Handling Encrypted Email at the Gateway of a Network

Nalin Subramanian
nvsubram@uncc.edu
Dept. of Computer Science
UNCC, Charlotte
NC, USA

Bradley Wagner
bjwagne1@uncc.edu
Dept. of Computer Science
UNCC, Charlotte
NC, USA

Yousef Khader
ykhader@uncc.edu
Dept. of Computer Science
UNCC, Charlotte
NC, USA

Abstract

Protection for email is provided by client side encryption from being read or hacked. But the security risk in this protection is that it also protects virus from being sensed by the anti virus software. It also hides the hacker's destructive payload. The encrypted message can't be examined by any anti virus software rather filtered by content specific filtering engine. In case of providing perimeter defense for email virus, the network administrator faces the problem of disclosing the email or denies sending the email. If the email is allowed then network is not secure, if the message is disclosed then the message privacy is not maintained. Hence we suggest having an Email Virus Certification Authority which certifies all the emails sent from the message generating gateway or mail exchange server. This paper addresses the above problem and provides a theoretical solution for such security and privacy issue. Simple simulation is also implemented.

Keywords: *Virus, Encryption, Email, Certification Authority*

Participants and Research:
Yousef Khader
- Email protocol & Exchange servers Research

Nalin VimalKumar Subramanian
- Gateway & Network technology Research

Bradley Wagner
- Detecting encrypted data & Antivirus inner-workings

Java Code – written by all of us

I. INTRODUCTION:

With the increasing efficiency of firewalls and other protection against traditional hacking, viruses, worms and other forms of attacks have become the preferred means of viciously attacking enterprises. To implement an effective virus protection strategy it is necessary to understand the source of infections.

Email has become one of the most frequent means of communicating with clients, suppliers, students and employees. Emails these days are considered to be the number one source for initiating electronic attacks, using malicious code embedded in email or attachments. Perhaps the best way to protect a network from email viruses would be to install anti-virus software at the network gateway. However, there are problems with this approach for example; what if an email is encrypted and must be decrypted by the receiver's PC, then the gateway would not be able to detect such viruses.

Scanning encrypted e-mail messages for viruses poses such a difficulty for the enterprise. Strong person-to-person encryption is great for privacy, but hampers virus-scanning efforts. Conversely, aggressive antivirus scanning at the gateway means that a potentially sensitive message is decrypted somewhere before it reaches its intended recipient. Scanning encrypted email can be tricky but it can be done without sacrificing security for privacy or vice versa. The Encrypted Email virus problem is shown in Figure 3.

II. OBJECTIVE:

Develop an antivirus email system that resides on a network gateway and overcomes the problems associated with encrypted email data. The system must be able to provide strong virus protection and an ability to detect encrypted email. If encrypted email is detected, the system must decrypt the data and then recheck it for any viruses. If a virus were found attached to any of the emails, data would not be sent through the gateway into the computer network. The system must be able to decrypt the email in efficient fashion in order to avoid slowing down the email delivery service.

In terms of deliverables, we will provide our research along with any simulations and/or code we have developed. Our system will overcome the problems mentioned in our preceding motivation section.

III. BACKGROUND / RELATED WORK

In order to fully understand the complexities involved with developing a system to handle this problem, other research has been examined. Five common methods for dealing with encrypted viruses at the gateway have been identified and been examined. Also, a comparison between these techniques has been established.

Technique #1
Cryptographic facilities in the enterprise may be configured to always include a particular enterprise administrative key when encrypting a dataset, and virus-scanning processes may be given that key. (Morar & Chess)

Problem
Not all cryptographic facilities include such a feature (the *Windows 2000* recovery key has a similar but not directly applicable function, and recent enterprise versions of PGP apparently have a related feature), and the risk posed by the possible theft of that administrative 'read everything' key may be too high to bear. (Morar & Chess) Although this technique would be very effective in solving the encrypted email problem, there are huge issues of privacy that essentially make it unfeasible. Many users would undoubtedly be uncomfortable encrypting private data that they know can be viewed by administrators.

Advantages
Very efficient technique in disclosing encrypted email. It is very good protection method for enterprise from email viruses.

Disadvantages
One major disadvantage of this technique is the risk of having a key that have a huge capability (like reading everything) to get lost or stolen from the administrative personals. Another disadvantage is the compromising of the privacy of the email. Since the administrator will have the key he will have to decrypt the received message in order for him to check for viruses, and by doing that the privacy of that message has been invaded. The final disadvantage of this technique would be the user satisfaction issue. Since the client would know that the private message will be shared with the administrator they will not feel comfortable using this method at all times.

Technique #2
End-to-end encryption may simply be forbidden in an enterprise, with all traffic checked for viruses before it leaves the trusted intranet, and encrypted afterward for travel outside (via for instance a Virtual Private Network). (Morar & Chess)

Problem
Enforcing such a rule would be very difficult to sustain. This technique would be effective in eliminating viruses from leaving a network but may actually be less effective than doing nothing for viruses coming in. The no encryption rule would most likely give users within the network a sense of security that can easily be exploited by encrypted viruses that enter the network from other systems with less stringent encryption rules.
Advantages
Very effective technique when it comes to eliminating the viruses from going outside or leaving a network.

Disadvantages
There is no such privacy of the email not so ever. Therefore, that will lead for no user satisfaction of using such method or technique.

Technique #3
Users making use of encryption may be required to have up-to-date real-time virus scanning in place on the client machine. (Morar & Chess)

Problem
It is notoriously difficult, especially with the proliferation of mobile users and laptop computers, to track all the client systems that might be attached to your enterprise Intranet and ensure that certain software is always installed and active, although various commercially-available enterprise anti-virus solutions take some steps in that direction. On the other hand, having good anti-virus software on client systems is desirable in any case, so most enterprises already have such a requirement, however well or badly they are able to enforce it. (Morar & Chess)

Technique #4
All encrypted emails that enter the enterprise gateway will be quarantined and placed into a separate, secure folder.

Problem
Although this technique would be effective, it provides a disservice to the users within the network. Users who are unable to receive their emails because they contain encrypted elements would no doubt cause users to simple switch email servers.

Advantages
It is very excellent technique for maintaining the privacy of the clients. Also, it is used in conjunction with gateway checking to implement "hybrid" systems. If maintained effectively, provides a high level of security

Disadvantages
It is very difficult to maintain. Also it is very hard to track all client systems (laptops, etc).

Technique #5
Encrypted email will be forwarded to a folder that contains all such emails for a specific user. The user then must have some means of decrypting all of his/her emails and have them scanned by the anti-virus software.

Problem
This technique provides a viable system for checking and delivering emails but has a major drawback. There must be some method that allows the intended receiver of an encrypted email to send his/her secret key to the gateway so the email can be checked. Another problem with this technique is the idea of storing all of the emails in folders designated for each user in the network. Storing this data in an area that can be accessed

by administrators poises more issues about privacy and security. We have improved this method and the following simulation and implementation technique outlines a solution.

Advantages
Encrypted emails that enter the gateway will be quarantined and placed into a separate and secure folder. In which it will provide a very good security for the encrypted emails. High security and privacy

Disadvantages
Too many drawbacks, also what to do with the encrypted emails once they are in the folder? It is very low user satisfaction.

Here is a chart that we have established to compare all these previous methods that has been mentioned above. It is to show comparison between the different methods and techniques that we have researched. Figure 1

IV. SIMULATION:

In the scenario, the problem is the encrypted virus. The issue is how to scan an encrypted file/email for the virus. We mainly have two terms rolled into this scenario, "virus" and "encryption". So the focus can be diverted to solve the problem focusing on both the terms. Let us start looking at the term "virus". We can scan for the virus in the encrypted file for virus pattern, via pattern matching. But there is a problem with this technique. The encrypted file scrambles the virus code all over the file. Even though the idea can be rolled to find a heuristic to figure out the virus in the encrypted file, there is higher possibility that it can give fault alert. Even higher level of pattern matching may result in such scenario. This is because it is hard to detect the latest polymorphic viruses, which is one of the key research areas.

So the next term is "Encryption". The main reason for encryption is to provide privacy, confidentiality for the message/data. This claims to have a penalty of network security when virus is sent in the encrypted form. This is pictorially represented in the figure 3.

So for each and every network or email server there should be an Email Virus Certification Authority (EVCA). This EVCA is a highly trusted party and known to all in the network. Each and every user in the gateway/Network has a unique shared secret key with the EVCA. All transactions between EVCA and User will be secure. If the user is new to the gateway, by proving his/her identity using their public key/ certificate with the EVCA, a session key is generated to establish a secure channel. Another optimum solution for implementing the EVCA is to use Identity based public key cryptosystem. By this way each user receives a private key and public key from the EVCA's master key using their identity. EVCA knows all users private key for decryption. The private key is protected by means of EVCA's master key. This way the secure channel can be implemented indirectly.

Here is the idea, for each and every encrypted email the EVCA is responsible for the virus protection. For each encrypted email the EVCA requests the sender for the decryption key (through the secure channel) to decrypt, check for virus, certify and forward the email if it is virus free.

Over here there is a problem of exposing the key to the EVCA. But as EVCA is a trusted party the key can be exchanged, and it should be provided key expiration or encrypted key form or some kind of privacy in the key, but still the EVCA should be able to decrypt the file for checking. It should be assured that the EVCA doesn't store any key. The key transaction is secured via shared session key/ secret key / master key of EVCA between EVCA and Sender.

But another issue is that the email is passed through various gateways before it reaches the receiver. So, there comes the over head of verification phase and key distribution in each and every gateway. Having inter-trusted EVCA among the gateways can solve this. Here the inter operation among the EVCA's should be information assured. Hence the trust is maintained. So the first EVCA receiving the encrypted email should perform the verification task. It then assures the email for virus free and passes it to other gateway to reach the receiver.

By this way the sending of encrypted virus is protected at the gateway. Vice versa, the receiving of encrypted virus will also be avoided. If this module protects the gateway from letting the virus out, then the virus can't spread to other gateways. Vice versa, if the module protects against receiving any virus into the gateway then the gateway is free from virus. The EVCA module discussed above can be installed at each gateway or email server. For providing more transparency and flexibility the inter operation should be assured. In case, if an email is sent from a gateway which has no EVCA, then the first EVCA gateway request for the secure channel with the sender and certifies the email.

The major responsibilities of EVCA are Obtaining Valid certification, Inter operability between multiple gateways EVCA, recent Virus Updates, and mainly Non storage of any process/ event /file in the session other than Certified Email. The EVCA component pictorial representation is shown in Figure 4.

V. ANALYSIS OF THE APPROACH:

Advantages

By this approach Spreading of virus via Email will be eliminated completely. User as a recipient will not receive any email virus. Even the affected machine will spread no virus automatically via email. The Virus is controlled within a gateway. Works well with both symmetric and asymmetric encryption technique. This reduces the false virus detection by pattern matching. The Virus update or Scanning is not required at recipient side. EVCA Gateway provides complete protection and has High Reliability, reduced redundant check

Disadvantages

The only disadvantage of this approach is that EVCA comes to know the key during the session in the case of non-identity based public key cryptosystem. Confidentiality is lost to only one EVCA per Email.

But when we consider a gateway with no EVCA, all users don't do Virus Updates. This may result in infecting entire system and entire gateway. Finally, all the user in the gateway losses their entire confidentiality. So when compared the drawback of not having an EVCA, having EVCA is beneficial, provided the EVCA is a highly trusted entity.

VI. CASE STUDY / CURRENT RESEARCH:

There are Organizations which has Email servers which does not allows encrypted or compresses email attachments. Say for example Wachovia Bank strips off all the .zip attachments or encrypted attachments. Lot many Network Email Protection Software are in use and being focused on this issue. Lot of major Business organization is concerned about this issue, and has invested on this project.

VII. IMPLEMENTATION:

The program is written in Java. To run this program initially few packages have to be installed and class path have to be set properly. The packages we have used are,

1. Bouncy Castle security package (www.bouncycastle.org) – an open source security package
    a. JDK 1.4 bouncy castle provider
    b. Clean Room JCE and provider
2. Java Mail API
3. Java Mail Activation files
4. CORBA Package

We have to download the .jar files and set the environment variable "CLASSPATH" properly to all the above .jar files.

Now a walk through of the programming will be given in this following section. There are four files include this simulation: three are java files, one is an IDL file. CORBA is used to simulate the communication between the CA server component and Client component. The IDL file gives the interface for the functions that the client uses to communication with the EVCA server.

Then among the three java files, one is EVCA Client component which is responsible as a sender, the next one is EVCA Server which acts as the CA and does all the main part, the last java file is meant to decrypt the received mail.

Initially, the IDL file is converted into java file and all the CORBA components are created. Then, ORBD is started at any available port. Then the EVCA server is started at the same port. The server starts running. Then EVCA clients get started in the same port.

EVCA Client and EVCA Server:

This client component generates a message. It generates its private and public key to prove its identity. It also generates the secret key which it uses to encrypt the message. This secret key is stored in a file for the receiver to use it to decrypt the message. The client gets the link with EVCA server. Then establishes a secure channel by means of proving each of their identity by means of their public key. Any thing passed through this secure channel is encrypted.

The EVCA server generates a session key and passes to EVCA client through the secure channel. The client sends the message in the encrypted form using the secret key, and passes the encrypted secret key using the session key.

EVCA server receives the encrypted message, key extracts them; uses the secret key to decrypt the message verifies with some fixed virus patterns. If the message is free from virus, then the message is sent to the receiver as attachment; otherwise a virus alert message is sent to the receiver.

decryptMessage:

This is last component used by the receiver for decrypting the message. When the receiver receives the message, he should save the attachment as "c:\msg.enc". Provided the receiver holds the secret key in the "c:\KeyToDecrypt1" file, this component uses the key and extracts the message from the file.

As this is for simulation, we have assumed that all the messages sent are encrypted, EVCA and sender are in the same port, UNCC university email account is used for sending purpose. Initially the sender is set to nvsubram@uncc.edu and receiver is bjwagne1@uncc.edu. The Client and server have to be started each time sending the mail. Client console displays the actions performed. Finally the server displayed the message which it is sending. The virus patterns are taken to be predefined. This is only for demo purpose.

How to Compile / Run:

1. Open command prompt windows and change to jdk directory or where you have stored the simulation files
2. Execute the command: idlj –fall EVCA.idl

3. Compile EVCAServer: javac EVCAServer.java EVCAApp/*.java
4. Compile EVCAClient: javac EVCAClient.java EVCAApp/*.java
5. Compile decryptMessage: javac decryptMessage.java
6. Run: start orbd -ORBInitialPort 6060
7. Run EVCAServer: start java EVCAServer -ORBInitialPort 6060
8. Run EVCAClient: java EVCAClient –ORBInitialPort 6060
9. Receiver receives the encrypted email attachment if it is virus free otherwise warning
10. save the attachment to "c:\MSG.enc"
11. store the secret key to decrypt in the "c:\KeyToDecrypt1"
12. Run the decryptMessage: java decryptMessage

Whenever you want to change the message, sender, receiver just change in the client component, recompile and run the client component. If you need to change the virus patterns change it in the server component, recompile and run.

VIII. CONCLUSION:

The main aim of this paper is how to eliminate spreading of Encrypted Email Virus. As the encrypted email can't be scanned for virus with any antivirus software, here we propose the idea of having an email virus certification authority (EVCA). EVCA monitors Email at the gateway level. It certifies the Virus Free Email and forward to next gateway. The certification is done after verification by either obtaining decryption key from the sender through a secure channel or by the use of Identity based public key cryptosystem.

This work can be extended to a location transparent decryption, verification and certification. A Zero Knowledge approach can also be incorporated. Added Intrusion Detection for the detected encrypted email virus will help tracking the affected system and prevent further spreading.

REFERENCES

[1] Nicholas Weaver, Vern Paxson, Stuart Staniford, Robert Cunningham *"Internet WORMS: past, present, and future: A taxonomy of computer worms "*, Proceedings of the 2003 ACM workshop on Rapid Malcode , October 2003.

[2] Dave Jones *"Building an e-mail virus detection system for your network ",* Linux Journal, Volume 2001 Issue 92, December 2001

[3] Paul Schmehl *"Technical Session: Barbarians at the gateway, defeating viruses in EDU"*, Proceedings of the 29th annual ACM SIGUCCS conference on User services, October 2001

[4] Manasi Bhattacharyya, Shlomo Hershkop, Eleazar Eskin, *"Intrusion detection and response: MET: an experimental system for Malicious Email Tracking",* Proceedings of the 2002 workshop on New security paradigms, September 2002.

[5] Martin Ferris, *"New Email Security Infrastructure",* Proceedings of the 1994 workshop on New security paradigms, Aug 1994.

[6] http://www.ciphertrust.com/files/pdf/bitpipe/securing_email_systems.pdf

[7] http://www.astaro.com/data/pdf/whitepapers/Whitepaper_Desktop_Gatewayen.pdf

[8] http://www.research.ibm.com/antivirus/SciPapers/VB2000JFM.pdf, John.F.Morar & David.M.Chess

[9] Mark W. Eichin and Jon A. Rochlis,*" With Microscope and Tweezers: An Analysis of the Internet Virus of November 1988"*, Proceedings of the 1989 IEEE Computer Society Symposium on security and Privacy 1989

[10] Darrell M. Kienzle, Matthew C. Elder *"Internet WORMS: past, present, and future: Recent worms: a survey and trends "*, October 2003 Proceedings of the 2003 ACM workshop on Rapid Malcode.

[11] Dennis Fowler *"Attack of the killer virus!"*, December 2003 net Worker, Volume 7 Issue 4

[12] The Symantec Enterprise Papers Volume, *"Understanding and Managing Polymorphic Virus",*security response.symantec.com/avcenter/ reference/striker.pdf

[13] Matthew M. Williamson *"Design, Implementation and Test of an Email Virus Throttle "*, December 2003 Proceedings of the 19th Annual Computer Security Applications Conference

[14] http://www.indiciisalus.com/OurTechnology/downloads/AntiVirus_WhitePaper.pdf

APPENDIXES FOR FIGURES

Figure 1 – Comparison of Techniques for Dealing with Encrypted Email

-	Backdoor Key	No Encryption	Client Machine	Quarantine
Security (from Virus)	Very High	Low	Fair	Very High
Privacy	Potentially Extremely Low	Extremely Low	Very High	Very High
Feasibility	Fair	Low	High	High(with reliable delivery method)
User Satisfaction	Very Low	Low	Fair	Low (dependent on delivery)

Figure 2: The picture figures out the area of focus to look up for the virus in a gateway.

http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/exchange/exchange200

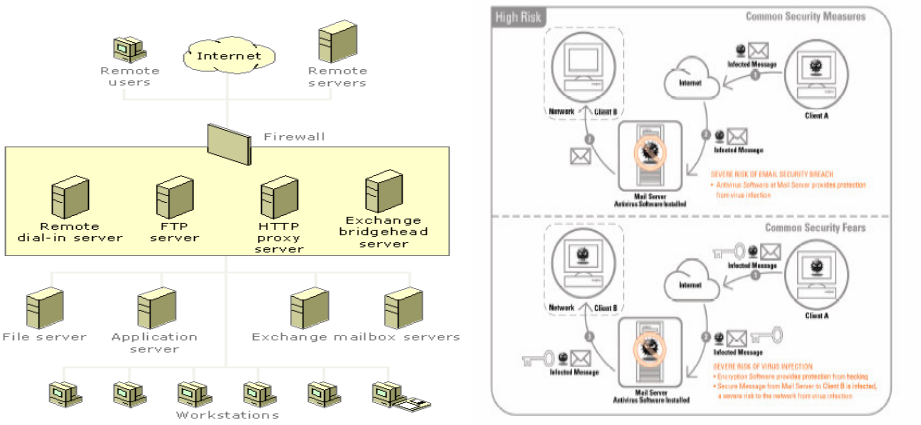

Figure 3: Visual representation of the problem
http://www.indiciisalus.com/OurTechnology/downloads/AntiVirus_WhitePaper.pdf

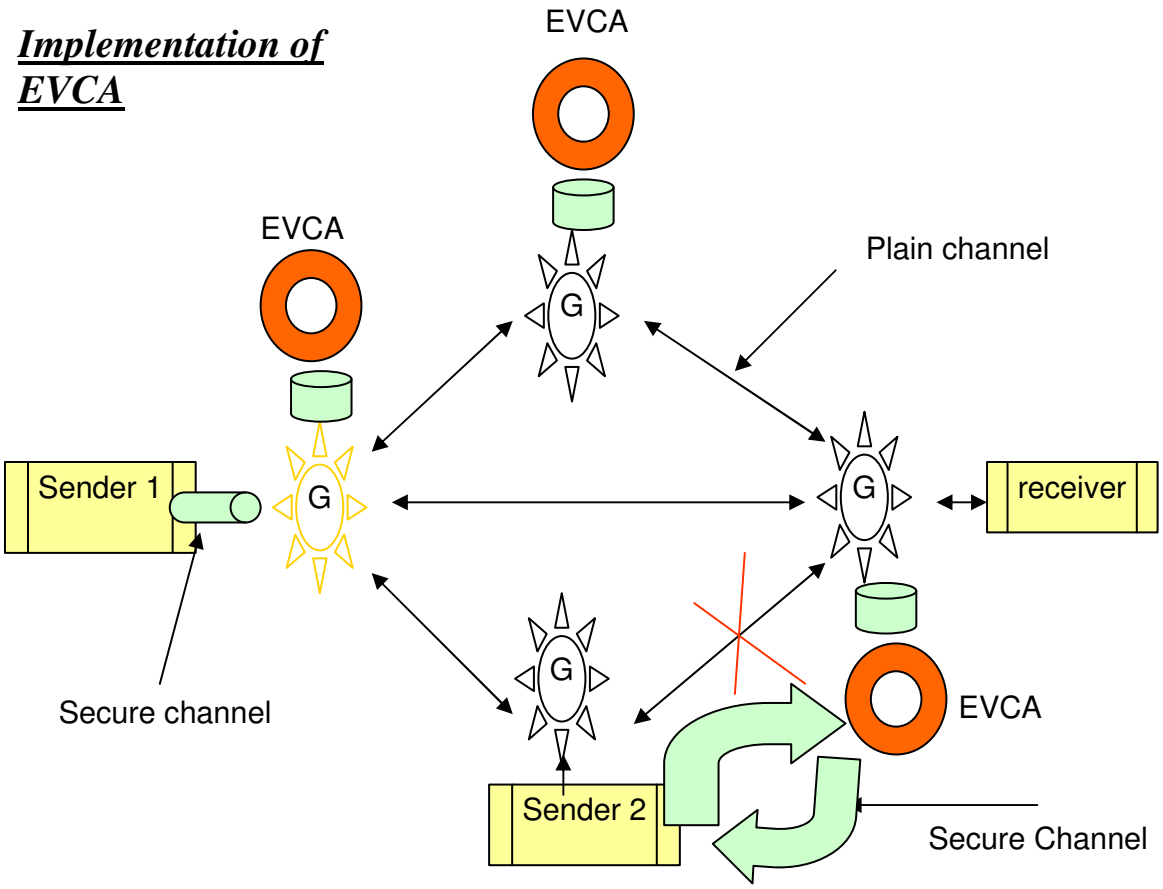

Figure 4: Solution to the problem – our approach

PRESENTATION HANDOUT

Attached separately as it is in .ppt format. Actual implementation varies slightly from the presentation slides, as we eliminated the receiver component and added the receiver decryption component.

PSEUDO CODE

Communication Interface:

```
module EVCAApp {

exception EVCAException{};
typedef octet theByte;
typedef sequence <theByte> byteArray;
        interface Comm {
                byteArray getPubKey(in byteArray pubKey) raises (EVCAException);
                string getEncMessage(in byteArray encMessage) raises (EVCAException);
                byteArray getOkay(in long ack) raises (EVCAException);
                long getDec(in string decMessage) raises (EVCAException);
                byteArray getRand(in string k1) raises (EVCAException);
                long getEncPriKey(in byteArray encPriKey, in byteArray encMessage, in string where, in string
from, in string to) raises (EVCAException);
                oneway void shutdown();
        };
};
```

Sender:

```
    // create and initialize the ORB
   ORB orb = ORB.init(argv, null);

    // get the root naming context
    org.omg.CORBA.Object objRef =
                                orb.resolve_initial_references("NameService");
    // Use NamingContextExt instead of NamingContext. This is
    // part of the Interoperable naming Service.
    NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

    // resolve the Object Reference in Naming
    String name = "EVCA";
    commImpl = CommHelper.narrow(ncRef.resolve_str(name));

    System.out.println("Obtained a handle on server object ");

        //RSA key generation for personal identification
                JDKKeyPairGenerator.RSA kpgRSA=new JDKKeyPairGenerator.RSA() ;// =
                KeyPairGenerator.getInstance("RSA","BC");
                kpgRSA.initialize(512,new SecureRandom());
                KeyPair kp = kpgRSA.generateKeyPair();
                RSAprivateKey = (RSAPrivateKey)kp.getPrivate();
                RSApublicKey = (RSAPublicKey)kp.getPublic();

        //sending public key and receiving CA's public key
                ec.sendPubKey(RSApublicKey.getEncoded() );

        //sending an encrypted message for testing CA's public key
                if (ec.sendEncMessage("This is the Encrypted Message for testing CA's public Key")){

                        //proving personal identification with CA
                        if (ec.sendDec(ec.sendOkay(GOOD))==1){

                                //receiving the session key from CA
                                ec.sendRand("Requesting CA the session key");
                                //creating secret key between sender and receiver
```

```
                    CipherKeyGenerator kg =new CipherKeyGenerator();
                    kg.init(new KeyGenerationParameters(new SecureRandom(),128));
                    secKey=kg.generateKey();

            //storing secret key for reciever usage
                    OutputStream f0=new FileOutputStream("C:\\KeyToDecrypt1");
                    f0.write(secKey);
                    f0.close();

            //key and message to send and sending to CA
                    String msgToSend="Hi Brad, this is a test Message to be encrypted and sent, Nalin";
        ec.sendEncPriKey(secKey,msgToSend,"smtpexpress.uncc.edu","nvsubram@uncc.edu","bjwagne1@uncc.ed
u");

            //closing the session with CA
            commImpl.shutdown();

                    }//end the personal verification to CA
                    else{
                            System.out.println("Error Reported by CA for clients public Key");
                            commImpl.shutdown();
                    }//end the invalid personal verification to CA

            }//end checking true CA's public Key
            else{
                    System.out.println("Error in CA's public Key");
                    commImpl.shutdown();
            }//end checking false CA's public Key
```

EVCA:

```
        public byte[] getPubKey(byte[] pubKey) throws EVCAException {

            try{
            clientPubKey=pubKey;
            return RSApublicKey.getEncoded() ;
                    }
                    catch(Exception e){return null;}

  }

        public String getEncMessage(byte[] encMessage) throws EVCAException {
        try{
        //RSA Decryption

                RSAKeyParameters                                                paramRSA1=new
RSAKeyParameters(false,RSAprivateKey.getModulus(),RSAprivateKey.getPrivateExponent() ) ;
                RSAEngine engRSA1=new RSAEngine();
                engRSA1.init(false,paramRSA1);
                byte[] decryptedData=engRSA1.processBlock(encMessage,0,encMessage.length);


                returnString = new String(decryptedData);
                return returnString;
                }
                        catch(Exception e){return null;}
  }
        public byte[] getOkay(int ack) throws EVCAException {
                returnString = "This is the encrypted message to test client's public Key";
                try{
//RSA encryption using clients's public key
```

```
                byte[]data=returnString.getBytes();
                RSAPublicKey                                pubkey                                =(RSAPublicKey)
KeyFactory.getInstance("RSA","BC").generatePublic(new X509EncodedKeySpec(clientPubKey));
                RSAKeyParameters                                                                  paramRSA1=new
RSAKeyParameters(true,pubkey.getModulus(),pubkey.getPublicExponent() ) ;
                RSAEngine engRSA1=new RSAEngine();
                engRSA1.init(true,paramRSA1);
                byte[] encryptedData=engRSA1.processBlock(data,0,data.length);
                return encryptedData;
        }
                        catch(Exception e){
                                        System.out.println(e);
                                        return null;}

 }
        public int getDec(String decMessage) throws EVCAException {
                ack = 1;

        //receiving decrypted message for checking
                if ("This is the encrypted message to test client's public Key".equals(decMessage))
                        return 1;
                else
                        return 0;

 }
        public byte[] getRand(String msg) throws EVCAException {
        try{
        //creating session key with client
                BigInteger k1=new BigInteger(128, new Random());
                BigInteger k2=new BigInteger(128, new Random());
                SecureRandom fixed=new SecureRandom();
                CipherKeyGenerator kg =new CipherKeyGenerator();
                kg.init(new DHKeyGenerationParameters(fixed , new DHParameters(k1,k2)));
                sessionKey=kg.generateKey();

        //RSA encryption using clients's public key
                byte[]data=sessionKey;
                RSAPublicKey                                pubkey                                =(RSAPublicKey)
KeyFactory.getInstance("RSA","BC").generatePublic(new X509EncodedKeySpec(clientPubKey));
                RSAKeyParameters                                                                  paramRSA1=new
RSAKeyParameters(true,pubkey.getModulus(),pubkey.getPublicExponent() ) ;
                RSAEngine engRSA1=new RSAEngine();
                engRSA1.init(true,paramRSA1);
                byte[] encryptedKey=engRSA1.processBlock(data,0,data.length);
                return encryptedKey;
        }
                        catch(Exception e){
                                        System.out.println(e);
                                        return null;}
 }
        public int getEncPriKey(byte[] encPriKey, byte[] encMessage, String where,

        String from, String to) throws EVCAException {
                ack=0;
                try{
                //decrypting secret Key using session key
                RC4Engine cp=new RC4Engine();
                cp.init(false,new KeyParameter(sessionKey));
                byte[] secKey=new byte[encPriKey.length];
                cp.processBytes(encPriKey,0,encPriKey.length,secKey,0);
                System.out.println("secret key extracted");
```

```
//decrytping message using secret key
cp.reset();
cp.init(false,new KeyParameter(secKey));
//byte [] encMessage=encMsg.getBytes();
byte [] decryptedData=new byte[encMessage.length];
cp.processBytes(encMessage,0,encMessage.length,decryptedData,0);
String message=new String(decryptedData);


//scan for virus patterns....

//if there is no virus....
ack = 0;

//if there is virus...
if(message.indexOf(virusPattern[0])>=0)
ack = 1;
if(message.indexOf(virusPattern[1])>=0)
ack = 1;
if(message.indexOf(virusPattern[2])>=0)
ack = 1;


//setting mail properties
Properties props = System.getProperties();
props.put("mail.smtp.host", where);
Session session = Session.getDefaultInstance(props, null);


        Message msg = new MimeMessage(session);
        msg.setFrom(new InternetAddress(from));
        msg.setRecipients(Message.RecipientType.TO,
                                                InternetAddress.parse(to, false));
        msg.setSubject("Test Message");
        if (ack==0)//if mail has no virus sending the message in attachment
        {
                msg.setFileName("MSG1.enc");
                msg.setText(new String(encMessage));
                System.out.println("Message to Send :" +message);
        }
        else //if virus is found then send the virus notification
        {
                msg.setText("This email has virus");
                System.out.println("Message to Send :This email has virus");
        }
        msg.setHeader("X-Mailer", "Sample Email");
        //sending email
        Transport.send(msg);

        //writing into a file the encrypted data for verification at the receiver side and check with
decryption
        OutputStream f1=new FileOutputStream("C:\\MSG");
        f1.write(encMessage);
        f1.close();

} catch (AddressException ae) {
        ae.printStackTrace(System.out);
} catch (MessagingException me) {
}catch(Exception e){
        System.out.println(e);
}
return ack; }}
```