Email Using Microsoft Message Queue and RSA Encryption Technique

Dr.Taif Sami Hassan
Computer Science Engineering Department, University of Al- Mammon /Baghdad
Email: Dr_taif15@yahoo.com

Received on: 24/1/2012 & Accepted on: 3/5/2012

ABSTRACT

The information exchange through the internet and especially the electronic email is one of the most recent used areas. The paper presents simple system for efficient email handling. The system depends mainly on the Microsoft Message Queue MSMQ technique, introduced by Microsoft Corporation. The system is well protected using many tools offered by MSMQ Environment and the RSA encryption method. The system characterized by its simplicity, fastness, and has the ability to install and implement using simple steps in Local Area Network (LAN) without internet and also with the internet.

Keywords: RSA, MSMQ, MOD.

البريد الالكتروني باستخدام تقنية مايكروسوفت لطابور الرسائل وتقنية (RSA) للتشفير

الخلاصة

ان تبادل المعلومات بواسطة الانترنت وخاصة باستخدام تطبيق البريد الالكتروني يعد من اهم المجالات المستخدمة. يقدم البحث نظام بسيط لمعالجة فعالة للبريد الالكتروني. يعتمد النظام بشكل اساسي على تقنية نظام طابور الرسائل لمايكروسوفت. ان النظام المقترح يتميز بقدر كبير من الحماية وذلك باستخدام العديد من الادوات التي تتوفر في نظام طابور الرسائل لمايكروسوفت وكذلك باستحدام تقنية المفتاح العام للتشفير RSA . يتميز النظام بالسهولة والسرعة وكذلك القدرة على التنصيب والتنفيذ باستخدام خطوات بسيطة في الشبكة المخلية المرتبطة بالانترنت وكذلك الشبكة المحلية غير المرتبطة بالانترنت.

INTRODUCTION

In recent days there is huge number of peoples, using email (sending and receiving message) through any LAN especially the internet. The paper presents simple and rigid system for handling email. The system depends on the Microsoft Message Queue (MSMQ). The MSMQ contain server for storing messages denoted as queues, enabling users to sending and receiving messages. The message is stored in MSMQ server, and is ready to be received by specific user. The message is divided into two parts; the headers and the message body. The header information determine the destination and source address, while the body contains the messageFinally public key encrypt (RSA technique no shred key) is applied to the message body before sending it. [1].

THE PROPOSE SYSTEM

The propose system depend basically on the Microsoft Message Queue (MSMQ) and The RSA technique as shown in Figure (1).

1-The message to be sent or receive.

2-The Microsoft Message Queue (MSMQ). Which is mainly used to store, send, and receive messages?

3-RSA Encryption technique to the message body in order to protect messages.

MICROSOFT MESSAGE QUEUE (MSMQ)

MSMQ provides services that let you manage queues of messages and route messages between queues. While the messages have a standard format, the body of each message is application-specific. MSMQ can be used to build applications that interoperate with other messaging applications running on mainframe computers or with other time-independent applications. [2]

THE MSMQ APPLICATION MODEL

The basic application model for messaging applications is straightforward. An application creates a message and sends it to a queue. Another application—or another part of the same application—reads the message from the queue andProcesses it. If necessary, the receiver can respond by sending its own message. The sending application is not required to wait for a response from the reader. In fact, the sending and receiving applications don't even need to be running at the same time. Sent messages are stored in the queue until they are retrieved by a receiver. The persistent storage of unread messages makes message queuing extremely useful for scenarios in which the applications are time-independent, in which machines might be disconnected from the network, or in which receivers might not be able to process requests as quickly as they are generated.

MSMQ supports three types of machine configurations: *server, independent client,* and *dependent client*. Servers can use all the features of MSMQ. Independent clients can create and modify local queues and can send and receive messages just like MSMQ servers. Local queues and messages can be created even when an MSMQ server is unavailable. However, independent clients do not have the intermediate store-and-forward capability of MSMQ servers, nor do they store information from the distributed MSMQ database. Dependent clients require synchronous access to a MSMQ server and cannot be used in disconnected scenarios.

MSMQ queues can be transactional, meaning that sending a message to or receiving a message from a queue is an atomic operation that participates in a transaction. These actions can be combined with the work required to create or process the message to ensure that the entire operation of creating and sending or of receiving and processing a message succeeds or fails as a whole. Because the acts of sending and receiving messages can occur over a long period of time, these operations are part of two different transactions. If a transaction on the receiver fails, it might be necessary to provide a compensating transaction on the sender to undo the effect of the already completed send transaction. In addition, the act of waiting for a message to be received can be a lengthy operation and is typically non transactional. Thus, the receiving action is usually divided into two parts. A long-lived object, application, or server running outside MTS, known as the *listener,* is used to wait for messages coming into the queue. When a message arrives, the listener calls a method on a transactional receiver component that retrieves the message from the queue and processes it. [2]

MSMQ supports both public and private message queues. Public queues are registered in the MSMQ Information Store (MQIS) so that they can be located by any MSMQ application. Public queues are persistent, and their registration Information can be backed up; thus, these queues are good for long-term use. Private queues are registered on a local computer and usually cannot be seen by other applications. Information about private queues is stored in the local queue storage (LQS) directory on the local computer. The advantage of private queues is that they have no MQIS overhead, which means they are faster to create, have no latency, and do not need to be replicated. In addition, they can be used when MQIS is not available. Private queues can be exposed to another application by sending the queue's location to the other application.

Queue properties are defined when the queue is created. The properties can be set by MSMQ or by the application. Properties include ways to identify the queue and ways to specify who has access to the queue, whether the queue is transactional, whether a journal of queue operations should be maintained, and more. You can also use the queue properties as filters to locate public queues that have particular characteristics.

PROGRAMMING MSMQ APPLICATIONS

MSMQ provides two ways to access MSMQ functionality. First, it provides a set of API functions. These functions are most easily used from C or C++ programs and also VB. The MSMQ API includes functions for creating, opening, and deleting queues; functions for locating existing queues and messages in queues; functions for sending messages and reading them in queues; and functions for setting and retrieving properties. Second, MSMQ provides a set of Automation components that can be used from any language that supports Automation, including scripting languages. These components do not support all the functionality of the MSMQ API, but they do support the functions most often needed by applications: queue lookup, queue management, queue administration, message management, and transaction support. We will focus here on these [3].

SENDING AND RECEIVING MESSAGES

Sending messages is a straightforward process. After opening a queue for sending create an MSMQ Message object, set its properties, and call the Send method. If you have created a transactional queue, the message will participate in MTS automatic transactions by default. The Body property of the object is used to specify the contents of the message. This property can be set to any intrinsic Variant type, array of bytes, or a persistent object. The MSMQMessage object figures out what type of data is provided by inspecting the Variant data assigned to the Body property. Be aware that when you assign an object to the Body property, you should not use the Microsoft Visual Basic Set operator because the actual content of the Variant data is copied into the message.

Receiving messages is slightly more complicated. There is ability to read messages synchronously or asynchronously. When a message is received synchronously, execution of the receiver is blocked until the message is available or a time-out period expires. When a message is received asynchronously, execution of the receiver continues until *Arrived* or *Arrived Error* events are received by an *MSMQEvent* object registered with the queue. However, even with asynchronous reads, the act of receiving a message might occur over a long period of time. Thus, it is usually impossible to wait for messages to be received within a transactional component.

THE ENCRYPTION

One of the biggest problems in cryptography is the distribution of keys. Suppose you live in the United States and want to pass information secretly to your friend in Europe. If you truly want to keep the information secret, you need to agree on some sort of key that you and he can use to encode/decode messages. But you don't want to keep using the same key, or you will make it easier and easier for others to crack your cipher. But it's also a pain to get keys to your friend. If you mail them, they

might be stolen. If you send them cryptographically, and someone has broken your code, that person will also have the next key. If you have to go to Europe regularly to hand-deliver the next key, that is also expensive. If you hire some courier to deliver the new key, you have to trust the courier.

THE RSA TECHNIQUE

The best known public key method is the RSA algorithm, named after its three inventors: Rivest, Shamir and Adelman. The fundamental difference between a private key system and a public key system is that the latter uses a different key to decrypt the ciphertext from the key that was used to encrypt it. A public key system uses a pair of keys: one for the sender and the other for the recipient. [4]

Although this may not seem to help, the inventors of the RSA algorithm used number theory to develop a method of generating a pair of numbers- the keys- in such a way that a message encrypted using the first number of the pair can be decrypted only by the second number. Furthermore, the second number cannot be derived from the first. This second property means that the first number of the pair can be made available to anyone who wishes to send an encrypted message to the holder of the second number since only that person can decrypt the resulting ciphertext message. The first number of the pair is known as the public key and the second the private or secret key. The principle of the method is shown in Figure 10.8. As indicated, the derivation of the two keys is based on number theory and is therefore outside the scope of this book. However, the basic algorithm used to compute the two keys is simple and is summarized here together with a much simplified example. [5].

To create the public key Kp:

■ Select two large positive prime numbers P and Q P= 7, Q = 17

■ Compute X = (P – 1) × (Q – 1) X = 96

■ Choose an integer E which is prime relative to X, i.e., not a prime factor of X or a multiple of it, and which satisfies the condition indicated below for the computation of Ks E = 5

■ Compute N = P ×Q N= 119

■ KP is then N concatenated with E KP = 119, 5

To create the secret key Ks:

■ Compute D such that MOD (D × E, X) = 1 D × 5/96 = 1, D = 77

■ Ks is then N concatenated with D Ks = 119, 77

To compute the cipher text C of plaintext P:

■ Treat P as a numerical value P = 19

■ C = MOD (PE, N) C = MOD (195, 119) C = 66

To compute the plaintext P of cipher text C:
■ P = MOD (CD, N) P = MOD (6677, 119)
P = 19
The choice of E and D in this example is best seen by considering the factors
Of 96. These are 1, 2, 3, 4, 6, 8, 12, 16, 24, 32, and 48. The list of numbers that are
Prime relative to 96 are thus 5, 7, 9, 10, 11, and so on. If we try the first of
These, E = 5, then there is also a number D = 77 that satisfies the condition
MOD (D × E, X)=1 and hence these are chosen**.**

THE ADOPTED SYSTEM

First the message must be partition into:

The source address: 10 Byte

- The destination address: 10 Byte
- The Subject: 10 Byte
- The Body (variable but it is fixed for testing)

The destination address is the name for the region directory in the MSMQ that is related to specific person. The directory contains all messages for the person. The destination address is useful for the other side in order to know the sender and can reply by a message.[6] The subject is only to inform the receiver for the subject of the message.

The body is the real data that contain the message. The body was encrypted using the RSA technique and the public key is 5, while the private key is 77. As shown in the following figure2.In the other side the message is open and the header is obtained, also the body message is decrypt. Example for the RSA algorithmThe character is A

1. The character is a
2. Find the ASCII code ASC("a")= 97
3. Normalize the number in order to lower the characters range
4. 97-97+1=1
5. Implement the RSA by
6. cc = (1^ ee) Mod nn
   Where ee is public key and is 5, while nn =pp X qq =7 X 17 =119
   cc=1
7. Sending the number

The system is shown in Figure (2)

RESULTS AND DISCUSSION

The system was programmed using Visual Basic. At Figure (3) the MSMQ environment was shown; while the Figure (4) shows the MSMQ environment and the message body**.**

Figure (5) introduce the system screen (message without encryption). It includes

- ü The left side contain the frame with source address, destination address, subject, and body message for sending.
- ü The right side contains the frame with source address, destination address, subject, and body message for the receiving.
- ü Create button is used for create queue in MSMQ.
- ü Send button is use for sending the message (in the text box).
- ü RSA Encryption is used for encrypt the text in text box(the body) .
- ü Receive is used for receiving the data**.**

Figure 6 introduce the system screen (message with RSA encryption)**.**

CONCLUSIONS

The paper presents efficient email handling system. The system depends basically on the Microsoft Message Queue (MSMQ). The message is constructed by filling the headers and the message body. The MSMQ enable us to choose the proper security issue, which could be handling by Windows Server Operating System. And also to determine the storage of the server. There is another security issue, RSA encryption technique which is efficient public key encryption. The main system facilities compared with the tradition system is shown in table (1).

Table (1) Comparison between Yahoo and System facilities.

Criteria	*The Yahoo Email*	*The Adopted System Email*
Simplicity	**Complex**	**Simple**
Viruses	**Large probability to attack.**	**Low probability to attack.**
Reliability	**Need fast internet line.**	**Can work with slow internet line.**
Availability	**Work with only internet**	**Work with internet or with any network**
Security	**Determine by Yahoo corporation.**	**There are many methods that could be used.**

REFERENCES

[1] Beharous A. Forouzan, "Data Communication and Network", McGRAW. HILL international edition, 4 edit 2007'

[2] Mary Kirtland, "Designing Component Based Application" Microsoft Press, Redmond, Washington 98052-6399, 1998'

[3] Stig F. Mjolsnes ,"Multidisciplinary Introduction to Information Security (Discrete Mathematics and Its Applications)", Nov 9, 2011.

[4] Fred Halsall, "Computer Networking and the Internet", 5th edition, Person Education Limited 2005'

[5] Jianying Zhou and Moti Yung,"Applied Cryptography and Network Security", 8th International Conference, ACNS 2010, Beijing, China, June 22-25, 2010,

[6] Proceedings (Lecture Notes in Computer Science / Security and Cryptology). (Aug 11, 2010).

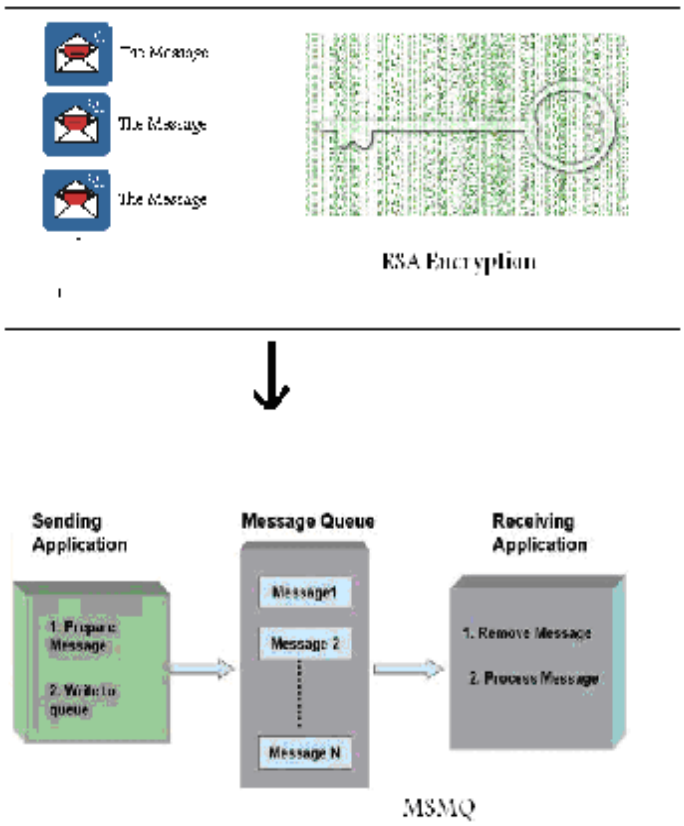

Figure (1) the Proposed System

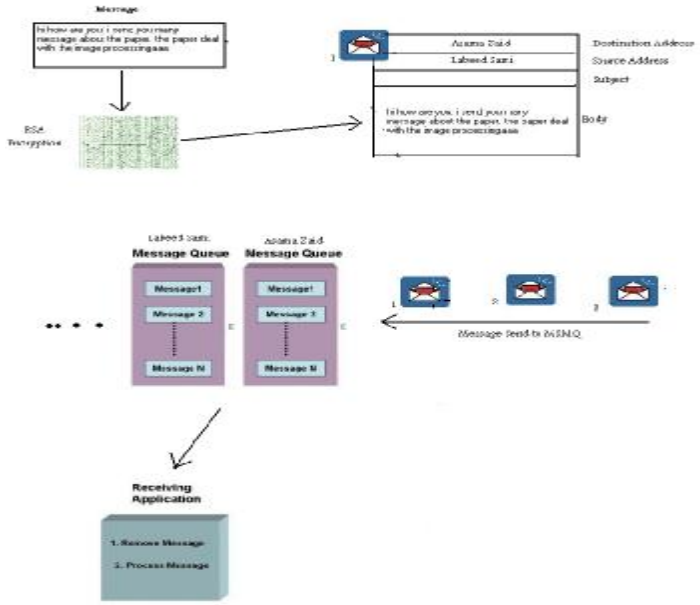

Figure (2) the System Detail.

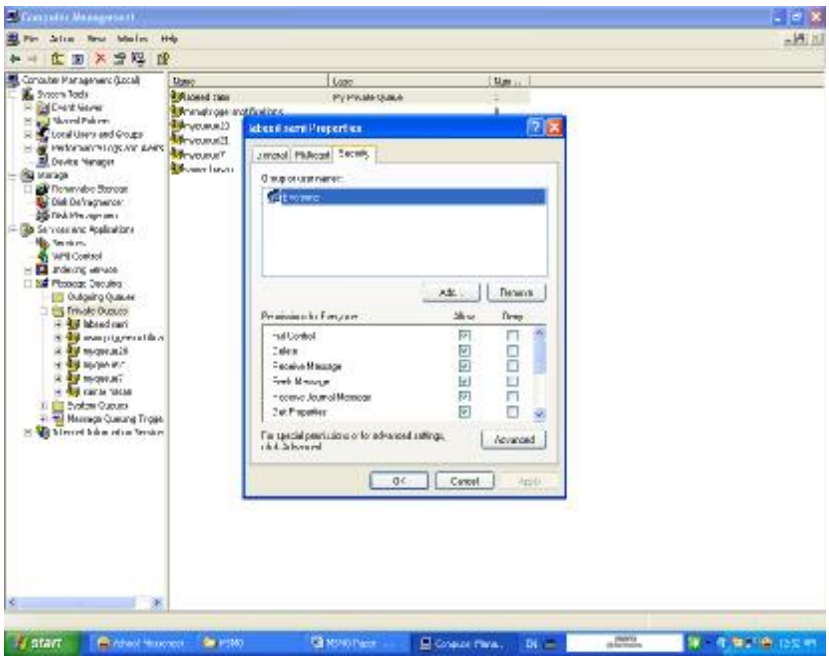

Figure (3) the MSMQ Environment.

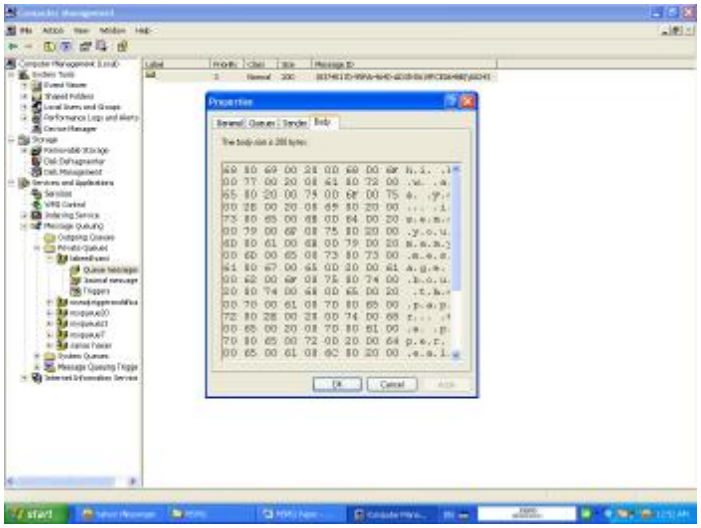

Figure (4) the MSMQ Message Body.

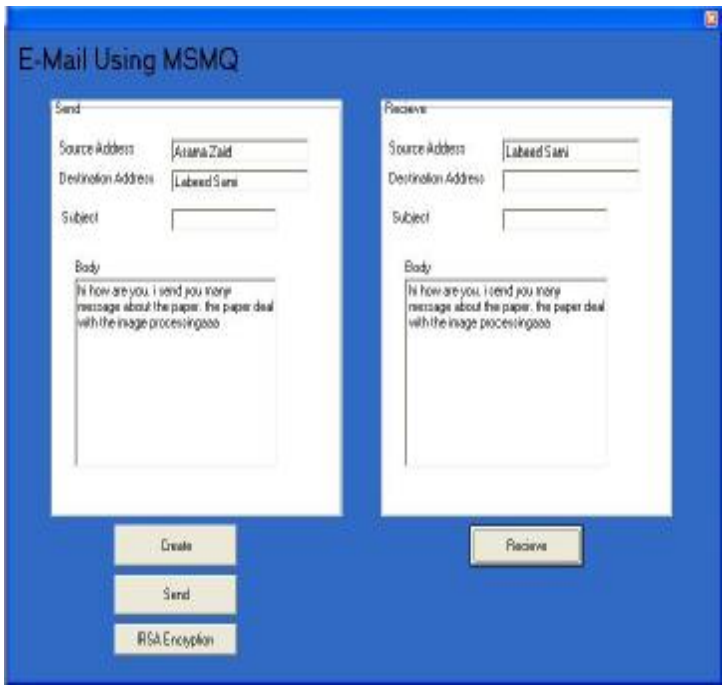

Figure (5) System Screen (message without encryption).

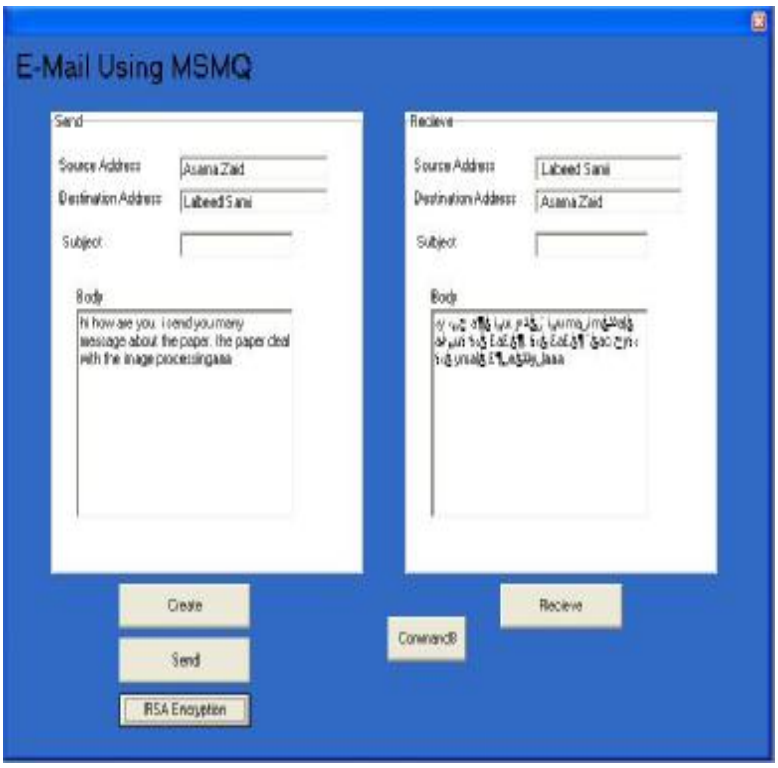

Figure (6) System Screen (message with encryption).