System.out.println("To User :"+output);
                            this.SendingLongMessage(output, out);
                        } else {
//                            byte[] inputbyte =
deBASE64tobyte.decodeBuffer(input);
//                            byte[] ciphertext =
this.SecretDecryption(inputbyte);
                            byte[] ciphertext =
deBASE64tobyte.decodeBuffer(input);
                            BigInteger rsabig = null;
                            if ((ownerinput.equalsIgnoreCase("Owner 2") ) ||
                                (ownerinput.equalsIgnoreCase("Owner-2") )) {
                                rsabig = this.RSADecryption(new
BigInteger(ciphertext),"nd2.txt","d2.txt");
                            }
                            if (ownerinput.equalsIgnoreCase("Owner-1_&_Owner-
2") ) {
                                rsabig = this.RSADecryption(new
BigInteger(ciphertext),"nd1.txt","d1b.txt");
                            }
                            byte[] rsabyte = rsabig.toByteArray();
                            byte[] outputbyte =
this.SecretEncryption(rsabyte);
                            output =
enBASE64toString.encodeBuffer(outputbyte);
                            System.out.println("To User :"+output);
                            this.SendingLongMessage(output, out);
                        }
                        input = this.ReceiveLongMessage(in);
                    } // while
                    System.out.println("From User : "+input);
                } else {
                    System.out.println("Not KAdone");
                }
                socket.close();
                System.out.println("Socket close");
                input = null;
                output = null;
            } catch (Exception e) {
                System.out.println("Error...."+ e);
            }
        } // while

    } // constructor

    public void KeyAgreement() throws Exception {
        System.out.println("Generating Diffie-Hellman Key Pair ...");
        KeyPairGenerator keypairgenerator =
  KeyPairGenerator.getInstance("DH");
        keypairgenerator.initialize(PARAMETER_SPEC);
        KeyPair keypair = keypairgenerator.genKeyPair();
        byte[] keybytes = keypair.getPublic().getEncoded();

        System.out.println("Sending server public key ...");
        dos.writeInt(keybytes.length);
        dos.write(keybytes);
        System.out.println("Done Sending server public key ...");

        System.out.println("Receiving Client's Public key ...");
        try {
          keybytes = new byte[dis.readInt()];
          dis.readFully(keybytes);
          System.out.println("Done Receiving Client's Public key ...");
        }catch (IOException ioe) {}

        KeyFactory keyfactory = KeyFactory.getInstance("DH");
        X509EncodedKeySpec x509spec = new X509EncodedKeySpec(keybytes);
        PublicKey clientpublickey = keyfactory.generatePublic(x509spec);

        System.out.println("Performing Agreement ...");
        KeyAgreement keyagreement = KeyAgreement.getInstance("DH");
        keyagreement.init(keypair.getPrivate());
        keyagreement.doPhase(clientpublickey,true);

        byte[] iv = new byte[8];
        SecureRandom securerandom = new SecureRandom();
        securerandom.nextBytes(iv);
        dos.write(iv);
        spec = new IvParameterSpec(iv); // this is the spec

        byte[] sessionkeybytes = keyagreement.generateSecret();
        SecretKeyFactory secretkeyfactory =
  SecretKeyFactory.getInstance("DESede");
        DESedeKeySpec desedekeyspec = new DESedeKeySpec(sessionkeybytes);
        secretkey = secretkeyfactory.generateSecret(desedekeyspec); // this
  is the secret key
        System.out.println("Done Performing Agreement ...");
  //      dos.close();
        KAdone = true;
  //      socket.close();
  //      System.out.println("Socket close");

    } // keyagreement

    public String ReceiveLongMessage(BufferedReader readin) {
        String text = "";
        String str1 = "";
        try {
            String str = readin.readLine();
            str1 = str1 + str;
            while (!((str==null)||(str.equalsIgnoreCase("@end")))) {
                str = readin.readLine();
                if (!((str==null)||(str.equalsIgnoreCase("@end")))) { str1 =
  str1 + str; }
            }
            text = str1;
        }catch (Exception exp) {}

        return text;
    } // ReceiveLongMessage

    public SecretKey getDESedekey(String filename) throws IOException {
      FileInputStream fis = new FileInputStream(filename);
      byte[] keybyte = new byte[fis.available()];
      fis.read(keybyte);
      SecretKeySpec mykey = new SecretKeySpec(keybyte,"DESede");
      fis.close();

      return mykey;
    }

    public byte[] SecretDecryption(byte[] inputbyte) throws Exception,
  IOException {
        SecretKey key;
        if (KAdone) { key = this.getSecretKey(); }
        else { key = this.getDESedekey("DESede.txt"); }
        cipher = Cipher.getInstance("DESede/ECB/PKCS5Padding");
        cipher.init(Cipher.DECRYPT_MODE, key);
        byte[] ciphertext = cipher.doFinal(inputbyte);

        return ciphertext;
    }

    public byte[] SecretEncryption(byte[] inputbyte) throws Exception,
  IOException {
        SecretKey key;
        if (KAdone) { key = this.getSecretKey(); }
        else { key = this.getDESedekey("DESede.txt"); }
  //      SecretKey key = this.getDESedekey("DESede.txt");
        cipher = Cipher.getInstance("DESede/ECB/PKCS5Padding");
        cipher.init(Cipher.ENCRYPT_MODE, key);
        byte[] ciphertext = cipher.doFinal(inputbyte);

        return ciphertext;
    }

    public String getdigest(String pass) throws Exception {
        String text = pass;
        System.out.println("Hashing...");
        MessageDigest md = MessageDigest.getInstance("MD5");
        byte[] strbyte = deBASE64tobyte.decodeBuffer(text);
  //      md.update(text.getBytes());
        md.update(strbyte);
        byte[] digest = md.digest();
        md.reset();
  //      String text1 = new String(digest);
        String text1 = enBASE64toString.encodeBuffer(digest);
        System.out.println("Done Hashing...");
```

```
//      return digest;
      return text1;
  }// getdigest

  public BigInteger getrsakeyimpl(String filename) throws Exception {
      FileInputStream fis = new FileInputStream(filename);
      byte[] keybytes = new byte[fis.available()];
      fis.read(keybytes);
      fis.close();
      BigInteger big= new BigInteger(keybytes);

      return big;
  }// getrsakeyimpl

  public RSAPublicKeyImpl getPublicKeyImpl(String mod, String expo) throws
Exception {
      BigInteger ne = this.getrsakeyimpl(mod);
      BigInteger e = this.getrsakeyimpl(expo);
      BigInteger phi = BigInteger.ONE;
      RSAPublicKeyImpl rsapublickey = new RSAPublicKeyImpl(ne, e, phi);

      return rsapublickey;
  }// getpublickey

  public RSAPrivateKeyImpl getPrivateKeyImpl(String mod, String expo)
throws Exception {
      BigInteger nd = this.getrsakeyimpl(mod);
      BigInteger d = this.getrsakeyimpl(expo);
      BigInteger phi = BigInteger.ONE;
      RSAPrivateKeyImpl rsaprivatekey = new RSAPrivateKeyImpl(nd, d, phi);

      return rsaprivatekey;
  }// getprivatekey

  public BigInteger RSADecryption(BigInteger textbig, String mod, String
expo) throws Exception {
      RSA rsa = new RSA();
      BigInteger decrypted = null;
      System.out.println("\nDecryption...");
      try {
          RSAPrivateKeyImpl rsaprivatekey = this.getPrivateKeyImpl(mod,
expo);
          decrypted = rsa.rsadp(rsaprivatekey,textbig);
          System.out.println("Done Encryption...");
      } catch (BadPaddingException bpe) {
          bpe.printStackTrace();
          throw new InvalidKeyException("Missing Crypto Algorithm");
      }
      return decrypted;
  } // RSADecryption

  public BigInteger RSAEncryption(BigInteger textbig, String mod, String
expo) throws Exception {
      RSA rsa = new RSA();
      BigInteger ciphertext = null;
      System.out.println("\nEncryption...");
      try {
          RSAPublicKeyImpl rsapublickey = this.getPublicKeyImpl(mod, expo);
          ciphertext = rsa.rsaep(rsapublickey,textbig);
          System.out.println("Done Encryption...");
      } catch (BadPaddingException bpe) {
          bpe.printStackTrace();
          throw new InvalidKeyException("Missing Crypto Algorithm");
      }
    return ciphertext;
  } // RSAEncryption

  public void SendingLongMessage(String text, PrintWriter admout) {
      try {
          int len = text.length();
          while (len>50) {
              String stext = text.substring(0,50);
              admout.println(stext);
              System.out.println(stext);
              text = text.substring(50);
              len = text.length();
          } //while
          admout.println(text);
          admout.println("@end");
          System.out.println(text);
      } catch (Exception exp) {}

  } // SendingLongMessage

  public void SendingToAdmin(String categori, String text, PrintWriter
admout) {
      BigInteger textbig;
      BigInteger textbigencrypted;
      byte[] textbyteencrypted;
      String texttosend;

      try {
          if (!(categori.equalsIgnoreCase("@doc-save"))) {
              textbig = new BigInteger(text.getBytes());
              textbigencrypted =
this.RSAEncryption(textbig,"n2.txt","e2.txt");
              textbyteencrypted = textbigencrypted.toByteArray();
              texttosend =
enBASE64toString.encodeBuffer(textbyteencrypted);
          } else {
              texttosend = text;
          }
          admout.println(categori);
          this.SendingLongMessage(texttosend, admout);
      } catch (Exception exp) {}
  }

  public void SendingHashToAdmin(String categori, String text, PrintWriter
admout) {
      try {
          StringTokenizer st = new StringTokenizer(text);
          while (st.hasMoreTokens()) {
              String stext = st.nextToken();
              int len = stext.length();
              if ((stext.substring(len-1).equalsIgnoreCase(","))||
                  (stext.substring(len-1).equalsIgnoreCase("."))) {
                  stext = stext.substring(0,len-1);
              }
              System.out.println(stext);
              if (!(stext.equalsIgnoreCase("and"))) {
                  String hashtext = this.getdigest(stext);
                  admout.println(categori);
                  admout.println(stext);
                  admout.println(hashtext);
              }
          } // while
      } catch (Exception exp) {}
  } /// SendingHashToAdmin

  public void OwnerSavingToAdmin(String text) {
      String hashtext = "";
      try {

          Socket admsocket = new Socket(adminhost, 6270);
          BufferedReader admin = new BufferedReader(new
InputStreamReader(admsocket.getInputStream()));
          PrintWriter admout = new PrintWriter(new
OutputStreamWriter(admsocket.getOutputStream()),true);

          admout.println(text);
          System.out.println(text);

          System.out.println("@author-save");
          String author = this.sendFrame207.AuthortextField.getText();
          this.SendingToAdmin("@author-save", author, admout);

          System.out.println("@title-save");
          String title = this.sendFrame207.TitletextField.getText();
          this.SendingToAdmin("@title-save",title, admout);

          System.out.println("@year-save");
          String year = this.sendFrame207.YeartextField.getText();
          this.SendingToAdmin("@year-save",year, admout);

          System.out.println("@doc-save");
          String doc = this.sendFrame207.Docchoice.getSelectedItem();
          this.SendingToAdmin("@doc-save",doc, admout);

          System.out.println("@author-hash");
          this.SendingHashToAdmin("@author-hash", author, admout);

          System.out.println("@title-hash");
          String keyword = this.sendFrame207.WordtextField.getText();
          this.SendingHashToAdmin("@title-hash", keyword, admout);

          System.out.println("@year-hash");
          this.SendingHashToAdmin("@year-hash", year, admout);

          System.out.println("@doc-hash");
          this.SendingHashToAdmin("@doc-hash", doc, admout);

          admout.println("@end");

          admsocket.close();
      } catch (Exception exp) {}

  } // OwnerSavingToAdmin

  public void OwnerReadFromAdmin(String input) throws Exception {
//      System.out.println("Beginning Communication ...");
      String text = "";
      String hashtext = "";
      Socket admsocket = new Socket(adminhost, 6270);
      BufferedReader admin = new BufferedReader(new
InputStreamReader(admsocket.getInputStream()));
      PrintWriter admout = new PrintWriter(new
OutputStreamWriter(admsocket.getOutputStream()),true);
      admout.println("@owner-read");
      hashtext = this.getdigest(input);
      admout.println(hashtext);
      text = admin.readLine();
      String ownertext = text;
      while (!(text.equalsIgnoreCase("@end"))) {
          if (!(text.equalsIgnoreCase("@not-found") )) {
              System.out.println(text);
              if ((text.equalsIgnoreCase("Owner 1") ) ||
                  (text.equalsIgnoreCase("Owner 2") ) ||
                  (text.equalsIgnoreCase("Owner-1") ) ||
                  (text.equalsIgnoreCase("Owner-2") ) ||
                  (text.equalsIgnoreCase("Owner-1_&_Owner-2") )) {
                  ownertext = text;
                  text = text;
              } else {
                  if ((ownertext.equalsIgnoreCase("Owner 2") ) ||
                      (ownertext.equalsIgnoreCase("Owner-2") )) {
                      byte[] textbyte = deBASE64tobyte.decodeBuffer(text);
                      BigInteger detextbig = this.RSADecryption(new
BigInteger(textbyte),"nd2.txt","d2.txt");
                      byte[] detextbyte = detextbig.toByteArray();
                      text = new String(detextbyte);
                  }
                  if ((ownertext.equalsIgnoreCase("Owner 1") ) ||
                      (ownertext.equalsIgnoreCase("Owner-1") )) {
                      Socket socket = new Socket("ikt02-16", 6275);
                      BufferedReader infromowner1 = new BufferedReader(new
InputStreamReader(socket.getInputStream())); // from owner
                      PrintWriter outtoowner1 = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()),true); // to owner
                      outtoowner1.println("@owner-2");
                      this.SendingLongMessage(text,outtoowner1);
                      text = this.ReceiveLongMessage(infromowner1);
                  }
                  if (ownertext.equalsIgnoreCase("Owner-1_&_Owner-2") ) {
                      byte[] textbyte = deBASE64tobyte.decodeBuffer(text);
                      BigInteger detextbig1 = this.RSADecryption(new
BigInteger(textbyte),"nd1.txt","d1b.txt");

                      Socket socket = new Socket("ikt02-16", 6275);
                      BufferedReader infromowner1 = new BufferedReader(new
InputStreamReader(socket.getInputStream())); // from owner
                      PrintWriter outtoowner1 = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()),true); // to owner
                      outtoowner1.println("@Owner-1_&_Owner-2");
                      System.out.println("To Owner 1   : "+text);
                      this.SendingLongMessage(text,outtoowner1);
                      String text2 = this.ReceiveLongMessage(infromowner1);
                      System.out.println("From Owner 1 : "+text2);
                      byte[] textbyte2 =
deBASE64tobyte.decodeBuffer(text2);

                      BigInteger detextbig2 = new BigInteger(textbyte2);
                      BigInteger n = this.getrsakeyimpl("nd1.txt");
                      BigInteger newbig =
detextbig1.multiply(detextbig2).mod(n);

                      text = new String(newbig.toByteArray());
                  }
              }
          }
          ownerFrame207.list1.add("@admin ("+admsocket.getInetAddress()+")
--> "+text);
          text = admin.readLine();
      } // while

  } // OwnerGetData

  public SecretKey getSecretKey() {
      return secretkey;
  }
  public IvParameterSpec getspec() {
      return spec;
  }
  public void sendMessage(String stringtosend) {
      out.println(stringtosend);
  }

  public String receiveMessage() throws Exception {
      return (in.readLine());
  }

} // Class AKAServer02

public class Owner207 {
  SecretKey secretkey;
  IvParameterSpec spec;
  String input = null;

  public Owner207() throws IOException, Exception {
      AKAServer01 akaserver = new AKAServer01();

  }
  public static void main(String[] args) throws IOException, Exception {
    Owner207 owner2071 = new Owner207();
  }
}
```

```
package owner207;

import java.awt.*;
import com.borland.jbcl.layout.*;
import java.awt.event.*;

/**
 * Title:
 * Description:
 * Copyright:    Copyright (c) 2002
 * Company:
 * @author
 * @version 1.0
 */

public class OwnerFrame207 extends Frame {
  Panel panel1 = new Panel();
  List list1 = new List();

  AKAServer01 akaserver01;
  Button SENDbutton = new Button();
  FlowLayout flowLayout1 = new FlowLayout();
  Button READbutton = new Button();
  Button DOWNbutton = new Button();
  Button CLEARbutton = new Button();

  public OwnerFrame207(AKAServer01 aka) {
  this.akaserver01 = aka;
    try {
      jbInit();
      this.setVisible(true);
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
//  public static void main(String[] args) {
//    OwnerFrame04 ownerFrame04 = new OwnerFrame04();
//  }
  private void jbInit() throws Exception {
    this.setSize(400,300);
    this.setTitle("Owner-2");
    this.addWindowListener(new java.awt.event.WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        this_windowClosing(e);
      }
    });
    panel1.setLayout(flowLayout1);
    SENDbutton.setLabel("SEND");
    SENDbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        SENDbutton_actionPerformed(e);
      }
    });
    READbutton.setLabel("READ");
    READbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        READbutton_actionPerformed(e);
      }
    });
    DOWNbutton.setLabel("DOWN");
    DOWNbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        DOWNbutton_actionPerformed(e);
      }
    });
    CLEARbutton.setLabel("CLEAR");
    CLEARbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        CLEARbutton_actionPerformed(e);
      }
    });
    this.add(list1, BorderLayout.CENTER);
    this.add(panel1, BorderLayout.SOUTH);
    panel1.add(CLEARbutton, null);
    panel1.add(SENDbutton, null);
    panel1.add(READbutton, null);
    panel1.add(DOWNbutton, null);
  }

  void DOWNbutton_actionPerformed(ActionEvent e) {
      System.exit(0);
  }

  void SENDbutton_actionPerformed(ActionEvent e) {
      akaserver01.sendFrame207.setVisible(true);
  }

  void READbutton_actionPerformed(ActionEvent e) {
      akaserver01.readFrame207.setVisible(true);
  }

  void CLEARbutton_actionPerformed(ActionEvent e) {
    this.list1.removeAll();
  }

  void this_windowClosing(WindowEvent e) {
      System.exit(0);
  }
}
```

```
package owner207;

import java.awt.*;
import com.borland.jbcl.layout.*;
import java.awt.event.*;

/**
 * Title:
 * Description:
 * Copyright:    Copyright (c) 2002
 * Company:
 * @author
 * @version 1.0
 */

public class ReadFrame207 extends Frame {
  AKAServer01 akaserver01;

  Label searchlabel = new Label();
  TextField textField = new TextField();
  VerticalFlowLayout verticalFlowLayout1 = new VerticalFlowLayout();
  Button Searchbutton = new Button();
  Label label1 = new Label();
  VerticalFlowLayout verticalFlowLayout2 = new VerticalFlowLayout();
  VerticalFlowLayout verticalFlowLayout3 = new VerticalFlowLayout();
  VerticalFlowLayout verticalFlowLayout4 = new VerticalFlowLayout();
  VerticalFlowLayout verticalFlowLayout5 = new VerticalFlowLayout();

  public ReadFrame207(AKAServer01 aka) {
    this.akaserver01 = aka;
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
//  public static void main(String[] args) {
//    ReadFrame04 readFrame04 = new ReadFrame04();
//  }
  private void jbInit() throws Exception {
  this.setSize(350, 175);
    this.setTitle("Read Frame Owner-2");
    this.addWindowListener(new java.awt.event.WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        this_windowClosing(e);
      }
    });
    Searchbutton.setLabel("BEGIN SEARCH");
    Searchbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        Searchbutton_actionPerformed(e);
      }
    });
    this.setLayout(verticalFlowLayout5);
    searchlabel.setText("Search Word");
    textField.setText("");
    label1.setText("You have to fill the textfield");
    this.add(searchlabel, null);
    this.add(Searchbutton, null);
    this.add(label1, null);
    this.add(textField, null);
  }

  void this_windowClosing(WindowEvent e) {
      this.setVisible(false);
  }

  void Searchbutton_actionPerformed(ActionEvent e) {
      String text = this.textField.getText();
      if (text.length()==0) {
          label1.setText("YOU HAVE TO FILL THE TEXTFIELD");
      } else {
      try {
          akaserver01.OwnerReadFromAdmin(text);
          label1.setText("You have to fill the textfield");
      } catch (Exception exc) {}
      }
  }
}
```

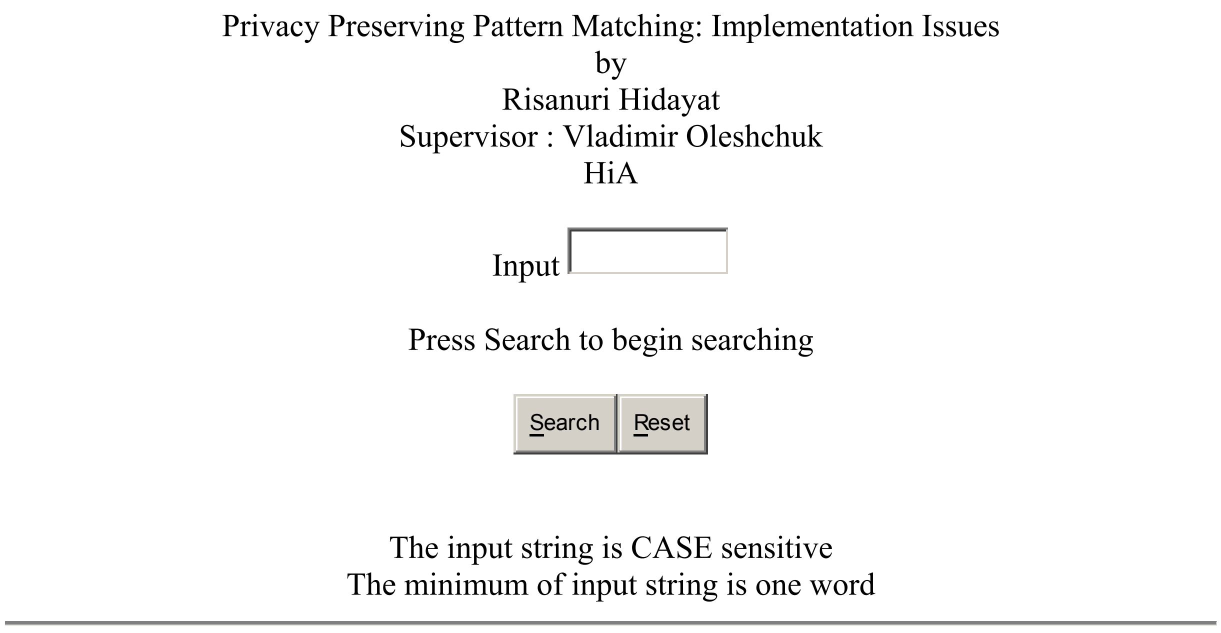
```
package owner207;

import java.awt.*;
import com.borland.jbcl.layout.*;
import java.awt.event.*;
import javax.swing.*;

/**
 * Title:
 * Description:
 * Copyright:    Copyright (c) 2002
 * Company:
 * @author
 * @version 1.0
 */

public class SendFrame207 extends Frame {
  AKAServer01 akaserver01;
  VerticalFlowLayout verticalFlowLayout1 = new VerticalFlowLayout();
  Label Wordlabel = new Label();
  TextField WordtextField = new TextField();
  Button SENDbutton = new Button();
  Label Authorlabel = new Label();
  TextField AuthortextField = new TextField();
  Label Titlelabel = new Label();
  TextField TitletextField = new TextField();
  Label Yearlabel = new Label();
  TextField YeartextField = new TextField();
  Label Documentlabel = new Label();
  Choice Docchoice = new Choice();
  Panel panel1 = new Panel();
  FlowLayout flowLayout1 = new FlowLayout();
  Button Cancellbutton = new Button();
  Button kEYWORDbutton = new Button();
  VerticalFlowLayout verticalFlowLayout2 = new VerticalFlowLayout();

  public SendFrame207(AKAServer01 aka) {
    this.akaserver01 = aka;
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
//  public static void main(String[] args) {
//    SendFrame04 sendFrame04 = new SendFrame04();
//  }
  private void jbInit() throws Exception {
    this.setSize(new Dimension(350, 355));
    this.setTitle("Send Frame Owner-2");
    this.addWindowListener(new java.awt.event.WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        this_windowClosing(e);
      }
    });
    Wordlabel.setFont(new java.awt.Font("Dialog", 1, 16));
    Wordlabel.setText("Key word for the title");
    this.setLayout(verticalFlowLayout2);
    SENDbutton.setLabel("SEND");
    SENDbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        SENDbutton_actionPerformed(e);
      }
    });
    WordtextField.setText("");
    Authorlabel.setText("Author");
    Titlelabel.setText("Title");
    Yearlabel.setText("Year");
    YeartextField.setText("2002");
    Documentlabel.setText("Owner");
    Docchoice.addItemListener(new java.awt.event.ItemListener() {
      public void itemStateChanged(ItemEvent e) {
        Docchoice_itemStateChanged(e);
      }
    });
    panel1.setLayout(flowLayout1);
    Cancellbutton.setLabel("CANCELL");
    Cancellbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        Cancellbutton_actionPerformed(e);
      }
    });
    kEYWORDbutton.setLabel("generate key word");
    kEYWORDbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        kEYWORDbutton_actionPerformed(e);
      }
    });
    Docchoice.add("Owner-2");
    this.add(Authorlabel, null);
    this.add(AuthortextField, null);
    this.add(Titlelabel, null);
    this.add(TitletextField, null);
    this.add(Yearlabel, null);
    this.add(YeartextField, null);
    this.add(Documentlabel, null);
    this.add(Docchoice, null);
    this.add(Wordlabel, null);
    this.add(WordtextField, null);
    this.add(panel1, null);
    panel1.add(kEYWORDbutton, null);
    panel1.add(SENDbutton, null);
    panel1.add(Cancellbutton, null);
  }

  void this_windowClosing(WindowEvent e) {
      this.setVisible(false);
  }

  void SENDbutton_actionPerformed(ActionEvent e) {
      akaserver01.OwnerSavingToAdmin("@owner-save");
//        this.Hashlabel.setText("Hashed : "+akaserver01.hashstring);
//        this.Encryptedlabel.setText("Encrypted :
"+akaserver01.hashstring.length());
  }

  void Docchoice_itemStateChanged(ItemEvent e) {
      System.out.println(Docchoice.getSelectedItem());
  }

  void Cancellbutton_actionPerformed(ActionEvent e) {
      this.setVisible(false);
  }

  void kEYWORDbutton_actionPerformed(ActionEvent e) {
      this.WordtextField.setText(TitletextField.getText());
  }

}
```

User (Key Agreement Implementation)

```
package servlet07;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.net.*;
import java.security.*;
import javax.crypto.*;
import java.security.spec.*;
import sun.misc.*;
import java.math.*;
import javax.crypto.spec.*;
import javax.crypto.interfaces.*;

class AKAClient01 {
  private static final byte SKIP_1024_MODULUS[] = {
   (byte)0xF4, (byte)0x88, (byte)0xFD, (byte)0x58,
   (byte)0x4E, (byte)0x49, (byte)0xDB, (byte)0xCD,
   (byte)0x20, (byte)0xB4, (byte)0x9D, (byte)0xE4,
   (byte)0x91, (byte)0x07, (byte)0x36, (byte)0x6B,
   (byte)0x33, (byte)0x6C, (byte)0x38, (byte)0x0D,
   (byte)0x45, (byte)0x1D, (byte)0x0F, (byte)0x7C,
   (byte)0x88, (byte)0xB3, (byte)0x1C, (byte)0x7C,
   (byte)0x5B, (byte)0x2D, (byte)0x8E, (byte)0xF6,
   (byte)0xF3, (byte)0xC9, (byte)0x23, (byte)0xC0,
   (byte)0x43, (byte)0xF0, (byte)0xA5, (byte)0x5B,
   (byte)0x18, (byte)0x8D, (byte)0x8E, (byte)0xBB,
   (byte)0x55, (byte)0x8C, (byte)0xB8, (byte)0x5D,
   (byte)0x38, (byte)0xD3, (byte)0x34, (byte)0xFD,
   (byte)0x7C, (byte)0x17, (byte)0x57, (byte)0x43,
   (byte)0xA3, (byte)0x1D, (byte)0x18, (byte)0x6C,
   (byte)0xDE, (byte)0x33, (byte)0x21, (byte)0x2C,
   (byte)0xB5, (byte)0x2A, (byte)0xFF, (byte)0x3C,
   (byte)0xE1, (byte)0xB1, (byte)0x29, (byte)0x40,
   (byte)0x18, (byte)0x11, (byte)0x8D, (byte)0x7C,
   (byte)0x84, (byte)0xA7, (byte)0x0A, (byte)0x72,
   (byte)0xD6, (byte)0x86, (byte)0xC4, (byte)0x03,
   (byte)0x19, (byte)0xC8, (byte)0x07, (byte)0x29,
   (byte)0x7A, (byte)0xCA, (byte)0x95, (byte)0x0C,
   (byte)0xD9, (byte)0x96, (byte)0x9F, (byte)0xAB,
   (byte)0xD0, (byte)0x0A, (byte)0x50, (byte)0x9B,
   (byte)0x02, (byte)0x46, (byte)0xD3, (byte)0x08,
   (byte)0x3D, (byte)0x66, (byte)0xA4, (byte)0x5D,
   (byte)0x41, (byte)0x9F, (byte)0x9C, (byte)0x7C,
   (byte)0xBD, (byte)0x89, (byte)0x4B, (byte)0x22,
   (byte)0x19, (byte)0x26, (byte)0xBA, (byte)0xAB,
   (byte)0xA2, (byte)0x5E, (byte)0xC3, (byte)0x55,
   (byte)0xE9, (byte)0x2F, (byte)0x78, (byte)0xC7
   };
  private static final BigInteger MODULUS = new
BigInteger(1,SKIP_1024_MODULUS);
  private static final BigInteger BASE = BigInteger.valueOf(2);
  private static final DHParameterSpec PARAMETER_SPEC = new
DHParameterSpec(MODULUS,BASE);

  Socket socket;
  SecretKey secretkey;
  IvParameterSpec spec;
  String ownerhost; // = "ikt02-16";
  int ownerport; // = 6275;
  String input = null;
  BufferedReader in;
  PrintWriter out;
      DataOutputStream dos;
      DataInputStream dis;
  BASE64Encoder enBASE64toString = new BASE64Encoder();
  BASE64Decoder deBASE64tobyte = new BASE64Decoder();
  Cipher cipher;
  boolean KAdone = false;

  public AKAClient01(String host, int port) throws Exception {
      this.ownerhost = host;
      this.ownerport = port;
      Socket socket = new Socket(ownerhost, ownerport);
      in = new BufferedReader(new
InputStreamReader(socket.getInputStream())); // from owner
      out = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()),true); // to owner
       dos = new DataOutputStream(socket.getOutputStream());
       dis = new DataInputStream(socket.getInputStream());
      this.sendMessage("@DoKeyAgreement");
      this.KeyAgreement();

  }// constructor AKAclient

  public void KeyAgreement() throws Exception {
//      DataOutputStream dos;
//      DataInputStream dis;
//       dos = new DataOutputStream(socket.getOutputStream());
//       dis = new DataInputStream(socket.getInputStream());

      System.out.println("Receiving server public key ...");
      byte[] keybytes = new byte[dis.readInt()];
      dis.readFully(keybytes);
      System.out.println("Done Receiving server public key ...");

      System.out.println("Generating Diffie-Hellman Key Pair ...");
      KeyPairGenerator keypairgenerator =
KeyPairGenerator.getInstance("DH");
      keypairgenerator.initialize(PARAMETER_SPEC);
      KeyPair keypair = keypairgenerator.genKeyPair();
      System.out.println("Done Generating Diffie-Hellman Key Pair ...");


      KeyFactory keyfactory = KeyFactory.getInstance("DH");
      X509EncodedKeySpec x509spec = new X509EncodedKeySpec(keybytes);
      PublicKey serverpublickey = keyfactory.generatePublic(x509spec);

      System.out.println("Sending Client's Public key ...");

      keybytes = keypair.getPublic().getEncoded();
      dos.writeInt(keybytes.length);
      dos.write(keybytes);

      System.out.println("Performing Agreement ...");
      KeyAgreement keyagreement = KeyAgreement.getInstance("DH");
      keyagreement.init(keypair.getPrivate());
      keyagreement.doPhase(serverpublickey,true);

      byte[] iv = new byte[8];
      dis.readFully(iv);
      spec = new IvParameterSpec(iv);

      byte[] sessionkeybytes = keyagreement.generateSecret();
      SecretKeyFactory secretkeyfactory =
SecretKeyFactory.getInstance("DESede");
      DESedeKeySpec desedekeyspec = new DESedeKeySpec(sessionkeybytes);
      secretkey = secretkeyfactory.generateSecret(desedekeyspec);
      KAdone = true;
      System.out.println("Done Performing Agreement ...");
//      dis.close();
//      dos.close();
  } // keyagreement

  public SecretKey getDESedekey(String filename) throws IOException {
    String text = "";
    FileInputStream fis = new FileInputStream(filename);
    byte[] keybyte = new byte[fis.available()];
    fis.read(keybyte);
    SecretKeySpec mykey = new SecretKeySpec(keybyte,"DESede");
    fis.close();

    return mykey;
  }

  public byte[] SecretDecryption(byte[] inputbyte) throws Exception,
IOException {
      SecretKey key;
      if (KAdone) { key = this.getSecretKey(); }
      else { key = this.getDESedekey("DESede.txt"); }
//      SecretKey key = this.getDESedekey("DESede.txt");
      cipher = Cipher.getInstance("DESede/ECB/PKCS5Padding");
      cipher.init(Cipher.DECRYPT_MODE, key);
      byte[] ciphertext = cipher.doFinal(inputbyte);

      return ciphertext;
  }

  public byte[] SecretEncryption(byte[] inputbyte) throws Exception,
IOException {
      SecretKey key;
      if (KAdone) { key = this.getSecretKey(); }
      else { key = this.getDESedekey("DESede.txt"); }
//      SecretKey key = this.getDESedekey("DESede.txt");
      cipher = Cipher.getInstance("DESede/ECB/PKCS5Padding");
      cipher.init(Cipher.ENCRYPT_MODE, key);
      byte[] ciphertext = cipher.doFinal(inputbyte);

      return ciphertext;
  }

  public void SendingLongMessage(String text, PrintWriter admout) {
      try {
          int len = text.length();
          while (len>50) {
              String stext = text.substring(0,50);
              admout.println(stext);
              text = text.substring(50);
              len = text.length();
          } //while
          admout.println(text);
```

```
            admout.println("@end");
        } catch (Exception exp) {}

    } // SendingLongMessage

    public String ReceiveLongMessage() {
        String text = "";
        String str1 = "";
        try {
            String str = this.receiveMessage();
            str1 = str1 + str;
            while (!((str==null)||(str.equalsIgnoreCase("@end")))) {
                str = this.receiveMessage();
                if (!((str==null)||(str.equalsIgnoreCase("@end")))) { str1 =
str1 + str; }
            }
            text = str1;
        }catch (Exception exp) {}

        return text;
    }

    public String UserAskDecryptionToOwner(String text) throws Exception {
        String str1 = "";
        String outputtext = null;
        String stringtosend = null;

        System.out.println("From Admin : "+text);
        System.out.println();
        if ((text.equalsIgnoreCase("Owner 1") ) ||
            (text.equalsIgnoreCase("Owner 2") ) ||
            (text.equalsIgnoreCase("Owner-1") ) ||
            (text.equalsIgnoreCase("Owner-2") ) ||
            (text.equalsIgnoreCase("Owner-1_&_Owner-2") )) {
            stringtosend = text;
        } else {
//          byte[] textbyte = deBASE64tobyte.decodeBuffer(text);
//          byte[] ciphertext = this.SecretEncryption(textbyte);

//          stringtosend = enBASE64toString.encodeBuffer(ciphertext);
            stringtosend = text;
        }
        System.out.println("To Owner :"+stringtosend);
        this.SendingLongMessage(stringtosend,out);
        str1 = this.ReceiveLongMessage();
        System.out.println();
        System.out.println("From Owner : "+str1);
        if ((str1.equalsIgnoreCase("Owner 1") ) ||
            (str1.equalsIgnoreCase("Owner 2") ) ||
            (str1.equalsIgnoreCase("Owner-1") ) ||
            (str1.equalsIgnoreCase("Owner-2") ) ||
            (str1.equalsIgnoreCase("Owner-1_&_Owner-2") )) {
            outputtext = enBASE64toString.encodeBuffer(str1.getBytes());
        } else {
            byte[] outputbyte = deBASE64tobyte.decodeBuffer(str1);
            byte[] DESbyte = this.SecretDecryption(outputbyte);
            outputtext = enBASE64toString.encodeBuffer(DESbyte);
        }
        System.out.println(outputtext);

        return outputtext;
```

```
  }//end getData

  public SecretKey getSecretKey() {
      return secretkey;
  }
  public IvParameterSpec getspec() {
      return spec;
  }

  public void sendMessage(String stringtosend) {
      out.println(stringtosend);
  }

  public String receiveMessage() throws Exception {
      return (in.readLine());
  }

} // class AKAClient

class Client02{
    String adminhost = "ikt02-16";
    int port = 6270; //echo port
    String outputtext = null;
    Socket socket;
    BufferedReader in;
    PrintWriter out;
    BASE64Decoder deBASE64tobyte = new BASE64Decoder();
    BASE64Encoder enBASE64toString = new BASE64Encoder();
    AKAClient01 toowner1 = null;
    AKAClient01 toowner2 = null;

  public Client02() throws Exception {
    try{
      Socket socket = new Socket(adminhost,port);
      in = new BufferedReader(new
InputStreamReader(socket.getInputStream())); // from admin
      out = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()),true); // to admin
    }//end try
    catch(UnknownHostException e){
      System.out.println(e);
      System.out.println("Must be online to run properly.");
    }//end catch UnknownHostException
    catch(IOException e){System.out.println(e);}
  } //Constructor Client02

  public void sendMessage(String string) {
      out.println(string);
  }

  public String receiveMessage() throws Exception {
      return (in.readLine());
  }

  public void SendingHashToAdmin(String text) {
      try {
          StringTokenizer st = new StringTokenizer(text);
          while (st.hasMoreTokens()) {
              String stext = st.nextToken();
              int len = stext.length();
              if ((stext.substring(len-1).equalsIgnoreCase(","))||
                  (stext.substring(len-1).equalsIgnoreCase("."))) {
                  stext = stext.substring(0,len-1);
              }
              System.out.println(stext);
              if (!(stext.equalsIgnoreCase("and"))) {
                  String hashtext = this.getdigest(stext);
                  out.println(hashtext);
              }
          } // while
          out.println("@end-from-user");
      } catch (Exception exp) {}
  } /// SendingHashToAdmin


  public void UsergetDataFromAdmin(String var0, PrintWriter htmlout) throws
Exception {
    String entext = "";
    String detext = "";
    String text = "";
    int i = 1;

    try {
      this.sendMessage("@user");
      this.SendingHashToAdmin(var0);
//        text = client.getdigest(var0);
//        this.sendMessage(text);
      entext = this.receiveMessage();
      String ownertext = entext;
      if (!(entext.equalsIgnoreCase("@end-decryption"))) {
          if ((ownertext.equalsIgnoreCase("Owner 1") ) ||
              (ownertext.equalsIgnoreCase("Owner-1") )) {
              String host = "ikt02-16"; // owner-1
              int port = 6275; // owner-1
              toowner1 = new AKAClient01(host,port);
              htmlout.println("Asking decryption to Owner 1...");
          }
          if ((ownertext.equalsIgnoreCase("Owner 2") ) ||
              (ownertext.equalsIgnoreCase("Owner-2") )) {
              String host = "ikt02-16"; // owner-2
              int port = 6276; // owner-2
              toowner1 = new AKAClient01(host,port);
              htmlout.println("Asking decryption to Owner 2...");
          }
          if (ownertext.equalsIgnoreCase("Owner-1_&_Owner-2")) {
              String host1 = "ikt02-16"; // owner-1
              String host2 = "ikt02-16"; // owner-2
              int port1 = 6275; // owner-1
              int port2 = 6276; // owner-2
              toowner1 = new AKAClient01(host1,port1);
              toowner2 = new AKAClient01(host2,port2);
              htmlout.println("Asking decryption to Owner 1 and Owner 2
...");
          }
          while (!(entext.equalsIgnoreCase("@end-decryption"))) {
              if (i%5==1) {
                  htmlout.println("</tr>");
                  htmlout.println("<tr>");
                  htmlout.println("<td>" + ((i+5)/5) + "</td>");i++;
              }
              if ((ownertext.equalsIgnoreCase("Owner 1") ) ||
                  (ownertext.equalsIgnoreCase("Owner-1") ) ||
                  (ownertext.equalsIgnoreCase("Owner 2") ) ||
                  (ownertext.equalsIgnoreCase("Owner-2") )) {
                  String detext1 =
toowner1.UserAskDecryptionToOwner(entext);
                  byte[] detextbyte = deBASE64tobyte.decodeBuffer(detext1);
                  detext = new String(detextbyte);
                  htmlout.println("<td>" + detext + "</td>");
                  entext = this.receiveMessage(); i++;
              }
              if ((ownertext.equalsIgnoreCase("Owner-1_&_Owner-2")) ) {
                  System.out.println("With Owner-1 ***");
                  String detext1 =
toowner1.UserAskDecryptionToOwner(entext);
                  byte[] detextbyte1 =
deBASE64tobyte.decodeBuffer(detext1);

                  System.out.println("With Owner-2 ***");
                  String detext2 =
toowner2.UserAskDecryptionToOwner(entext);
                  byte[] detextbyte2 =
deBASE64tobyte.decodeBuffer(detext2);
                  if ((new String(detextbyte1).equalsIgnoreCase("Owner-
1_&_Owner-2")) ||
                      (new String(detextbyte2).equalsIgnoreCase("Owner-
1_&_Owner-2")) ) {
                      detext = new String(detextbyte1);
                  } else {
                      BigInteger detextbig1 = new BigInteger(detextbyte1);
                      BigInteger detextbig2 = new BigInteger(detextbyte2);
                      BigInteger n = this.getrsakeyimpl("nd1.txt");
                      BigInteger newbig =
detextbig1.multiply(detextbig2).mod(n);
                      detext = new String(newbig.toByteArray());
                  }

                  htmlout.println("<td>" + detext + "</td>");
                  entext = this.receiveMessage(); i++;
              }
          } // while
          htmlout.println("</tr>");
          this.toowner1.SendingLongMessage("@end-decryption",toowner1.out);
          if ((ownertext.equalsIgnoreCase("Owner-1_&_Owner-2")) ) {
              this.toowner2.SendingLongMessage("@end-
decryption",toowner2.out);
          }

          htmlout.println(entext);
      } else {
          htmlout.println("Data is Not Found");
      }
    }
    catch (Exception e) {
      System.out.println(e.toString());
    }

  }//end UsergetDataFromAdmin

  public BigInteger getrsakeyimpl(String filename) throws Exception {
      FileInputStream fis = new FileInputStream(filename);
      byte[] keybytes = new byte[fis.available()];
      fis.read(keybytes);
      fis.close();
      BigInteger big= new BigInteger(keybytes);

      return big;
  }// getrsakeyimpl

  public String getdigest(String pass) throws Exception {
      String text = pass;
      MessageDigest md = MessageDigest.getInstance("MD5");
      byte[] strbyte = deBASE64tobyte.decodeBuffer(text);
//      md.update(text.getBytes());
      md.update(strbyte);
      byte[] digest = md.digest();
      md.reset();
//      String text1 = new String(digest);
      String text1 = enBASE64toString.encodeBuffer(digest);

//      return digest;
      return text1;
  } // getdigest

}//end class Client02

public class Servlet1 extends HttpServlet {
  private static final String CONTENT_TYPE = "text/html";
  String input = null;
  Client02 client;

//  AKAClient01 akaclient;

  /**Initialize global variables*/
  public void init() throws ServletException {
  }
  /**Process the HTTP Get request*/
  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    String var0 = "";
    try {
      var0 = request.getParameter("param0");
    }
    catch(Exception e) {
      e.printStackTrace();
    }
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    out.println("<font color=\"green\">");
    out.println("</font>");
  }
  /**Process the HTTP Post request*/
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    String var0 = "";
    String text1 = "";
    String text2 = "";

    try {
      var0 = request.getParameter("param0");
    }
    catch(Exception e) {
      e.printStackTrace();
    }
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<head><title>Servlet04</title></head>");
    out.println("<body>");
    out.println("<center>");
    out.println("Privacy Preserving Pattern Matching: Implementation
Issues<br>");
    out.println("by<br>");
    out.println("Risanuri Hidayat<br>");
    out.println("Supervisor : Vladimir Oleshchuk<br>");
    out.println("HiA<br><br>");
    out.println("<center><p>This is the result.</p>");
    out.println("<p></p>");
    out.println("Input Search : "+var0+"<br>");
    out.println("Getting Data from Administrator...<br></center>");
    out.println("Asking Decryption to Owner(s)...<br></center>");
    out.println("<table border=\"1\" width=\"100%\"><tr>");
    out.println("<p align=center>");
    out.println("<td width=\"5%\">No</td>");
    out.println("<td width=\"10%\">Owner</td>");
    out.println("<td width=\"25%\">Author</td>");
    out.println("<td width=\"55%\">Title</td>");
    out.println("<td width=\"5%\">Year</td></tr>");
    try {
      client = new Client02();
      this.client.UsergetDataFromAdmin(var0, out);
    }
    catch(Exception e) {
      e.printStackTrace();
    }
    out.println("</font>");
    out.println("</body></html>");
  }

  /**Clean up resources*/
  public void destroy() {
  }
}
```



Privacy Preserving Pattern Matching: Implementation Issues
by
Risanuri Hidayat
Supervisor : Vladimir Oleshchuk
HiA

Input

Press Search to begin searching

The input string is CASE sensitive
The minimum of input string is one word

Owner 1 (User´s RSA key Implementation)
```
package owner108;

import java.io.*;
import java.net.*;
import java.math.*;
import java.security.*;
import java.security.spec.*;
import javax.crypto.*;
import javax.crypto.spec.*;
import javax.crypto.interfaces.*;
import sun.misc.*;
import java.util.*;
import risanuri01.*;

class AKAServer01 extends Thread {
  String input = null;
  String output = null;
  String adminhost = "ikt02-16";
  SecretKey secretkey;
  IvParameterSpec spec;
  Cipher cipher;
  ServerSocket serversocket;
  Socket socket;
  int port = 6275;
  BufferedReader in;
  PrintWriter out;
  DataInputStream dis;
  DataOutputStream dos;
  BASE64Decoder deBASE64tobyte = new BASE64Decoder();
  BASE64Encoder enBASE64toString = new BASE64Encoder();
  boolean KAdone = true;
  OwnerFrame108 ownerFrame108 = new OwnerFrame108(this);
  ReadFrame108 readFrame108 = new ReadFrame108(this);
  SendFrame108 sendFrame108 = new SendFrame108(this);

  public AKAServer01()  throws Exception {
      this.start();
  }
  public void run() { //throws IOException { //, Exception {
      String str1 = "";
      try {
          serversocket = new ServerSocket(port);
          System.out.println("Owner-1 .. Listening on port "+port);
//          socket = serversocket.accept();
      } catch (Exception e) {
          System.out.println("Error creating server socket
...."+e.toString());
          System.exit(1);
      }

      while (true) {
          try {
              socket = serversocket.accept();
              dis = new DataInputStream(socket.getInputStream());
              dos = new DataOutputStream(socket.getOutputStream());
              in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
              out = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()),true);
              input = in.readLine();
              ownerFrame108.list1.add("@user ("+socket.getInetAddress()+")-
-> "+input);
              if (input.equalsIgnoreCase("@owner-2")) {
                  input = this.ReceiveLongMessage(in);
                      byte[] inputbyte =
deBASE64tobyte.decodeBuffer(input);
                      System.out.println(input);
                      BigInteger rsabig = this.RSADecryption(new
BigInteger(inputbyte),"nd1.txt","d1.txt");
                      byte[] rsabyte = rsabig.toByteArray();
                      output = new String(rsabyte);
                      System.out.println("To User :"+output);
                      this.SendingLongMessage(output, out);
              }
              if (input.equalsIgnoreCase("@owner-1_&_owner-2")) {
                  input = this.ReceiveLongMessage(in);
                      byte[] inputbyte =
deBASE64tobyte.decodeBuffer(input);
                      System.out.println(input);
                      BigInteger rsabig = this.RSADecryption(new
BigInteger(inputbyte),"nd1.txt","d1a.txt");
                      byte[] rsabyte = rsabig.toByteArray();
                      output = enBASE64toString.encodeBuffer(rsabyte);
                      System.out.println("To Owner-1 :"+output);
                      this.SendingLongMessage(output, out);
              }
              if (input.equalsIgnoreCase("@DoKeyAgreement")) {
              }
              if (KAdone) {
                  input = null;
                  input = this.ReceiveLongMessage(in);
                  String ownerinput = input;
                  while (!(input.equalsIgnoreCase("@end-decryption"))) {
                      ownerFrame108.list1.add("@user
("+socket.getInetAddress()+")--> encrypted message");
                      System.out.println("From User : "+input);
                      System.out.println();
                      if ((input.equalsIgnoreCase("Owner 1") ) ||
                          (input.equalsIgnoreCase("Owner-1") ) ||
                          (input.equalsIgnoreCase("Owner-1_&_Owner-2") )) {
                          output = input;
                          ownerinput = input;
                          System.out.println("To User :"+output);
                          this.SendingLongMessage(output, out);
                      } else {
                          byte[] ciphertext =
deBASE64tobyte.decodeBuffer(input);
                          BigInteger rsabig = null;
                          if ((ownerinput.equalsIgnoreCase("Owner 1") ) ||
                              (ownerinput.equalsIgnoreCase("Owner-1") )) {
                              rsabig = this.RSADecryption(new
BigInteger(ciphertext),"nd1.txt","d1.txt");
                          }
                          if (ownerinput.equalsIgnoreCase("Owner-1_&_Owner-
2") ) {
                              rsabig = this.RSADecryption(new
BigInteger(ciphertext),"nd1.txt","d1a.txt");
                          }
                          byte[] rsabyte = rsabig.toByteArray();
                          output = enBASE64toString.encodeBuffer(rsabyte);
                          System.out.println("To User :"+output);
```

```
                        this.SendingLongMessage(output, out);
                    }
                    input = this.ReceiveLongMessage(in);
                } // while
                System.out.println("From User : "+input);
            } else {
                System.out.println("Not KAdone");
            }
            socket.close();
            System.out.println("Socket close");
            input = null;
            output = null;
        } catch (Exception e) {
            System.out.println("Error...."+ e);
        }
    } // while

  } // constructor

  public String ReceiveLongMessage(BufferedReader readin) {
      String text = "";
      String str1 = "";
      try {
          String str = readin.readLine();
          str1 = str1 + str;
          while (!((str==null)||(str.equalsIgnoreCase("@end")))) {
              str = readin.readLine();
              if (!((str==null)||(str.equalsIgnoreCase("@end")))) { str1 =
str1 + str; }
          }
          text = str1;
      }catch (Exception exp) {}

      return text;
  } // ReceiveLongMessage

  public String getdigest(String pass) throws Exception {
      String text = pass;
      System.out.println("Hashing...");
      MessageDigest md = MessageDigest.getInstance("MD5");
      byte[] strbyte = deBASE64tobyte.decodeBuffer(text);
//      md.update(text.getBytes());
      md.update(strbyte);
      byte[] digest = md.digest();
      md.reset();
//      String text1 = new String(digest);
      String text1 = enBASE64toString.encodeBuffer(digest);
      System.out.println("Done Hashing...");

//      return digest;
      return text1;
  }// getdigest

  public BigInteger getrsakeyimpl(String filename) throws Exception {
      FileInputStream fis = new FileInputStream(filename);
      byte[] keybytes = new byte[fis.available()];
      fis.read(keybytes);
      fis.close();
      BigInteger big= new BigInteger(keybytes);

      return big;
  }// getrsakeyimpl

  public RSAPublicKeyImpl getPublicKeyImpl(String mod, String expo) throws
Exception {
      BigInteger ne = this.getrsakeyimpl(mod);
      BigInteger e = this.getrsakeyimpl(expo);
      BigInteger phi = BigInteger.ONE;
      RSAPublicKeyImpl rsapublickey = new RSAPublicKeyImpl(ne, e, phi);

      return rsapublickey;
  }// getpublickey

  public RSAPrivateKeyImpl getPrivateKeyImpl(String mod, String expo)
throws Exception {
      BigInteger nd = this.getrsakeyimpl(mod);
      BigInteger d = this.getrsakeyimpl(expo);
      BigInteger phi = BigInteger.ONE;
      RSAPrivateKeyImpl rsaprivatekey = new RSAPrivateKeyImpl(nd, d, phi);

      return rsaprivatekey;
  }// getprivatekey

  public BigInteger RSADecryption(BigInteger textbig, String mod, String
expo) throws Exception {
      RSA rsa = new RSA();
      BigInteger decrypted = null;
      System.out.println("\nDecryption...");
      try {
          RSAPrivateKeyImpl rsaprivatekey =
this.getPrivateKeyImpl(mod,expo);
          decrypted = rsa.rsadp(rsaprivatekey,textbig);
          System.out.println("Done Encryption...");
      } catch (BadPaddingException bpe) {
          bpe.printStackTrace();
          throw new InvalidKeyException("Missing Crypto Algorithm");
      }
      return decrypted;
  } // RSADecryption

  public BigInteger RSAEncryption(BigInteger textbig, String mod, String
expo) throws Exception {
      RSA rsa = new RSA();
      BigInteger ciphertext = null;
      System.out.println("\nEncryption...");
      try {
          RSAPublicKeyImpl rsapublickey = this.getPublicKeyImpl(mod,expo);
          ciphertext = rsa.rsaep(rsapublickey,textbig);
          System.out.println("Done Encryption...");
      } catch (BadPaddingException bpe) {
          bpe.printStackTrace();
          throw new InvalidKeyException("Missing Crypto Algorithm");
      }
    return ciphertext;
  } // RSAEncryption

  public void SendingLongMessage(String text, PrintWriter admout) {
      try {
          int len = text.length();
          while (len>50) {
              String stext = text.substring(0,50);
              admout.println(stext);
```

```
                System.out.println(stext);
                text = text.substring(50);
                len = text.length();
            } //while
            admout.println(text);
            admout.println("@end");
            System.out.println(text);
        } catch (Exception exp) {}

    } // SendingLongMessage

    public void SendingToAdmin(String categori, String text, PrintWriter
admout) {
        BigInteger textbig;
        BigInteger textbigencrypted;
        byte[] textbyteencrypted;
        String texttosend;

        try {
            if (!(categori.equalsIgnoreCase("@doc-save"))) {
                textbig = new BigInteger(text.getBytes());
                textbigencrypted =
this.RSAEncryption(textbig,"n1.txt","e1.txt");
                textbyteencrypted = textbigencrypted.toByteArray();
                texttosend =
enBASE64toString.encodeBuffer(textbyteencrypted);
            } else {
                texttosend = text;
            }
            admout.println(categori);
            this.SendingLongMessage(texttosend, admout);
        } catch (Exception exp) {}
    }

    public void SendingHashToAdmin(String categori, String text, PrintWriter
admout) {
        try {
            StringTokenizer st = new StringTokenizer(text);
            while (st.hasMoreTokens()) {
                String stext = st.nextToken();
                int len = stext.length();
                if ((stext.substring(len-1).equalsIgnoreCase(","))||
                    (stext.substring(len-1).equalsIgnoreCase("."))) {
                    stext = stext.substring(0,len-1);
                }
                System.out.println(stext);
                if (!(stext.equalsIgnoreCase("and"))) {
                    String hashtext = this.getdigest(stext);
                    admout.println(categori);
                    admout.println(stext);
                    admout.println(hashtext);
                }
            } // while
        } catch (Exception exp) {}
    } /// SendingHashToAdmin

    public void OwnerSavingToAdmin(String text) {
        String hashtext = "";
        try {

            Socket admsocket = new Socket(adminhost, 6270);
```

```
          BufferedReader admin = new BufferedReader(new
InputStreamReader(admsocket.getInputStream()));
          PrintWriter admout = new PrintWriter(new
OutputStreamWriter(admsocket.getOutputStream()),true);

          admout.println(text);
          System.out.println(text);

          System.out.println("@author-save");
          String author = this.sendFrame108.AuthortextField.getText();
          this.SendingToAdmin("@author-save", author, admout);

          System.out.println("@title-save");
          String title = this.sendFrame108.TitletextField.getText();
          this.SendingToAdmin("@title-save",title, admout);

          System.out.println("@year-save");
          String year = this.sendFrame108.YeartextField.getText();
          this.SendingToAdmin("@year-save",year, admout);

          System.out.println("@doc-save");
          String doc = this.sendFrame108.Docchoice.getSelectedItem();
          this.SendingToAdmin("@doc-save",doc, admout);

          System.out.println("@author-hash");
          this.SendingHashToAdmin("@author-hash", author, admout);

          System.out.println("@title-hash");
          String keyword = this.sendFrame108.WordtextField.getText();
          this.SendingHashToAdmin("@title-hash", keyword, admout);

          System.out.println("@year-hash");
          this.SendingHashToAdmin("@year-hash", year, admout);

          System.out.println("@doc-hash");
          this.SendingHashToAdmin("@doc-hash", doc, admout);

          admout.println("@end");

          admsocket.close();
      } catch (Exception exp) {}

  } // OwnerSavingToAdmin

  public void OwnerReadFromAdmin(String input) throws Exception {
//      System.out.println("Beginning Communication ...");
      String text = "";
      String hashtext = "";
      Socket admsocket = new Socket(adminhost, 6270);
      BufferedReader admin = new BufferedReader(new
InputStreamReader(admsocket.getInputStream()));
      PrintWriter admout = new PrintWriter(new
OutputStreamWriter(admsocket.getOutputStream()),true);
      admout.println("@owner-read");
      hashtext = this.getdigest(input);
      admout.println(hashtext);
      text = admin.readLine();
      String ownertext = text;
      while (!(text.equalsIgnoreCase("@end"))) {
          if (!(text.equalsIgnoreCase("@not-found") )) {
              System.out.println(text);
              if ((text.equalsIgnoreCase("Owner 1") ) ||
                  (text.equalsIgnoreCase("Owner 2") ) ||
                  (text.equalsIgnoreCase("Owner-1") ) ||
                  (text.equalsIgnoreCase("Owner-2") ) ||
                  (text.equalsIgnoreCase("Owner-1_&_Owner-2") )) {
                  ownertext = text;
                  text = text;
              } else {
                  if ((ownertext.equalsIgnoreCase("Owner 1") ) ||
                      (ownertext.equalsIgnoreCase("Owner-1") )) {
                      byte[] textbyte = deBASE64tobyte.decodeBuffer(text);
                      BigInteger detextbig = this.RSADecryption(new
BigInteger(textbyte),"nd1.txt","d1.txt");
                      byte[] detextbyte = detextbig.toByteArray();
                      text = new String(detextbyte);
                  }
                  if ((ownertext.equalsIgnoreCase("Owner 2") ) ||
                      (ownertext.equalsIgnoreCase("Owner-2") )) {
                      Socket socket = new Socket("ikt02-16", 6276);
                      BufferedReader infromowner1 = new BufferedReader(new
InputStreamReader(socket.getInputStream())); // from owner
                      PrintWriter outtoowner1 = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()),true); // to owner
                      outtoowner1.println("@owner-1");
                      this.SendingLongMessage(text,outtoowner1);
                      text = this.ReceiveLongMessage(infromowner1);
                  }
                  if (ownertext.equalsIgnoreCase("Owner-1_&_Owner-2") ) {
                      byte[] textbyte = deBASE64tobyte.decodeBuffer(text);
                      BigInteger detextbig1 = this.RSADecryption(new
BigInteger(textbyte),"nd1.txt","d1a.txt");

                      Socket socket = new Socket("ikt02-16", 6276);
                      BufferedReader infromowner2 = new BufferedReader(new
InputStreamReader(socket.getInputStream())); // from owner
                      PrintWriter outtoowner2 = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()),true); // to owner
                      outtoowner2.println("@Owner-1_&_Owner-2");
                      System.out.println("To Owner 2   : "+text);
                      this.SendingLongMessage(text,outtoowner2);
                      String text2 = this.ReceiveLongMessage(infromowner2);
                      System.out.println("From Owner 2 : "+text2);
                      byte[] textbyte2 =
deBASE64tobyte.decodeBuffer(text2);

                      BigInteger detextbig2 = new BigInteger(textbyte2);
                      BigInteger n = this.getrsakeyimpl("nd1.txt");

                      BigInteger newbig =
detextbig1.multiply(detextbig2).mod(n);

                      text = new String(newbig.toByteArray());
                  }
              }
          }
          ownerFrame108.list1.add("@admin ("+admsocket.getInetAddress()+")
--> "+text);
          text = admin.readLine();
      } // while

  } // OwnerGetData
  public SecretKey getSecretKey() {
      return secretkey;
  }
  public IvParameterSpec getspec() {
      return spec;
  }
  public void sendMessage(String stringtosend) {
      out.println(stringtosend);
  }

  public String receiveMessage() throws Exception {
      return (in.readLine());
  }

} // Class AKAServer02

public class Owner108 {
  SecretKey secretkey;
  IvParameterSpec spec;
  String input = null;

  public Owner108() throws IOException, Exception {
      AKAServer01 akaserver = new AKAServer01();

  }
  public static void main(String[] args) throws IOException, Exception {
    Owner108 owner1081 = new Owner108();
  }
}
```

```
package owner108;

import java.awt.*;
import com.borland.jbcl.layout.*;
import java.awt.event.*;

/**
 * Title:
 * Description:
 * Copyright:    Copyright (c) 2002
 * Company:
 * @author
 * @version 1.0
 */

public class OwnerFrame108 extends Frame {
  Panel panel1 = new Panel();
  List list1 = new List();

  AKAServer01 akaserver01;
  Button SENDbutton = new Button();
  FlowLayout flowLayout1 = new FlowLayout();
  Button READbutton = new Button();
  Button DOWNbutton = new Button();
  Button CLEARbutton = new Button();

  public OwnerFrame108(AKAServer01 aka) {
  this.akaserver01 = aka;
    try {
      jbInit();
      this.setVisible(true);
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
//  public static void main(String[] args) {
//    OwnerFrame04 ownerFrame04 = new OwnerFrame04();
//  }
  private void jbInit() throws Exception {
    this.setSize(400,300);
    this.setTitle("Owner-1");
    this.addWindowListener(new java.awt.event.WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        this_windowClosing(e);
      }
    });
    panel1.setLayout(flowLayout1);
    SENDbutton.setLabel("SEND");
    SENDbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        SENDbutton_actionPerformed(e);
      }
    });
    READbutton.setLabel("READ");
    READbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        READbutton_actionPerformed(e);
      }
    });
    DOWNbutton.setLabel("DOWN");
    DOWNbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        DOWNbutton_actionPerformed(e);
      }
    });
    CLEARbutton.setLabel("CLEAR");
    CLEARbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        CLEARbutton_actionPerformed(e);
      }
    });
    this.add(list1, BorderLayout.CENTER);
    this.add(panel1, BorderLayout.SOUTH);
    panel1.add(CLEARbutton, null);
    panel1.add(SENDbutton, null);
    panel1.add(READbutton, null);
    panel1.add(DOWNbutton, null);
  }

  void DOWNbutton_actionPerformed(ActionEvent e) {
      System.exit(0);
  }

  void SENDbutton_actionPerformed(ActionEvent e) {
      akaserver01.sendFrame108.setVisible(true);
  }

  void READbutton_actionPerformed(ActionEvent e) {
      akaserver01.readFrame108.setVisible(true);
  }

  void CLEARbutton_actionPerformed(ActionEvent e) {
    this.list1.removeAll();
  }

  void this_windowClosing(WindowEvent e) {
      System.exit(0);
  }
}
```

```
package owner108;

import java.awt.*;
import com.borland.jbcl.layout.*;
import java.awt.event.*;

/**
 * Title:
 * Description:
 * Copyright:    Copyright (c) 2002
 * Company:
 * @author
 * @version 1.0
 */

public class ReadFrame108 extends Frame {
  AKAServer01 akaserver01;

  Label searchlabel = new Label();
  TextField textField = new TextField();
  VerticalFlowLayout verticalFlowLayout1 = new VerticalFlowLayout();
  Button Searchbutton = new Button();
  Label label1 = new Label();
  VerticalFlowLayout verticalFlowLayout2 = new VerticalFlowLayout();
  VerticalFlowLayout verticalFlowLayout3 = new VerticalFlowLayout();
  VerticalFlowLayout verticalFlowLayout4 = new VerticalFlowLayout();

  public ReadFrame108(AKAServer01 aka) {
    this.akaserver01 = aka;
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
//  public static void main(String[] args) {
//    ReadFrame04 readFrame04 = new ReadFrame04();
//  }
  private void jbInit() throws Exception {
  this.setSize(350, 175);
    this.setTitle("Read Frame Owner-1");
    this.addWindowListener(new java.awt.event.WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        this_windowClosing(e);
      }
    });
    Searchbutton.setLabel("BEGIN SEARCH");
    Searchbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        Searchbutton_actionPerformed(e);
      }
    });
    this.setLayout(verticalFlowLayout4);
    searchlabel.setText("Search Word");
    textField.setText("");
    label1.setText("You have to fill the textfield");
    this.add(searchlabel, null);
    this.add(textField, null);
    this.add(label1, null);
    this.add(Searchbutton, null);
  }

  void this_windowClosing(WindowEvent e) {
      this.setVisible(false);
  }

  void Searchbutton_actionPerformed(ActionEvent e) {
      String text = this.textField.getText();
      if (text.length()==0) {
          label1.setText("YOU HAVE TO FILL THE TEXTFIELD");
      } else {
      try {
          akaserver01.OwnerReadFromAdmin(text);
          label1.setText("You have to fill the textfield");
      } catch (Exception exc) {}
      }
  }
}
```

```
package owner108;

import java.awt.*;
import com.borland.jbcl.layout.*;
import java.awt.event.*;
import javax.swing.*;

/**
 * Title:
 * Description:
 * Copyright:    Copyright (c) 2002
 * Company:
 * @author
 * @version 1.0
 */

public class SendFrame108 extends Frame {
  AKAServer01 akaserver01;
  VerticalFlowLayout verticalFlowLayout1 = new VerticalFlowLayout();
  Label Wordlabel = new Label();
  TextField WordtextField = new TextField();
  Button SENDbutton = new Button();
  Label Authorlabel = new Label();
  TextField AuthortextField = new TextField();
  Label Titlelabel = new Label();
  TextField TitletextField = new TextField();
  Label Yearlabel = new Label();
  TextField YeartextField = new TextField();
  Label Documentlabel = new Label();
  Choice Docchoice = new Choice();
  Panel panel1 = new Panel();
  FlowLayout flowLayout1 = new FlowLayout();
  Button Cancellbutton = new Button();
  Button kEYWORDbutton = new Button();
  VerticalFlowLayout verticalFlowLayout2 = new VerticalFlowLayout();
  VerticalFlowLayout verticalFlowLayout3 = new VerticalFlowLayout();

  public SendFrame108(AKAServer01 aka) {
    this.akaserver01 = aka;
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
//  public static void main(String[] args) {
//    SendFrame04 sendFrame04 = new SendFrame04();
//  }
  private void jbInit() throws Exception {
    this.setSize(new Dimension(350, 355));
    this.setTitle("Send Frame Owner-1");
    this.addWindowListener(new java.awt.event.WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        this_windowClosing(e);
      }
    });
    Wordlabel.setFont(new java.awt.Font("Dialog", 1, 16));
    Wordlabel.setText("Key word for the title");
    this.setLayout(verticalFlowLayout3);
    SENDbutton.setLabel("SEND");
    SENDbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        SENDbutton_actionPerformed(e);
      }
    });
    WordtextField.setText("");
    Authorlabel.setText("Author");
    Titlelabel.setText("Title");
    Yearlabel.setText("Year");
    YeartextField.setText("2002");
    Documentlabel.setText("Owner");
    Docchoice.addItemListener(new java.awt.event.ItemListener() {
      public void itemStateChanged(ItemEvent e) {
        Docchoice_itemStateChanged(e);
      }
    });
    panel1.setLayout(flowLayout1);
    Cancellbutton.setLabel("CANCELL");
    Cancellbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        Cancellbutton_actionPerformed(e);
      }
    });
    kEYWORDbutton.setLabel("generate key word");
    kEYWORDbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        kEYWORDbutton_actionPerformed(e);
      }
    });
    Docchoice.add("Owner-1");
    Docchoice.add("Owner-1_&_Owner-2");
    panel1.add(kEYWORDbutton, null);
    panel1.add(SENDbutton, null);
    panel1.add(Cancellbutton, null);
    this.add(Authorlabel, null);
    this.add(AuthortextField, null);
    this.add(Titlelabel, null);
    this.add(TitletextField, null);
    this.add(Yearlabel, null);
    this.add(YeartextField, null);
    this.add(Documentlabel, null);
    this.add(Docchoice, null);
    this.add(Wordlabel, null);
    this.add(WordtextField, null);
    this.add(panel1, null);
  }

  void this_windowClosing(WindowEvent e) {
      this.setVisible(false);
  }

  void SENDbutton_actionPerformed(ActionEvent e) {
      if (!(WordtextField.getText().equalsIgnoreCase("")) ||
          !(AuthortextField.getText().equalsIgnoreCase("")) ||
          !(YeartextField.getText().equalsIgnoreCase("")) ) {
          akaserver01.OwnerSavingToAdmin("@owner-save");
      }
  }

  void Docchoice_itemStateChanged(ItemEvent e) {
      System.out.println(Docchoice.getSelectedItem());
  }

  void Cancellbutton_actionPerformed(ActionEvent e) {
      this.setVisible(false);
  }

  void kEYWORDbutton_actionPerformed(ActionEvent e) {
      this.WordtextField.setText(TitletextField.getText());
  }

}
```

Owner 2 (User´s RSA key Implementation)
```
package owner208;
import java.io.*;
import java.net.*;
import java.math.*;
import java.security.*;
import java.security.spec.*;
import javax.crypto.*;
import javax.crypto.spec.*;
import javax.crypto.interfaces.*;
import sun.misc.*;
import java.util.*;
import risanuri01.*;

class AKAServer01 extends Thread {
  String input = null;
  String output = null;
  String adminhost = "ikt02-16";
  SecretKey secretkey;
  IvParameterSpec spec;
  Cipher cipher;
  ServerSocket serversocket;
  Socket socket;
  int port = 6276;
  BufferedReader in;
  PrintWriter out;
  DataInputStream dis;
  DataOutputStream dos;
  BASE64Decoder deBASE64tobyte = new BASE64Decoder();
  BASE64Encoder enBASE64toString = new BASE64Encoder();
  boolean KAdone = true;
  OwnerFrame208 ownerFrame208 = new OwnerFrame208(this);
  ReadFrame208 readFrame208 = new ReadFrame208(this);
  SendFrame208 sendFrame208 = new SendFrame208(this);

  public AKAServer01()  throws Exception {
      this.start();
  }
  public void run() { //throws IOException { //, Exception {
      String str1 = "";
      try {
          serversocket = new ServerSocket(port);
          System.out.println("Owner-2 .. Listening on port "+port);
//          socket = serversocket.accept();
      } catch (Exception e) {
          System.out.println("Error creating server socket
...."+e.toString());
          System.exit(1);
      }

      while (true) {
          try {
              socket = serversocket.accept();
              dis = new DataInputStream(socket.getInputStream());
              dos = new DataOutputStream(socket.getOutputStream());
              in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
              out = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()),true);
              input = in.readLine();
              ownerFrame208.list1.add("@user ("+socket.getInetAddress()+")-
-> "+input);
              if (input.equalsIgnoreCase("@owner-1")) {
                  input = this.ReceiveLongMessage(in);
                      byte[] inputbyte =
deBASE64tobyte.decodeBuffer(input);
                      System.out.println(input);
                      BigInteger rsabig = this.RSADecryption(new
BigInteger(inputbyte),"nd2.txt","d2.txt");
                      byte[] rsabyte = rsabig.toByteArray();
                      output = new String(rsabyte);
                      System.out.println("To owner-1 :"+output);
                      this.SendingLongMessage(output, out);
              }
              if (input.equalsIgnoreCase("@owner-1_&_owner-2")) {
                  input = this.ReceiveLongMessage(in);
                      byte[] inputbyte =
deBASE64tobyte.decodeBuffer(input);
                      System.out.println(input);
                      BigInteger rsabig = this.RSADecryption(new
BigInteger(inputbyte),"nd1.txt","d1b.txt");
                      byte[] rsabyte = rsabig.toByteArray();
                      output = enBASE64toString.encodeBuffer(rsabyte);
                      System.out.println("To Owner-1 :"+output);
                      this.SendingLongMessage(output, out);
              }
              if (input.equalsIgnoreCase("@DoKeyAgreement")) {
              }
              if (KAdone) {
                  input = null;
                  input = this.ReceiveLongMessage(in);
                  String ownerinput = input;
                  while (!(input.equalsIgnoreCase("@end-decryption"))) {
                      ownerFrame208.list1.add("@user
("+socket.getInetAddress()+")--> encrypted message");
                      System.out.println("From User : "+input);
                      System.out.println();
                      if ((input.equalsIgnoreCase("Owner 2") ) ||
                          (input.equalsIgnoreCase("Owner-2") ) ||
                          (input.equalsIgnoreCase("Owner-1_&_Owner-2") )) {
                          output = input;
                          ownerinput = input;
                          System.out.println("To User :"+output);
                          this.SendingLongMessage(output, out);
                      } else {
                          byte[] ciphertext =
deBASE64tobyte.decodeBuffer(input);
                          BigInteger rsabig = null;
                          if ((ownerinput.equalsIgnoreCase("Owner 2") ) ||
                              (ownerinput.equalsIgnoreCase("Owner-2") )) {
                              rsabig = this.RSADecryption(new
BigInteger(ciphertext),"nd2.txt","d2.txt");
                          }
                          if (ownerinput.equalsIgnoreCase("Owner-1_&_Owner-
2") ) {
                              rsabig = this.RSADecryption(new
BigInteger(ciphertext),"nd1.txt","d1b.txt");
                          }
                          byte[] rsabyte = rsabig.toByteArray();
                          output = enBASE64toString.encodeBuffer(rsabyte);
                          System.out.println("To User :"+output);
```

```
                        this.SendingLongMessage(output, out);
                    }
                    input = this.ReceiveLongMessage(in);
                } // while
                System.out.println("From User : "+input);
            } else {
                System.out.println("Not KAdone");
            }
            socket.close();
            System.out.println("Socket close");
            input = null;
            output = null;
        } catch (Exception e) {
            System.out.println("Error...."+ e);
        }
    } // while

  } // constructor

  public String ReceiveLongMessage(BufferedReader readin) {
      String text = "";
      String str1 = "";
      try {
          String str = readin.readLine();
          str1 = str1 + str;
          while (!((str==null)||(str.equalsIgnoreCase("@end")))) {
              str = readin.readLine();
              if (!((str==null)||(str.equalsIgnoreCase("@end")))) { str1 =
str1 + str; }
          }
          text = str1;
      }catch (Exception exp) {}

      return text;
  } // ReceiveLongMessage

  public String getdigest(String pass) throws Exception {
      String text = pass;
      System.out.println("Hashing...");
      MessageDigest md = MessageDigest.getInstance("MD5");
      byte[] strbyte = deBASE64tobyte.decodeBuffer(text);
//      md.update(text.getBytes());
      md.update(strbyte);
      byte[] digest = md.digest();
      md.reset();
//      String text1 = new String(digest);
      String text1 = enBASE64toString.encodeBuffer(digest);
      System.out.println("Done Hashing...");

//      return digest;
      return text1;
  }// getdigest

  public BigInteger getrsakeyimpl(String filename) throws Exception {
      FileInputStream fis = new FileInputStream(filename);
      byte[] keybytes = new byte[fis.available()];
      fis.read(keybytes);
      fis.close();
      BigInteger big= new BigInteger(keybytes);

      return big;
  }// getrsakeyimpl

  public RSAPublicKeyImpl getPublicKeyImpl(String mod, String expo) throws
Exception {
      BigInteger ne = this.getrsakeyimpl(mod);
      BigInteger e = this.getrsakeyimpl(expo);
      BigInteger phi = BigInteger.ONE;
      RSAPublicKeyImpl rsapublickey = new RSAPublicKeyImpl(ne, e, phi);

      return rsapublickey;
  }// getpublickey

  public RSAPrivateKeyImpl getPrivateKeyImpl(String mod, String expo)
throws Exception {
      BigInteger nd = this.getrsakeyimpl(mod);
      BigInteger d = this.getrsakeyimpl(expo);
      BigInteger phi = BigInteger.ONE;
      RSAPrivateKeyImpl rsaprivatekey = new RSAPrivateKeyImpl(nd, d, phi);

      return rsaprivatekey;
  }// getprivatekey

  public BigInteger RSADecryption(BigInteger textbig, String mod, String
expo) throws Exception {
      RSA rsa = new RSA();
      BigInteger decrypted = null;
      System.out.println("\nDecryption...");
      try {
          RSAPrivateKeyImpl rsaprivatekey = this.getPrivateKeyImpl(mod,
expo);
          decrypted = rsa.rsadp(rsaprivatekey,textbig);
          System.out.println("Done Encryption...");
      } catch (BadPaddingException bpe) {
          bpe.printStackTrace();
          throw new InvalidKeyException("Missing Crypto Algorithm");
      }
      return decrypted;
  } // RSADecryption

  public BigInteger RSAEncryption(BigInteger textbig, String mod, String
expo) throws Exception {
      RSA rsa = new RSA();
      BigInteger ciphertext = null;
      System.out.println("\nEncryption...");
      try {
          RSAPublicKeyImpl rsapublickey = this.getPublicKeyImpl(mod, expo);
          ciphertext = rsa.rsaep(rsapublickey,textbig);
          System.out.println("Done Encryption...");
      } catch (BadPaddingException bpe) {
          bpe.printStackTrace();
          throw new InvalidKeyException("Missing Crypto Algorithm");
      }
    return ciphertext;
  } // RSAEncryption

  public void SendingLongMessage(String text, PrintWriter admout) {
      try {
          int len = text.length();
          while (len>50) {
              String stext = text.substring(0,50);
              admout.println(stext);
```

```
                System.out.println(stext);
                text = text.substring(50);
                len = text.length();
            } //while
            admout.println(text);
            admout.println("@end");
            System.out.println(text);
        } catch (Exception exp) {}

    } // SendingLongMessage

    public void SendingToAdmin(String categori, String text, PrintWriter
admout) {
        BigInteger textbig;
        BigInteger textbigencrypted;
        byte[] textbyteencrypted;
        String texttosend;

        try {
            if (!(categori.equalsIgnoreCase("@doc-save"))) {
                textbig = new BigInteger(text.getBytes());
                textbigencrypted =
this.RSAEncryption(textbig,"n2.txt","e2.txt");
                textbyteencrypted = textbigencrypted.toByteArray();
                texttosend =
enBASE64toString.encodeBuffer(textbyteencrypted);
            } else {
                texttosend = text;
            }
            admout.println(categori);
            this.SendingLongMessage(texttosend, admout);
        } catch (Exception exp) {}
    }

    public void SendingHashToAdmin(String categori, String text, PrintWriter
admout) {
        try {
            StringTokenizer st = new StringTokenizer(text);
            while (st.hasMoreTokens()) {
                String stext = st.nextToken();
                int len = stext.length();
                if ((stext.substring(len-1).equalsIgnoreCase(","))||
                    (stext.substring(len-1).equalsIgnoreCase("."))) {
                    stext = stext.substring(0,len-1);
                }
                System.out.println(stext);
                if (!(stext.equalsIgnoreCase("and"))) {
                    String hashtext = this.getdigest(stext);
                    admout.println(categori);
                    admout.println(stext);
                    admout.println(hashtext);
                }
            } // while
        } catch (Exception exp) {}
    } /// SendingHashToAdmin

    public void OwnerSavingToAdmin(String text) {
        String hashtext = "";
        try {

            Socket admsocket = new Socket(adminhost, 6270);
```

```
          BufferedReader admin = new BufferedReader(new
InputStreamReader(admsocket.getInputStream()));
          PrintWriter admout = new PrintWriter(new
OutputStreamWriter(admsocket.getOutputStream()),true);

          admout.println(text);
          System.out.println(text);

          System.out.println("@author-save");
          String author = this.sendFrame208.AuthortextField.getText();
          this.SendingToAdmin("@author-save", author, admout);

          System.out.println("@title-save");
          String title = this.sendFrame208.TitletextField.getText();
          this.SendingToAdmin("@title-save",title, admout);

          System.out.println("@year-save");
          String year = this.sendFrame208.YeartextField.getText();
          this.SendingToAdmin("@year-save",year, admout);

          System.out.println("@doc-save");
          String doc = this.sendFrame208.Docchoice.getSelectedItem();
          this.SendingToAdmin("@doc-save",doc, admout);

          System.out.println("@author-hash");
          this.SendingHashToAdmin("@author-hash", author, admout);

          System.out.println("@title-hash");
          String keyword = this.sendFrame208.WordtextField.getText();
          this.SendingHashToAdmin("@title-hash", keyword, admout);

          System.out.println("@year-hash");
          this.SendingHashToAdmin("@year-hash", year, admout);

          System.out.println("@doc-hash");
          this.SendingHashToAdmin("@doc-hash", doc, admout);

          admout.println("@end");

          admsocket.close();
      } catch (Exception exp) {}

  } // OwnerSavingToAdmin

  public void OwnerReadFromAdmin(String input) throws Exception {
//      System.out.println("Beginning Communication ...");
      String text = "";
      String hashtext = "";
      Socket admsocket = new Socket(adminhost, 6270);
      BufferedReader admin = new BufferedReader(new
InputStreamReader(admsocket.getInputStream()));
      PrintWriter admout = new PrintWriter(new
OutputStreamWriter(admsocket.getOutputStream()),true);
      admout.println("@owner-read");
      hashtext = this.getdigest(input);
      admout.println(hashtext);
      text = admin.readLine();
      String ownertext = text;
      while (!(text.equalsIgnoreCase("@end"))) {
          if (!(text.equalsIgnoreCase("@not-found") )) {
              System.out.println(text);
              if ((text.equalsIgnoreCase("Owner 1") ) ||
                  (text.equalsIgnoreCase("Owner 2") ) ||
                  (text.equalsIgnoreCase("Owner-1") ) ||
                  (text.equalsIgnoreCase("Owner-2") ) ||
                  (text.equalsIgnoreCase("Owner-1_&_Owner-2") )) {
                  ownertext = text;
                  text = text;
              } else {
                  if ((ownertext.equalsIgnoreCase("Owner 2") ) ||
                      (ownertext.equalsIgnoreCase("Owner-2") )) {
                      byte[] textbyte = deBASE64tobyte.decodeBuffer(text);
                      BigInteger detextbig = this.RSADecryption(new
BigInteger(textbyte),"nd2.txt","d2.txt");
                      byte[] detextbyte = detextbig.toByteArray();
                      text = new String(detextbyte);
                  }
                  if ((ownertext.equalsIgnoreCase("Owner 1") ) ||
                      (ownertext.equalsIgnoreCase("Owner-1") )) {
                      Socket socket = new Socket("ikt02-16", 6275);
                      BufferedReader infromowner1 = new BufferedReader(new
InputStreamReader(socket.getInputStream())); // from owner
                      PrintWriter outtoowner1 = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()),true); // to owner
                      outtoowner1.println("@owner-2");
                      this.SendingLongMessage(text,outtoowner1);
                      text = this.ReceiveLongMessage(infromowner1);
                  }
                  if (ownertext.equalsIgnoreCase("Owner-1_&_Owner-2") ) {
                      byte[] textbyte = deBASE64tobyte.decodeBuffer(text);
                      BigInteger detextbig1 = this.RSADecryption(new
BigInteger(textbyte),"nd1.txt","d1b.txt");

                      Socket socket = new Socket("ikt02-16", 6275);
                      BufferedReader infromowner1 = new BufferedReader(new
InputStreamReader(socket.getInputStream())); // from owner
                      PrintWriter outtoowner1 = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()),true); // to owner
                      outtoowner1.println("@Owner-1_&_Owner-2");
                      System.out.println("To Owner 1   : "+text);
                      this.SendingLongMessage(text,outtoowner1);
                      String text2 = this.ReceiveLongMessage(infromowner1);
                      System.out.println("From Owner 1 : "+text2);
                      byte[] textbyte2 =
deBASE64tobyte.decodeBuffer(text2);

                      BigInteger detextbig2 = new BigInteger(textbyte2);
                      BigInteger n = this.getrsakeyimpl("nd1.txt");

                      BigInteger newbig =
detextbig1.multiply(detextbig2).mod(n);

                      text = new String(newbig.toByteArray());
                  }
              }
          }
          ownerFrame208.list1.add("@admin ("+admsocket.getInetAddress()+")
--> "+text);
          text = admin.readLine();
      } // while

  } // OwnerGetData
  public SecretKey getSecretKey() {
      return secretkey;
  }
  public IvParameterSpec getspec() {
      return spec;
  }
  public void sendMessage(String stringtosend) {
      out.println(stringtosend);
  }

  public String receiveMessage() throws Exception {
      return (in.readLine());
  }

} // Class AKAServer02

public class Owner208 {
  SecretKey secretkey;
  IvParameterSpec spec;
  String input = null;

  public Owner208() throws IOException, Exception {
      AKAServer01 akaserver = new AKAServer01();

  }
  public static void main(String[] args) throws IOException, Exception {
    Owner208 owner2081 = new Owner208();
  }
}
```

```
package owner208;

import java.awt.*;
import com.borland.jbcl.layout.*;
import java.awt.event.*;

/**
 * Title:
 * Description:
 * Copyright:    Copyright (c) 2002
 * Company:
 * @author
 * @version 1.0
 */

public class OwnerFrame208 extends Frame {
  Panel panel1 = new Panel();
  List list1 = new List();

  AKAServer01 akaserver01;
  Button SENDbutton = new Button();
  FlowLayout flowLayout1 = new FlowLayout();
  Button READbutton = new Button();
  Button DOWNbutton = new Button();
  Button CLEARbutton = new Button();

  public OwnerFrame208(AKAServer01 aka) {
  this.akaserver01 = aka;
    try {
      jbInit();
      this.setVisible(true);
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
//  public static void main(String[] args) {
//    OwnerFrame04 ownerFrame04 = new OwnerFrame04();
//  }
  private void jbInit() throws Exception {
    this.setSize(400,300);
    this.setTitle("Owner-2");
    this.addWindowListener(new java.awt.event.WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        this_windowClosing(e);
      }
    });
    panel1.setLayout(flowLayout1);
    SENDbutton.setLabel("SEND");
    SENDbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        SENDbutton_actionPerformed(e);
      }
    });
    READbutton.setLabel("READ");
    READbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        READbutton_actionPerformed(e);
      }
    });
    DOWNbutton.setLabel("DOWN");
    DOWNbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        DOWNbutton_actionPerformed(e);
      }
    });
    CLEARbutton.setLabel("CLEAR");
    CLEARbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        CLEARbutton_actionPerformed(e);
      }
    });
    this.add(list1, BorderLayout.CENTER);
    this.add(panel1, BorderLayout.SOUTH);
    panel1.add(CLEARbutton, null);
    panel1.add(SENDbutton, null);
    panel1.add(READbutton, null);
    panel1.add(DOWNbutton, null);
  }

  void DOWNbutton_actionPerformed(ActionEvent e) {
      System.exit(0);
  }

  void SENDbutton_actionPerformed(ActionEvent e) {
      akaserver01.sendFrame208.setVisible(true);
  }

  void READbutton_actionPerformed(ActionEvent e) {
      akaserver01.readFrame208.setVisible(true);
  }

  void CLEARbutton_actionPerformed(ActionEvent e) {
    this.list1.removeAll();
  }

  void this_windowClosing(WindowEvent e) {
      System.exit(0);
  }
}
```

```
package owner208;

import java.awt.*;
import com.borland.jbcl.layout.*;
import java.awt.event.*;

/**
 * Title:
 * Description:
 * Copyright:    Copyright (c) 2002
 * Company:
 * @author
 * @version 1.0
 */

public class ReadFrame208 extends Frame {
  AKAServer01 akaserver01;

  Label searchlabel = new Label();
  TextField textField = new TextField();
  VerticalFlowLayout verticalFlowLayout1 = new VerticalFlowLayout();
  Button Searchbutton = new Button();
  Label label1 = new Label();
  VerticalFlowLayout verticalFlowLayout2 = new VerticalFlowLayout();
  VerticalFlowLayout verticalFlowLayout3 = new VerticalFlowLayout();
  VerticalFlowLayout verticalFlowLayout4 = new VerticalFlowLayout();
  VerticalFlowLayout verticalFlowLayout5 = new VerticalFlowLayout();
  VerticalFlowLayout verticalFlowLayout6 = new VerticalFlowLayout();

  public ReadFrame208(AKAServer01 aka) {
    this.akaserver01 = aka;
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
//  public static void main(String[] args) {
//    ReadFrame04 readFrame04 = new ReadFrame04();
//  }
  private void jbInit() throws Exception {
  this.setSize(350, 175);
    this.setTitle("Read Frame Owner-2");
    this.addWindowListener(new java.awt.event.WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        this_windowClosing(e);
      }
    });
    Searchbutton.setLabel("BEGIN SEARCH");
    Searchbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        Searchbutton_actionPerformed(e);
      }
    });
    this.setLayout(verticalFlowLayout6);
    searchlabel.setText("Search Word");
    textField.setText("");
    label1.setText("You have to fill the textfield");
    this.add(searchlabel, null);
    this.add(textField, null);
    this.add(label1, null);
    this.add(Searchbutton, null);
  }

  void this_windowClosing(WindowEvent e) {
      this.setVisible(false);
  }

  void Searchbutton_actionPerformed(ActionEvent e) {
      String text = this.textField.getText();
      if (text.length()==0) {
          label1.setText("YOU HAVE TO FILL THE TEXTFIELD");
      } else {
      try {
          akaserver01.OwnerReadFromAdmin(text);
          label1.setText("You have to fill the textfield");
      } catch (Exception exc) {}
      }
  }
}
```

```
package owner208;

import java.awt.*;
import com.borland.jbcl.layout.*;
import java.awt.event.*;
import javax.swing.*;

/**
 * Title:
 * Description:
 * Copyright:    Copyright (c) 2002
 * Company:
 * @author
 * @version 1.0
 */

public class SendFrame208 extends Frame {
  AKAServer01 akaserver01;
  VerticalFlowLayout verticalFlowLayout1 = new VerticalFlowLayout();
  Label Wordlabel = new Label();
  TextField WordtextField = new TextField();
  Button SENDbutton = new Button();
  Label Authorlabel = new Label();
  TextField AuthortextField = new TextField();
  Label Titlelabel = new Label();
  TextField TitletextField = new TextField();
  Label Yearlabel = new Label();
  TextField YeartextField = new TextField();
  Label Documentlabel = new Label();
  Choice Docchoice = new Choice();
  Panel panel1 = new Panel();
  FlowLayout flowLayout1 = new FlowLayout();
  Button Cancellbutton = new Button();
  Button kEYWORDbutton = new Button();
  VerticalFlowLayout verticalFlowLayout2 = new VerticalFlowLayout();

  public SendFrame208(AKAServer01 aka) {
    this.akaserver01 = aka;
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
//  public static void main(String[] args) {
//    SendFrame04 sendFrame04 = new SendFrame04();
//  }
  private void jbInit() throws Exception {
    this.setSize(new Dimension(350, 355));
    this.setTitle("Send Frame Owner-2");
    this.addWindowListener(new java.awt.event.WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        this_windowClosing(e);
      }
    });
    Wordlabel.setFont(new java.awt.Font("Dialog", 1, 16));
    Wordlabel.setText("Key word for the title");
    this.setLayout(verticalFlowLayout2);
    SENDbutton.setLabel("SEND");
    SENDbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        SENDbutton_actionPerformed(e);
      }
    });
    WordtextField.setText("");
    Authorlabel.setText("Author");
    Titlelabel.setText("Title");
    Yearlabel.setText("Year");
    YeartextField.setText("2002");
    Documentlabel.setText("Owner");
    Docchoice.addItemListener(new java.awt.event.ItemListener() {
      public void itemStateChanged(ItemEvent e) {
        Docchoice_itemStateChanged(e);
      }
    });
    panel1.setLayout(flowLayout1);
    Cancellbutton.setLabel("CANCELL");
    Cancellbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        Cancellbutton_actionPerformed(e);
      }
    });
    kEYWORDbutton.setLabel("generate key word");
    kEYWORDbutton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        kEYWORDbutton_actionPerformed(e);
      }
    });
    Docchoice.add("Owner-2");
    this.add(Authorlabel, null);
    this.add(AuthortextField, null);
    this.add(Titlelabel, null);
    this.add(TitletextField, null);
    this.add(Yearlabel, null);
    this.add(YeartextField, null);
    this.add(Documentlabel, null);
    this.add(Docchoice, null);
    this.add(Wordlabel, null);
    this.add(WordtextField, null);
    this.add(panel1, null);
    panel1.add(kEYWORDbutton, null);
    panel1.add(SENDbutton, null);
    panel1.add(Cancellbutton, null);
  }

  void this_windowClosing(WindowEvent e) {
      this.setVisible(false);
  }

  void SENDbutton_actionPerformed(ActionEvent e) {
      akaserver01.OwnerSavingToAdmin("@owner-save");
//      this.Hashlabel.setText("Hashed : "+akaserver01.hashstring);
//      this.Encryptedlabel.setText("Encrypted :
"+akaserver01.hashstring.length());
  }

  void Docchoice_itemStateChanged(ItemEvent e) {
      System.out.println(Docchoice.getSelectedItem());
  }

  void Cancellbutton_actionPerformed(ActionEvent e) {
      this.setVisible(false);
  }

  void kEYWORDbutton_actionPerformed(ActionEvent e) {
      this.WordtextField.setText(TitletextField.getText());
  }

}
```

User (User´s RSA key Implementation)
```
package servlet08;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.net.*;
import java.security.*;
import javax.crypto.*;
import java.security.spec.*;
import sun.misc.*;
import java.math.*;
import javax.crypto.spec.*;
import javax.crypto.interfaces.*;
import risanuri01.*;

class AKAClient01 {
  private static final BigInteger MODULUS = new
BigInteger(1,SKIP_1024_MODULUS);
  private static final BigInteger BASE = BigInteger.valueOf(2);
  private static final DHParameterSpec PARAMETER_SPEC = new
DHParameterSpec(MODULUS,BASE);

  Socket socket;
  SecretKey secretkey;
  IvParameterSpec spec;
  String ownerhost; // = "ikt02-16";
  int ownerport; // = 6275;
  String input = null;
  BufferedReader in;
  PrintWriter out;
      DataOutputStream dos;
      DataInputStream dis;
  BASE64Encoder enBASE64toString = new BASE64Encoder();
  BASE64Decoder deBASE64tobyte = new BASE64Decoder();
  Cipher cipher;
  boolean KAdone = false;

  public AKAClient01(String host, int port) throws Exception {
      this.ownerhost = host;
      this.ownerport = port;
      Socket socket = new Socket(ownerhost, ownerport);
      in = new BufferedReader(new
InputStreamReader(socket.getInputStream())); // from owner
      out = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()),true); // to owner
       dos = new DataOutputStream(socket.getOutputStream());
       dis = new DataInputStream(socket.getInputStream());
      this.sendMessage("@DoKeyAgreement");

  }// constructor AKAclient


  public BigInteger getrsakeyimpl(String filename) throws Exception {
      FileInputStream fis = new FileInputStream(filename);
      byte[] keybytes = new byte[fis.available()];
      fis.read(keybytes);
      fis.close();
      BigInteger big= new BigInteger(keybytes);
      return big;
  }// getrsakeyimpl

  public RSAPublicKeyImpl getPublicKeyImpl(String mod, String expo) throws
Exception {
      BigInteger ne = this.getrsakeyimpl(mod);
      BigInteger e = this.getrsakeyimpl(expo);
      BigInteger phi = BigInteger.ONE;
      RSAPublicKeyImpl rsapublickey = new RSAPublicKeyImpl(ne, e, phi);

      return rsapublickey;
  }// getpublickey

  public RSAPrivateKeyImpl getPrivateKeyImpl(String mod, String expo)
throws Exception {
      BigInteger nd = this.getrsakeyimpl(mod);
      BigInteger d = this.getrsakeyimpl(expo);
      BigInteger phi = BigInteger.ONE;
      RSAPrivateKeyImpl rsaprivatekey = new RSAPrivateKeyImpl(nd, d, phi);

      return rsaprivatekey;
  }// getprivatekey

  public BigInteger RSADecryption(BigInteger textbig, String mod, String
expo) throws Exception {
      RSA rsa = new RSA();
      BigInteger decrypted = null;
      System.out.println("\nDecryption...");
      try {
          RSAPrivateKeyImpl rsaprivatekey = this.getPrivateKeyImpl(mod,
expo);
          decrypted = rsa.rsadp(rsaprivatekey,textbig);
          System.out.println("Done Encryption...");
      } catch (BadPaddingException bpe) {
          bpe.printStackTrace();
          throw new InvalidKeyException("Missing Crypto Algorithm");
      }
      return decrypted;
  } // RSADecryption

  public BigInteger RSAEncryption(BigInteger textbig, String mod, String
expo) throws Exception {
      RSA rsa = new RSA();
      BigInteger ciphertext = null;
      System.out.println("\nEncryption...");
      try {
          RSAPublicKeyImpl rsapublickey = this.getPublicKeyImpl(mod, expo);
          ciphertext = rsa.rsaep(rsapublickey,textbig);
          System.out.println("Done Encryption...");
      } catch (BadPaddingException bpe) {
          bpe.printStackTrace();
          throw new InvalidKeyException("Missing Crypto Algorithm");
      }
    return ciphertext;
  } // RSAEncryption

  public void SendingLongMessage(String text, PrintWriter admout) {
      try {
          int len = text.length();
          while (len>50) {
              String stext = text.substring(0,50);
```

```
                admout.println(stext);
                text = text.substring(50);
                len = text.length();
            } //while
            admout.println(text);
            admout.println("@end");
        } catch (Exception exp) {}

    } // SendingLongMessage

    public String ReceiveLongMessage() {
        String text = "";
        String str1 = "";
        try {
            String str = this.receiveMessage();
            str1 = str1 + str;
            while (!((str==null)||(str.equalsIgnoreCase("@end")))) {
                str = this.receiveMessage();
                if (!((str==null)||(str.equalsIgnoreCase("@end")))) { str1 =
str1 + str; }
            }
            text = str1;
        }catch (Exception exp) {}

        return text;
    }

    public String UserAskDecryptionToOwner(String text) throws Exception {
        String str1 = "";
        String outputtext = null;
        String stringtosend = null;

        System.out.println("From Admin : "+text);
        System.out.println();
        if ((text.equalsIgnoreCase("Owner 1") ) ||
            (text.equalsIgnoreCase("Owner 2") ) ||
            (text.equalsIgnoreCase("Owner-1") ) ||
            (text.equalsIgnoreCase("Owner-2") ) ||
            (text.equalsIgnoreCase("Owner-1_&_Owner-2") )) {
            stringtosend = text;
        } else {
            byte[] textbyte = deBASE64tobyte.decodeBuffer(text);
            BigInteger rsabig = this.RSAEncryption(new
BigInteger(textbyte),"nu.txt","eu.txt");
            byte[] rsabyte = rsabig.toByteArray();
            String output = enBASE64toString.encodeBuffer(rsabyte);
            stringtosend = output;
        }
        System.out.println("To Owner :"+stringtosend);
        this.SendingLongMessage(stringtosend,out);
        str1 = this.ReceiveLongMessage();
        System.out.println();
        System.out.println("From Owner : "+str1);
        if ((str1.equalsIgnoreCase("Owner 1") ) ||
            (str1.equalsIgnoreCase("Owner 2") ) ||
            (str1.equalsIgnoreCase("Owner-1") ) ||
            (str1.equalsIgnoreCase("Owner-2") ) ||
            (str1.equalsIgnoreCase("Owner-1_&_Owner-2") )) {
            outputtext = enBASE64toString.encodeBuffer(str1.getBytes());
        } else {
            byte[] outputbyte = deBASE64tobyte.decodeBuffer(str1);
```

```
          BigInteger rsabig = this.RSADecryption(new
BigInteger(outputbyte),"ndu.txt","du.txt");
          byte[] rsabyte = rsabig.toByteArray();
          outputtext = enBASE64toString.encodeBuffer(rsabyte);
      }
      System.out.println(outputtext);

      return outputtext;
  }//end getData

  public SecretKey getSecretKey() {
      return secretkey;
  }
  public IvParameterSpec getspec() {
      return spec;
  }

  public void sendMessage(String stringtosend) {
      out.println(stringtosend);
  }

  public String receiveMessage() throws Exception {
      return (in.readLine());
  }

} // class AKAClient

class Client02{
    String adminhost = "ikt02-16";
    int port = 6270; //echo port
    String outputtext = null;
    Socket socket;
    BufferedReader in;
    PrintWriter out;
    BASE64Decoder deBASE64tobyte = new BASE64Decoder();
    BASE64Encoder enBASE64toString = new BASE64Encoder();
    AKAClient01 toowner1 = null;
    AKAClient01 toowner2 = null;

  public Client02() throws Exception {
    try{
      Socket socket = new Socket(adminhost,port);
      in = new BufferedReader(new
InputStreamReader(socket.getInputStream())); // from admin
      out = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()),true); // to admin
    }//end try
    catch(UnknownHostException e){
      System.out.println(e);
      System.out.println("Must be online to run properly.");
    }//end catch UnknownHostException
    catch(IOException e){System.out.println(e);}
  } //Constructor Client02

  public void sendMessage(String string) {
      out.println(string);
  }

  public String receiveMessage() throws Exception {
      return (in.readLine());
  }
```

```
   public void SendingHashToAdmin(String text) {
       try {
           StringTokenizer st = new StringTokenizer(text);
           while (st.hasMoreTokens()) {
               String stext = st.nextToken();
               int len = stext.length();
               if ((stext.substring(len-1).equalsIgnoreCase(","))||
                   (stext.substring(len-1).equalsIgnoreCase("."))) {
                   stext = stext.substring(0,len-1);
               }
               System.out.println(stext);
               if (!(stext.equalsIgnoreCase("and"))) {
                   String hashtext = this.getdigest(stext);
                   out.println(hashtext);
               }
           } // while
           out.println("@end-from-user");
       } catch (Exception exp) {}
   } /// SendingHashToAdmin


   public void UsergetDataFromAdmin(String var0, PrintWriter htmlout) throws
 Exception {
     String entext = "";
     String detext = "";
     String text = "";
     int i = 1;

     try {
       this.sendMessage("@user");
       this.SendingHashToAdmin(var0);
       entext = this.receiveMessage();
       String ownertext = entext;
       if (!(entext.equalsIgnoreCase("@end-decryption"))) {
           if ((ownertext.equalsIgnoreCase("Owner 1") ) ||
               (ownertext.equalsIgnoreCase("Owner-1") )) {
               String host = "ikt02-16"; // owner-1
               int port = 6275; // owner-1
               toowner1 = new AKAClient01(host,port);
               htmlout.println("Asking decryption to Owner 1...");
           }
           if ((ownertext.equalsIgnoreCase("Owner 2") ) ||
               (ownertext.equalsIgnoreCase("Owner-2") )) {
               String host = "ikt02-16"; // owner-2
               int port = 6276; // owner-2
               toowner1 = new AKAClient01(host,port);
               htmlout.println("Asking decryption to Owner 2...");
           }
           if (ownertext.equalsIgnoreCase("Owner-1_&_Owner-2")) {
               String host1 = "ikt02-16"; // owner-1
               String host2 = "ikt02-16"; // owner-2
               int port1 = 6275; // owner-1
               int port2 = 6276; // owner-2
               toowner1 = new AKAClient01(host1,port1);
               toowner2 = new AKAClient01(host2,port2);
               htmlout.println("Asking decryption to Owner 1 and Owner 2
 ...");
           }
           while (!(entext.equalsIgnoreCase("@end-decryption"))) {
               if (i%5==1) {
                   htmlout.println("</tr>");
                   htmlout.println("<tr>");
                   htmlout.println("<td>" + ((i+5)/5) + "</td>");i++;
               }
               if ((ownertext.equalsIgnoreCase("Owner 1") ) ||
                   (ownertext.equalsIgnoreCase("Owner-1") ) ||
                   (ownertext.equalsIgnoreCase("Owner 2") ) ||
                   (ownertext.equalsIgnoreCase("Owner-2") )) {
                   String detext1 =
 toowner1.UserAskDecryptionToOwner(entext);
                   byte[] detextbyte = deBASE64tobyte.decodeBuffer(detext1);
                   detext = new String(detextbyte);
                   htmlout.println("<td>" + detext + "</td>");
                   entext = this.receiveMessage(); i++;
               }
               if ((ownertext.equalsIgnoreCase("Owner-1_&_Owner-2")) ) {
                   System.out.println("With Owner-1 ***");
                   String detext1 =
 toowner1.UserAskDecryptionToOwner(entext);
                   byte[] detextbyte1 =
 deBASE64tobyte.decodeBuffer(detext1);

                   System.out.println("With Owner-2 ***");
                   String detext2 =
 toowner2.UserAskDecryptionToOwner(entext);
                   byte[] detextbyte2 =
 deBASE64tobyte.decodeBuffer(detext2);
                   if ((new String(detextbyte1).equalsIgnoreCase("Owner-
 1_&_Owner-2")) ||
                       (new String(detextbyte2).equalsIgnoreCase("Owner-
 1_&_Owner-2")) ) {
                       detext = new String(detextbyte1);
                   } else {
                       BigInteger detextbig1 = new BigInteger(detextbyte1);
                       BigInteger detextbig2 = new BigInteger(detextbyte2);
                       BigInteger n = this.getrsakeyimpl("nd1.txt");
                       BigInteger newbig =
 detextbig1.multiply(detextbig2).mod(n);
                       detext = new String(newbig.toByteArray());
                   }

                   htmlout.println("<td>" + detext + "</td>");
                   entext = this.receiveMessage(); i++;
               }
           } // while
           htmlout.println("</tr>");
           this.toowner1.SendingLongMessage("@end-decryption",toowner1.out);
           if ((ownertext.equalsIgnoreCase("Owner-1_&_Owner-2")) ) {
               this.toowner2.SendingLongMessage("@end-
 decryption",toowner2.out);
           }

           htmlout.println(entext);
       } else {
           htmlout.println("Data is Not Found");
       }
     }
     catch (Exception e) {
       System.out.println(e.toString());
     }
```

```
  }//end UsergetDataFromAdmin

  public BigInteger getrsakeyimpl(String filename) throws Exception {
      FileInputStream fis = new FileInputStream(filename);
      byte[] keybytes = new byte[fis.available()];
      fis.read(keybytes);
      fis.close();
      BigInteger big= new BigInteger(keybytes);

      return big;
  }// getrsakeyimpl

  public String getdigest(String pass) throws Exception {
      String text = pass;
      MessageDigest md = MessageDigest.getInstance("MD5");
      byte[] strbyte = deBASE64tobyte.decodeBuffer(text);
      md.update(strbyte);
      byte[] digest = md.digest();
      md.reset();
      String text1 = enBASE64toString.encodeBuffer(digest);

      return text1;
  } // getdigest

}//end class Client02

public class Servlet1 extends HttpServlet {
  private static final String CONTENT_TYPE = "text/html";
  String input = null;
  Client02 client;

//  AKAClient01 akaclient;

  /**Initialize global variables*/
  public void init() throws ServletException {
  }
  /**Process the HTTP Get request*/
  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    String var0 = "";
    try {
      var0 = request.getParameter("param0");
    }
    catch(Exception e) {
      e.printStackTrace();
    }
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    out.println("<font color=\"green\">");
    out.println("</font>");
  }
  /**Process the HTTP Post request*/
  public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    String var0 = "";
    String text1 = "";
    String text2 = "";

    try {
      var0 = request.getParameter("param0");
    }
    catch(Exception e) {
      e.printStackTrace();
    }
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<head><title>Servlet04</title></head>");
    out.println("<body>");
    out.println("<center>");
    out.println("Privacy Preserving Pattern Matching: Implementation
Issues<br>");
    out.println("by<br>");
    out.println("Risanuri Hidayat<br>");
    out.println("Supervisor : Vladimir Oleshchuk<br>");
    out.println("HiA<br><br>");
    out.println("<center><p>This is the result.</p>");
    out.println("<p></p>");
    out.println("Input Search : "+var0+"<br>");
    out.println("Getting Data from Administrator...<br></center>");
    out.println("Asking Decryption to Owner(s)...<br></center>");
    out.println("<table border=\"1\" width=\"100%\"><tr>");
    out.println("<p align=center>");
    out.println("<td width=\"5%\">No</td>");
    out.println("<td width=\"10%\">Owner</td>");
    out.println("<td width=\"25%\">Author</td>");
    out.println("<td width=\"55%\">Title</td>");
    out.println("<td width=\"5%\">Year</td></tr>");
    try {
      client = new Client02();
      this.client.UsergetDataFromAdmin(var0, out);
    }
    catch(Exception e) {
      e.printStackTrace();
    }
    out.println("</font>");
    out.println("</body></html>");
  }

  /**Clean up resources*/
  public void destroy() {
  }
}
```

RSA Encryption/Decryption Code

```
package risanuri01;

import java.math.*;
import java.security.*;
import java.security.interfaces.*;
import javax.crypto.*;

public class RSA {

    public RSA() {
    }

    public static BigInteger rsaep( RSAPublicKey publicKey, BigInteger m)
throws IllegalBlockSizeException {
        BigInteger e = publicKey.getPublicExponent();
        BigInteger n = publicKey.getModulus();
        BigInteger nMinusOne = n.subtract(BigInteger.ONE);

        if (m.compareTo(BigInteger.ZERO) < 0 ) { // m<0
            throw new IllegalBlockSizeException("Ciphertext too small");
        }
        if (m.compareTo(nMinusOne) > 0 ) { // m>n-1
            throw new IllegalBlockSizeException("Ciphertext too large");
        }
        BigInteger c = m.modPow(e,n);
        return c;
    }

    public static BigInteger rsadp( RSAPrivateKey privateKey, BigInteger c)
{

        if (!(privateKey instanceof RSAPrivateCrtKey)) {
            BigInteger d = privateKey.getPrivateExponent();
            BigInteger n = privateKey.getModulus();
            BigInteger m = c.modPow(d,n);
            return m;
        }

        RSAPrivateCrtKey privateCrtKey = (RSAPrivateCrtKey)privateKey;

        BigInteger p = privateCrtKey.getPrimeP();
        BigInteger q = privateCrtKey.getPrimeQ();
        BigInteger dP = privateCrtKey.getPrimeExponentP();
        BigInteger dQ = privateCrtKey.getPrimeExponentQ();
        BigInteger qInv = privateCrtKey.getCrtCoefficient();

        BigInteger m1 = c.modPow(dP,p);
        BigInteger m2 = c.modPow(dQ,q);

        BigInteger h = m1.subtract(m2);
        h = h.multiply(qInv);
        h = h.mod(m2);

        BigInteger m = h.multiply(q);
        m = m.add(m2);

        return m;
    }
}
```

```
package risanuri01;

import java.security.*;
import java.math.BigInteger;

public final class RSAKeyPairGenerator extends KeyPairGeneratorSpi{
    private static final BigInteger E = BigInteger.valueOf(65537);
    private static final int CERTAINTY = 85;
    private static final int DEFAULT_STRENGTH = 1024;
    private int mKeysize;
    private SecureRandom mSecureRandom;
    private BigInteger mPublicExponent;
    private boolean mInitialized = false;

    public RSAKeyPairGenerator() {
        super();
    } // constructor

    public void initialize( int keysize, SecureRandom random) {
        this.mKeysize = keysize;
        this.mSecureRandom = random;
        this.mInitialized = true;
    }

    public KeyPair generateKeyPair() {
        if (!(mInitialized)) {
            initialize(DEFAULT_STRENGTH, new SecureRandom());
        }
        int pSize = mKeysize/2;
        int qSize = mKeysize - pSize;
        BigInteger p, q, pMinus1, qMinus1, phi, n, d;

        do {
            do {
                p = new BigInteger(pSize, CERTAINTY, mSecureRandom);
                pMinus1 = p.subtract(BigInteger.ONE);
            } while (!(pMinus1.gcd(E).equals(BigInteger.ONE)));
            do {
                q = new BigInteger(qSize, CERTAINTY, mSecureRandom);
                qMinus1 = q.subtract(BigInteger.ONE);
            } while (!(qMinus1.gcd(E).equals(BigInteger.ONE)));
        } while ((p.multiply(q)).bitLength() != mKeysize);

        phi = pMinus1.multiply(qMinus1);
        n = p.multiply(q);
        d = E.modInverse(phi);

//        return new KeyPair( new RSAPublicKeyImpl(n,d), new
RSAPrivateCrtKeyImpl(n,E,d,p,q));
        return new KeyPair( new RSAPublicKeyImpl(n,E,phi), new
RSAPrivateKeyImpl(n,d,phi));
    } // generateKeyPair
}
```

```
package risanuri01;

import java.math.BigInteger;
import java.security.*;
import java.security.interfaces.*;

public class RSAPrivateKeyImpl implements RSAPrivateKey {
    private BigInteger mModulus;
    private BigInteger mPrivateExponent;
    private BigInteger mPhi;

    public RSAPrivateKeyImpl(BigInteger modulus, BigInteger
privateExponent, BigInteger Phi) {
        this.mModulus = modulus;
        this.mPrivateExponent = privateExponent;
    } // constructor

    public BigInteger getModulus() {
        return mModulus;
    }

    public BigInteger getPrivateExponent() {
        return mPrivateExponent;
    }

    public BigInteger getPhi() {
        return mPhi;
    }

    public String getAlgorithm() {
        return "RSA";
    }

    public String getFormat() {
        return "NONE";
    }

    public byte[] getEncoded() {
        return null;
    }

}
```

```
package risanuri01;

/**
 * Title:
 * Description:
 * Copyright:    Copyright (c) 2002
 * Company:
 * @author
 * @version 1.0
 */
import java.math.BigInteger;
import java.security.*;
import java.security.interfaces.*;

public class RSAPublicKeyImpl implements RSAPublicKey{
    private BigInteger mModulus;
    private BigInteger mPublicExponent;
    private BigInteger mPhi;

    public RSAPublicKeyImpl(BigInteger modulus, BigInteger publicExponent,
BigInteger Phi) {
        this.mModulus = modulus;
        this.mPublicExponent = publicExponent;
        this.mPhi = Phi;
    } // constructor

    public BigInteger getModulus() {
        return mModulus;
    }

    public BigInteger getPublicExponent() {
        return mPublicExponent;
    }

    public BigInteger getPhi() {
        return mPhi;
    }

    public String getAlgorithm() {
        return "RSA";
    }

    public String getFormat() {
        return "NONE";
    }

    public byte[] getEncoded() {
        return null;
    }
}
```

```
package risanuri01;
import java.security.*;
import java.math.BigInteger;

public final class RSAKeyPairGenerator02 extends KeyPairGeneratorSpi{
//    private static final BigInteger E = BigInteger.valueOf(65537);
    private BigInteger E,E2,phi;
    private static final int CERTAINTY = 85;
    private static final int DEFAULT_STRENGTH = 1024;
    private int mKeysize;
    private SecureRandom mSecureRandom;
    private BigInteger mPublicExponent;
    private boolean mInitialized = false;

    public RSAKeyPairGenerator02() {
        super();
    } // constructor

    public void initialize( int keysize, SecureRandom random) {
        this.mKeysize = keysize;
        this.mSecureRandom = random;
        this.mInitialized = true;
    }

    public boolean primalitytest(BigInteger p, BigInteger pMinus1) {
        BigInteger test, m;

        m = new BigInteger(mKeysize/2, CERTAINTY, mSecureRandom);
        pMinus1 = p.subtract(BigInteger.ONE);
        test = m.modPow(pMinus1, p);
        if (test.equals(BigInteger.ONE)) {
            System.out.println(p+" is prime");
            return true;
        }
        else {
            return false;
        }
    }

    public KeyPair generateKeyPair() {
        if (!(mInitialized)) {
            initialize(DEFAULT_STRENGTH, new SecureRandom());
        }
        int pSize = mKeysize/2;
        int qSize = mKeysize - pSize;
        BigInteger p, q, pMinus1, qMinus1, n, d, d2;
        boolean fermattest = false;

        do {
            E = new BigInteger(pSize, CERTAINTY, mSecureRandom);
            fermattest =
this.primalitytest(E,(E.subtract(BigInteger.ONE)));
        } while (fermattest==false);

        do {
            do {
                p = new BigInteger(pSize, CERTAINTY, mSecureRandom);
                pMinus1 = p.subtract(BigInteger.ONE);
            } while (!(pMinus1.gcd(E).equals(BigInteger.ONE)));
            do {
                q = new BigInteger(qSize, CERTAINTY, mSecureRandom);
                qMinus1 = q.subtract(BigInteger.ONE);
            } while (!(qMinus1.gcd(E).equals(BigInteger.ONE)));
        } while ((p.multiply(q)).bitLength() != mKeysize);

        phi = pMinus1.multiply(qMinus1);
        n = p.multiply(q);
        d = E.modInverse(phi);

//        return new KeyPair( new RSAPublicKeyImpl(n,d), new
RSAPrivateCrtKeyImpl(n,E,d,p,q));
        return new KeyPair( new RSAPublicKeyImpl(n,E,phi), new
RSAPrivateKeyImpl(n,d,phi));
    } // generateKeyPair

    public KeyPair generateSecondKeyPair(BigInteger phi, BigInteger n) {
        BigInteger d2;
        boolean fermattest = false;
        int pSize = mKeysize/2;

        do {
            E2 = new BigInteger(pSize, CERTAINTY, mSecureRandom);
        } while (!(phi.gcd(E2).equals(BigInteger.ONE)));

        d2 = E2.modInverse(phi);

//        return new KeyPair( new RSAPublicKeyImpl(n,d), new
RSAPrivateCrtKeyImpl(n,E,d,p,q));
        return new KeyPair( new RSAPublicKeyImpl(n,E2,phi), new
RSAPrivateKeyImpl(n,d2,phi));
    } // generateKeyPair

}
```