KEY ESCROW AND KEY RECOVERY LABORATORY MODULE FOR COMPUTER SECURITY AND INFORMATION ASSURANCE CURRICULUM

Brenda Hinkemeyer, Sharvari Siddula and Jayantha Herath
Computer Science Department
Saint Cloud State University
St. Cloud, MN 56301
{brenda@nikosha.net, siddulasharvari@yahoo.com, and
jherath@stcloudstate.edu}

Abstract

Key recovery involves either the recovery of an encryption key or the ability to decrypt a message with a different decryption key. Although there are many key recovery methodologies, they all share the same basic characteristics: gathering the recovery information, storing the recovery information, and recovering the information when needed. Very often students have difficulty understanding key recovery and escrow algorithms. To help students understand special features of those algorithms, a computer security learning module has been developed with hands-on classroom activities. The snapshots illustrate how to use this key recovery and escrow learning tool. This paper reviews basic key recovery and key escrow techniques, and outlines a laboratory module designed and implemented to demonstrate those concepts in an instructional setting.

1. Introduction

There is a national need for high-quality trained information system security professionals with the ability to learn in a short period of time and stay current with the information technology advances. Colleges can help to meet this demand by redesigning their information security courses, integrating security aspects in every possible course and allowing more students to succeed at the entry level. Successful application of active learning with rapidly changing technologies in the learning process is a way to remove the difficulties at entry level. Such changes will help the students to improve their skills. Consequently, both the public and private sectors will benefit with the greater number of highly-skilled trained professionals.

Traditionally, key recovery and other security related subject matter has been presented to a less than enthusiastic student body in a relatively passive classroom environment. To help students learn in a short period of time and stay current with the information technology advances it is important to redesign the information security courses allowing more students to succeed at the entry level. Successful application of active learning with rapidly changing technologies in the learning process is a way to remove the difficulties at entry level. Such changes will help the students to improve their skills.

In an ideal setting, parties involved in an encrypted message exchange would never lose their passwords (keys), nor would they ever leave. In fact, both of these happen all the time. Consider the following scenario: A customer deals with a sales representative from a company. The customer sends an encrypted order to the sales representative. Before the order is completed, the sales representative leaves the company. In order for the transaction to be completed without requesting a repeat order from the customer, the sales representative's supervisor or replacement must be able to read the message sent by the customer. This can only be done if one of these parties was included in the original transaction, or if they can recover the encryption key used.

Key recovery or escrow is a process by which some key may be recovered, allowing a message to be decrypted. In many cases, a third party is entrusted with the key. They are known as escrow agents or key recovery agents [7]. Key recovery may be used for anything from a backup mechanism for business records to law enforcement to government spying [8]. Note that this does not involve breaking or reverse engineering a key; rather a key is held in trust. In many cases, the actual keys are not held by any one party. Rather, the data recovery key is a different value that is used for determining the encryption/decryption keys [16]. Basically, key recovery is useful in situations where the private key of a party is lost or forgotten, or perhaps the public key used when encrypting the message has expired [9]. To extract a message encrypted in any of these situations, key recovery is necessary.

This paper outlines and compares several different key recovery systems and presents a teaching tool for engineering, computer science and information systems undergraduates to help them understand key escrow and recovery concepts in an instructional setting. Section two of this paper discusses several different methodologies for key recovery.

Section three compares the techniques and highlights concerns with key recovery in general. Section four describes the laboratory module developed and outlines the user documentation and demonstration. Section five highlights conclusions and future work for this project.

2. Key Recovery Methodologies

There are a number of algorithms, proposals, and commercial products available that describe or implement key recovery using key escrow. Most key escrow systems have three main components: a User Security Component (USC), a Key Escrow Component (KEC), and a Data Recovery Component (DRC). The USC encrypts and decrypts the data. It often incorporates a Data Recovery Field (DRF) in the message to facilitate key recovery. The KEC stores the data recovery key(s). The DRC performs the recovery of the plaintext from the encrypted text using the piece(s) from the KEC and the DRF set up by the USC [16].

Key recovery requirements desired by governments include (1) Access without knowledge or consent of the users involved, (2) Omnipresent adoption, and (3) Speedy retrieval [8]. Law enforcement demands access to the encrypted communication data as well - something not inherent for any commercial needs [8]. These requirements are at odds with some of the goals of the original encryption. The following sections outline techniques that are representative of the different methodologies currently available.

2.1 Multiple Agent Based Scheme

Lim, Kang, and Sohn [9] proposed an architecture for key recovery based on the Multiple Agent Based scheme. The keys are encapsulated with the message and sent to a randomly selected, secretly chosen recovery agent. The keys are reconstructed after the recovery request is certified. The basic scenario consists of three phases: (1) initialization, (2) communication and key recovery information generation, and (3) key recovery. In phase 1, public keys are set and distributed for all agents. These keys are distributed with certificates to authenticate them. In phase 2, a key agent randomly chooses a key recovery agent (KRA) to use. The key recovery information is generated using the public keys of the key recovery center (KRC) and the KRA. This information is used inside the message along with the cipher text. In phase 3, a user realizes the need to perform key recovery. The authorization is certified by an agent. This agent contacts the KRC. The KRC gathers the key recovery information (KRI) from the appropriate KRA(s) and returns the information to the user.

2.1 Proactive Secret-Sharing

Numao [10] put forth a key registration system based on distributed secret sharing called Proactive Secret-Sharing. Rather than registering the key with a single key recovery agent, multiple agents are involved. The key cannot be retrieved or reconstructed without some minimum number of these agents. In this way, no single point of attack is available for exploitation. This is commonly called a *(t,n)*-threshold scheme, as t servers (or agents) must cooperate to recover the key. In the proposed scheme, the secret is never revealed when the message is decrypted. Proactive secret-sharing means that the sharing is planned and performed at the time the key is created. This secret sharing scheme consists of 3 basic steps: key registration, secret sharing, and key recovery. For the key registration step, the session keys used for specific messages are encrypted with the public key of the key recovery agent. This information is attached to that message. In this case, the key escrow is performed by the user(s) who hold the message, rather than a single escrow agent. The user retains privacy while still allowing for key recovery. For the secret sharing step, the secret, after encryption, is split up and distributed among a number of different agents. These agents have no information with which they alone can reconstruct the key or decrypt the message. When key recovery needs to be performed, the user sends a request for reconstruction of the secret to at least $t+1$ servers, along with the encrypted key portion of the message. These servers each authenticate the request, and apply their pieces of the secret to the encrypted key to partially decrypt it. The user uses the decrypted key to decrypt the message. This is based on an ElGamal encryption scheme.

2.3 Key Recovery Entry

Al-Salqan [12] proposes the Key Recovery Entry scheme. It is a key recovery system that adds a small field to a message when it is transmitted. It also relies on a Certificate Authority (CA) body to authenticate requests and provide a key that opens the extra field. The author breaks his approach into four cases: (1) no key recovery, (2) recovery of session key, (3) recovery of private keys, and (4) PGP key recovery. For the first case, the normal encryption and decryption steps are performed for the algorithm used. A session key is used, along with private keys for each user. Both the private key and the session key are needed to decrypt the message. If either key is forgotten, the message cannot be recovered. For the second case, a CA is used to uniquely identify individuals. These certificates are retained in a database. The CA stores the answers to a series of questions such as mother's maiden name, etc. The public key is stored with the answers. When an encrypted message is sent, an additional field is added to the message. This is the key recovery entry (KRE). It includes the session key and the public key of the recipient, as stored with the CA. If the recipient has lost his private key, the message may still be decrypted by contacting the CA and responding to the stored questions. If the answers match, the session key is provided. For the third case, a session key does not exist that will allow decryption of the message. Whenever a private key is created or updated using a key package, a key recovery file is also created. This file contains the private key, secured with a password created by a hash of the verification answers stored

with the CA. The CA holds the key for decrypting this file. If the private key is forgotten, the recovery file is used to retrieve the key. In the final case, there exists no CA. In this case, a lightweight certificate authority (LCA) must be added to the system. It is responsible for storing the public key and verification answers. Key recovery is similar to the cases already discussed.

2.4 Clipper

McConnell and Appel [11] provide a proposal for key management prepared for a governmental body. They called it Clipper. They propose an infrastructure based on public key cryptography, where each user has one or more pairs of public and private keys. It is similar to the Multiple Agent Based scheme already discussed, except that certificates are used for authentication. This is also a hardware-based system. After a user generates his public and private keys, he registers with a Certified Escrow Authority (CEA) by distributing both the public and private components of his keys. The user receives a public key certificate digitally signed by the CEA using its private key. This binds the user's identity to the public key for authentication. In the process of obtaining a certificate, the CEA obtains sufficient information from the individual to verify his/her identity. When a message must be sent, the user generates a session key and encrypts the message with that key. The message is also encrypted with the recipient's public key exchange session key provided by the CEA. The result of this encryption is stored along with the header portion of the encrypted message for recovery by the recipient or the government. If the private key is lost later or the government needs access to the data, decryption of the information is accomplished by requesting a release of the private key exchange from the CEA after appropriate authorization is received.

2.5 Threshold for RSA

Okamoto [13] proposes a technique similar to the Proactive Secret-Sharing technique called Threshold for RSA. His system is based on RSA as opposed to ElGamal used in Proactive Secret-Sharing. Since key recovery for an RSA encryption algorithm is based on factoring, it is very difficult. The author's system is a threshold key recovery system, so some subset of key holders is needed to reconstruct the key. The author proposes three schemes. In the first scheme, the user generates the public/private key pairs for RSA comprised of two primes. The trustees generate additional values and send them to the user. The user generates random numbers. These values are sent back to the trustees. The trustees validate the values. When recovery is needed, the user requests a specific value from each trustee, and recalculates the key. The other two schemes are similar to the first with different pieces exchanged between the user and the trustees.

2.6 Commercial Key Recovery

Walker, Lipner, Ellison, and Balenson [14] took an approach where each individual expects a right to privacy even from the government. They called it Commercial Key Recovery. They also explain that government key escrow (as described in the Clipper scheme earlier) doesn't help an individual user who loses his key. It only ensures that the government will always have access to the keys. They built their proposal in two steps: a Clipper Software Key Escrow, and a Commercial Key Recovery (CKR) System. The Clipper Software Key Escrow step involved developing a software version of the Clipper design. It uses an asymmetric key for the key escrow functions, where the government holds the private key component. It builds a Law Enforcement Access Field (LEAF) using the public key portion. This portion is used for gaining access to the private key, as well as authenticating the message. The authors contend that their approach provides better security, and is more tamper-resistant than Clipper. The CKR system provides emergency access to the encrypted data by obtaining a key from a Data Recovery Center (DRC) using the Data Recovery Field (DRF). The DRC is generally established by a commercial entity. It may be internal to an organization or it may be an external bonded organization. When a user installs a recovery-enabled (RE) program, it is registered. Registration involves the user providing information to be used for authentication in the event a lost encryption key ever needs to be recovered. In return, the DRC sends its public key to the user's program. The private key is held secretly by the DRC. When the user runs an RE program, it encrypts messages using a session key and the DRF. The DRF contains the session key and user's identity encrypted with the DRC's public key and identifier. If an individual needs access to the encrypted data because a key has been lost, the DRF field is sent to the DRC. The DRC does not have the user's key, but it has a session key that can allow the message to be decrypted. There is no database of escrowed keys at the DRC or elsewhere.

2.7 DoD Key Recovery

The Department of Defense (DoD) [1] has also prepared a key recovery policy. Because they require certificates in all communications, this policy applies to the key pair used to acquire the certificate. They use a key escrow system consisting of four parties: a Subscriber, a Requestor, a Key Recovery Official (KRO), and a Key Recovery Agent (KRA). The Subscriber is the original key holder who submits the key for escrow. A Requestor makes the request for the escrowed key. The KRO receives the recovery request, verifies the identity and authorization of the requestor, and interacts with the KRA to fulfill the request. The KRA interacts with the key escrow database. There is an established hierarchy for determining which parties can make requests for keys.

2.8 NIST Key Recovery

The final methodology comes from the National Institutes of Standards and Technology (NIST) [2]. It is a demonstration project to illustrate key recovery. It is meant to be a standard that could be used in federal projects (not necessarily DoD projects as with the previous methodology). It relies on a Certificate Authority (CA) hierarchy, a Key Recovery Agent (KRA), and an Organization Registration Authority (ORA). The CA verifies the public keys of the various users in the system. The KRA performs the key recovery. This may or may not involve an escrowed key. The ORA receives the recovery request and orchestrates the recovery process. There are four different methods describing the interactions of these parties. For Method A, the keys are stored with the KRA. For Method B, the KRA has access to separate components of the key that are used to reconstruct it. For Method C, a session key is encrypted with the KRA's key.

3. Comparison of Key Recovery Techniques

The Multiple Agent Based and Clipper techniques rely on distributed key escrow agents holding the key. Proactive Secret-Sharing and Threshold for RSA distribute and reconstruct keys from pieces. Key Recovery Entry and Commercial Key Recovery use session keys for recovering the message as opposed to recovering the key itself. The DoD technique (similar to the Clipper technique) relies on an established hierarchy for establishing and retrieving the keys (held in a key database). The NIST technique is a simpler model of all of these methods. Although these techniques do have differences, they all still follow the general process outlined in the Methodologies section.

While key recovery/key escrow is useful in a number of valid situations (such as the scenario described in the Introduction), it is also dangerous. The ability to recover a key may negate the benefits of the security provided by the key. If the keys are stored in a database somewhere, the database could be compromised, allowing an unauthorized person access to the keys [6]. Also, authorizations may be forged [8]. In the case of third party escrows, a tremendous amount of trust is placed in these parties [15].

Falling into both the good and bad realms, key escrow may also provide governmental bodies and law enforcement with access to encryption keys [7]. While it is useful for criminal investigations, it is also a serious invasion of privacy when misused. In addition, any infrastructure added to enable key recovery must be secured or it will be vulnerable to attack and misuse [8]. One organization pushing to establish key recovery in all cryptographic applications is the Key Recovery Alliance (KRA). Their goal is to develop a standard approach for allowing any governmental body access to the information created by such an application. Adding to the concerns already stated are privacy concerns that come with widespread government access [8] [15]. Many in the cryptography field, including Abelson, et. al. [8] and Paine [7], insist that the proponents of key recovery provide the overwhelming proof that the benefits outweigh the risks.

Abelson, et. al. [8], outline three impacts associated with the addition of key recovery to any system. These items are risk, complexity, and economic cost. Risk refers to the risk of failure of the key recovery system in a critical situation. Complexity refers to the additional complexity added to any system to implement key recovery. Economic cost refers to the burden of adding key recovery to a system. All three of these impacts will weigh negatively on any implementation. To illustrate this point, the authors offer the following statistics: There are thousands of products on the market; there will be thousands of key escrow agents deployed around the world; there are tens of thousands of law enforcement agencies wanting access; there are millions of users of encryption; there are tens of millions of user keys, and there are hundreds of billions of temporary session keys. A single infrastructure to handle all of these uses would be difficult indeed. This is emphasized by Neumann [15] as well.

Finally, in cases where there are a large number of users and a large number of messages, the escrow agents could clearly become a bottleneck. For security reasons, the number of trusted agents would be limited. If every message needs to pass through these agents, communication could be slowed greatly [9]. This depends on the type of escrow used and whether all keys are escrowed.

4. Laboratory Module

This project demonstrates key escrow, encryption, and key recovery in a format suitable for a laboratory module. The application has a single user interface (shown in Figure 1). Three clients are started, along with a trustee. The third client (designated as "Supervisor C" on the user interface) is also set up as the supervisor of clients A and B. This allows Client C to retrieve the key for either client from the Trustee. The clients communicate with each other directly. The application initiates the actions with the clients and/or the trustee as requested by the user through the user interface. Any messages to the user appear at the bottom of the user interface screen. The structure and procedure of this project were based on the DoD model [1] and Method A from the NIST model [2]. Both are described in the Methodologies section. The source code is available from the authors by e-mail or downloading from [17].

The messages may be encrypted using Advanced Encryption Scheme (AES) Encryption [3, pp. 145-164] or RSA Encryption [3, pp. 268-274]. The authorizations to verify a client's identity are encrypted using the RSA encryption scheme. The basis for the code that implements the AES Encryption was borrowed from Wagner [4]. The basis for the code that implements the RSA Encryption was borrowed from Johnson [5]. Both implementations follow the steps for their algorithms exactly as described by Stallings [3]. Both encryption schemes use 128-bit encryption. AES is a symmetric encryption scheme. There is a single key shared between the parties who need to communicate. RSA is an asymmetric encryption scheme. There are public and private components to the keys needed to encrypt or decrypt a message. The steps performed in the actual project demonstration are outlined in the following paragraphs.

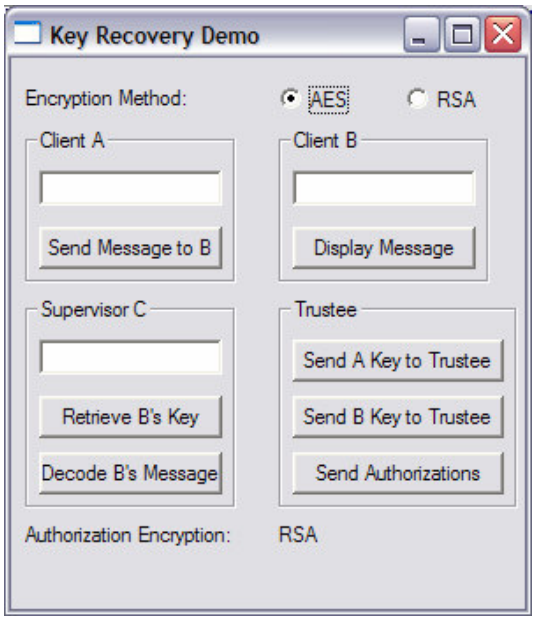

Figure 1: Key Recovery Demo User Interface

When the application is started, the clients and the trustee are created and started. RSA key sharing is performed between the clients. In the description below, "key escrow" refers to the step in which the key is given to the Trustee. "Key recovery" refers to the step in which the key is recovered by an authorized party.

4.1 Key Escrow

The first part of the demonstration involves setting up the key escrow. This is shown in Figure 2. AES Encryption should be selected. The key to be used is entered in the edit box for Client A (shown as "the key is 88888"). It needs to be 16 characters long. The button "Send A Key to Trustee" is pressed. This allows the application to send the key to Client A. Client A will also escrow its key with the trustee. Next, the key to be used for Client B is entered in that edit box. It also needs to be 16 characters long, and should be the same value as the key used for Client A. The button "Send B Key to Trustee" is pressed. This allows the application to send the key to Client B. Client B will also escrow its key with the Trustee.

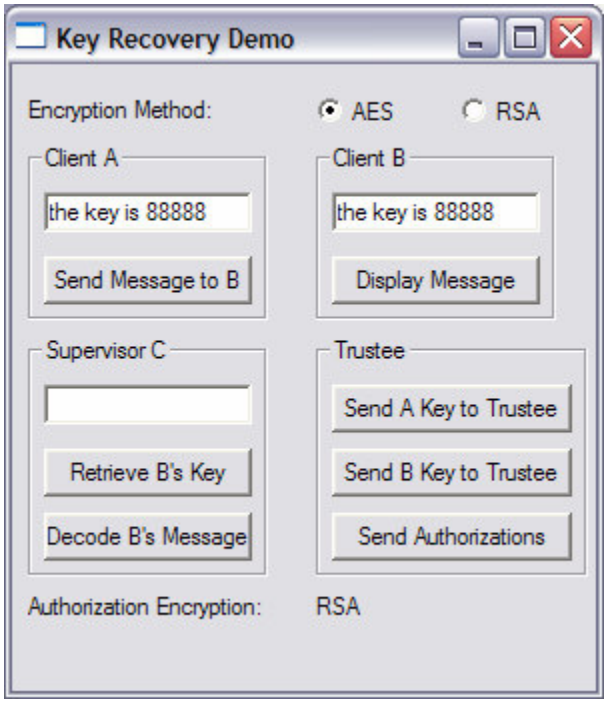

Figure 2: Set up key for Client A and Client B

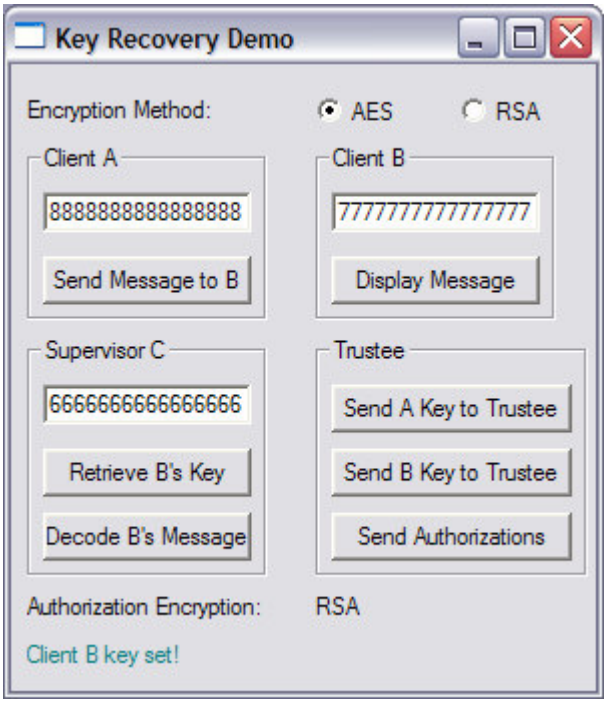

Figure 3: Set up authorization codes for Client A, Client B, and Client C (Supervisor)

For the RSA portion of the key escrow, the clients exchange keys when they are set up. It is not part of the key escrow demonstration.

All three clients also need to set up their digital signatures (or authorization codes) with the trustee. This is used to verify the identity of any specific client. It is compared when a client requests the key of another client from the trustee. Figure 3 illustrates this process. The authorization code for each client should be entered in the appropriate edit boxes. (The authorization codes must be all numeric.) When there is an authorization code in all three edit boxes, the button "Send Authorizations" is pressed. Each client will store its authorization code and contact the Trustee to save the code for later authentication. The authorization codes are encrypted so the Trustee does not actually see them. In this example, Client A is identified by the signature "8888888888888888", Client B by the signature "7777777777777777", and Client C by the signature "6666666666666666".

4.2 Encryption

The second part of the demonstration is encryption. The encryption scheme is selected: either AES or RSA. For the AES demonstration, the key must have been provided for both clients A and B as provided in the key escrow part of the demonstration. For the RSA demonstration, no additional key setup is necessary.

Refer to the left-hand side of Figure 4. The message is entered into the edit box for Client A (shown as "the secret is 88"). The button "Send Message to B" is pressed. Client A sends the message to Client B. Client B displays the raw message (which will be a cipher) in its edit box, as shown on the right-hand side of Figure 4. Pressing the button "Display Message" tells Client B to decrypt the message and display it in its edit box. As shown in Figure 5, the message displayed by Client B should match the original message entered by Client A. This is shown in Figure 5.

Figure 6 demonstrates a failure condition. If the key for Client B is modified to be different from the key used by Client A for encryption, the decryption will fail. To test this, a new key for Client B is entered into the edit box for Client B, as shown on the left-hand side of Figure 6. The "Send B Key to Trustee" button is pressed. Now when the button "Display Message" is pressed again, the message shows up as garbled, as shown on the right-hand side of Figure 6. Before continuing on to the key recovery part of the demonstration, the key must be reset to the same value used by Client A by sending it again as described earlier.

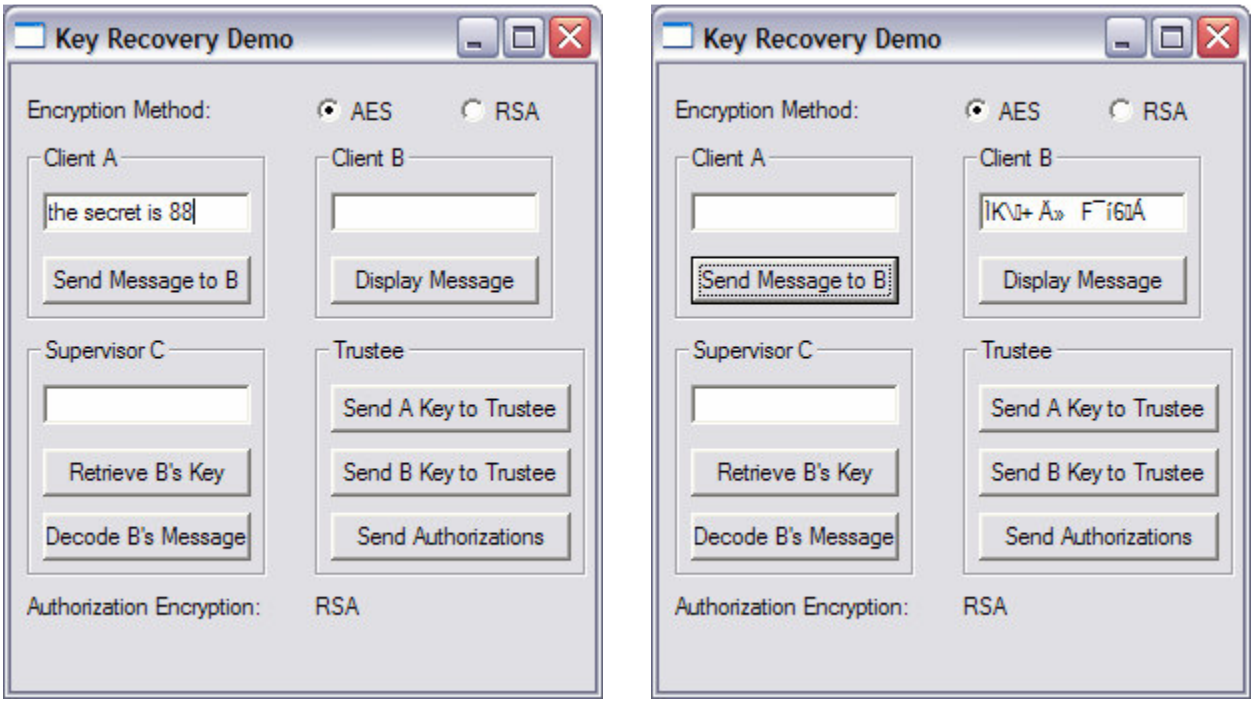

Figure 4: Client A enters the message for Client B and Client B displays the message received (encrypted)

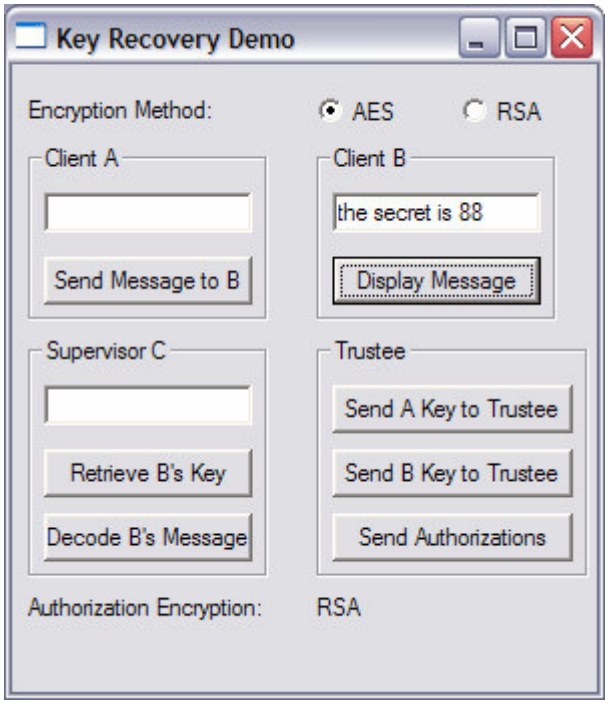

Figure 5: Client B displays the decrypted message

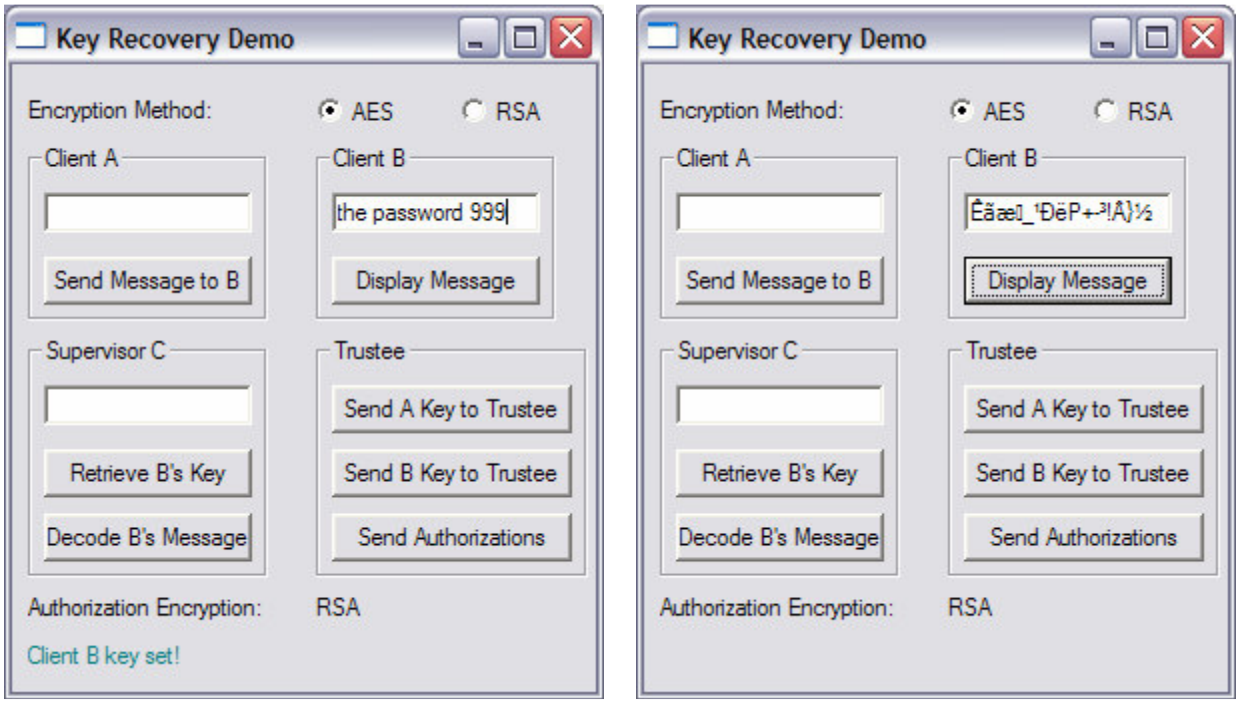

Figure 6: Client B sets up the wrong password and Client B uses this wrong password to decrypt the message (improperly)

The experiments for encryption are repeated in the same manner with the RSA encryption scheme. Note that the passwords for RSA must be numeric.

4.3 Key Recovery

The final part of the demonstration is key recovery. A client places a request with the trustee to retrieve another client's key. If that client is authorized to retrieve the key, and the identity of that client is verified, the key is given to the client. In this example, Client C has been set up as the supervisor of Client B; therefore, it is authorized to retrieve the key of Client B.

Referring to Figure 7, the authorization code for the Supervisor ("6666666666666666") is entered into the edit box. The button "Retrieve B's Key" is pressed. Client C encrypts the authorization code and sends it to the Trustee with a request for the key of Client B. After receiving the key from the Trustee, Client C stores the key and displays it in its edit box. This is shown on the left-hand side in Figure 7. Upon pressing the button "Decode B's Message", Client C requests B's message. It then applies the key it was given from the Trustee to decrypt the message. This is shown on the right-hand side in Figure 7. The decrypted message is displayed in the edit box for Client C. The encrypted message is displayed at the bottom of the application, for comparison purposes (it looks like nonsense characters). If the authorization for Client C was not verified, the Trustee will deny the request for the key of Client B as shown in Figure 8.

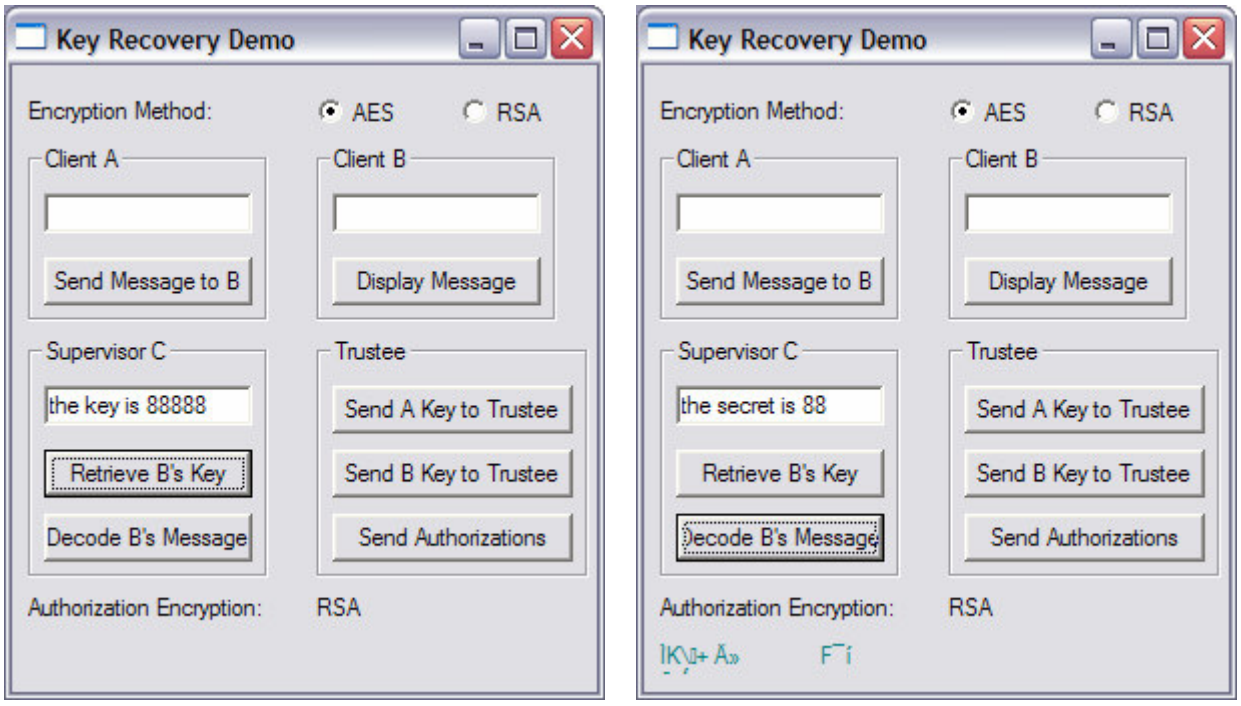

Figure 7: Supervisor C retrieves the key for Client B and uses it to decrypt the message that B holds

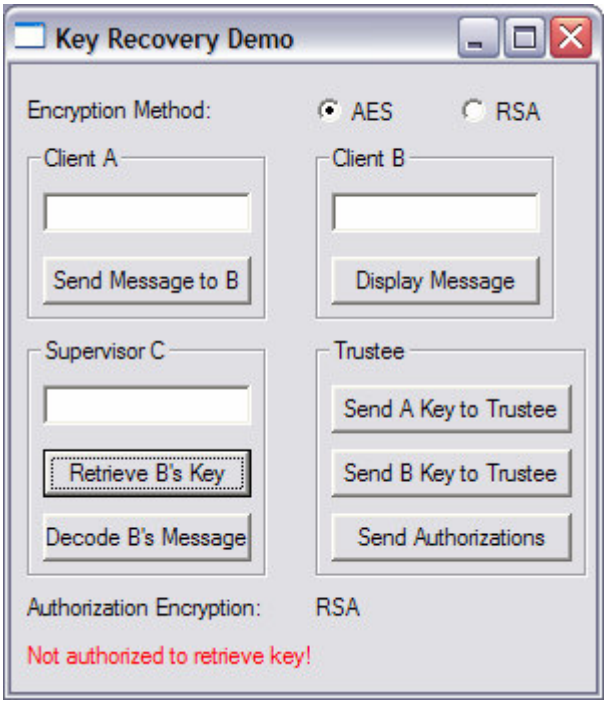

Figure 8: Supervisor C does not provide the proper authorization code to retrieve the key for Client B

All three components of the demonstration are shown in a simple form meant to illustrate the concepts and show that they work. Additional complexity may be added by modifying the types of communication between the clients, performing key recovery and message transmission between any two clients, and adding additional types of key escrow to the Trustee functions. In addition, as pointed out in several of the key recovery algorithms detailed, a single trusted point of contact may not be the most secure model. Additional trustees could be incorporated as well.

5. Conclusions and Future Work

Key recovery is a very important research area. The government wants access to user keys for law enforcement and spying. Individuals and corporations want access to user keys in the event that the key is lost or forgotten. Privacy advocates do not feel that government access should be the overwhelming concern in a key recovery system. Government law enforcement advocates feel that law enforcement access should be quick, easy, and untraceable. Because the privacy advocates and governmental bodies are at odds with each other, it is unlikely that any one system will be in use everywhere, unless it is forced by law.

Several methods were reviewed in this paper that attempt to satisfy all parties. Each of the methods is different, but they share many common structures. This paper described a teaching tool for engineering, computer science and information systems undergraduates

to easily learn key recovery concepts. This active learning module can be easily adapted to teach other key recovery algorithms to help the students understand quickly in preparation for advanced studies. This laboratory module successfully demonstrates the capture of a key by a trusted third party. Later the key is returned to an authorized party for use in decrypting a message.

This project could be expanded in a number of ways. Additional encryption methods could be incorporated. The clients could be located on different servers or even across the Internet. The number of clients/supervisors/trustees could be configurable, as opposed to hard-coded for a specific demonstration. Following the NIST [2] model, either of the other two methods (Method B or Method C) could be implemented as well. These other methods employ different escrow bodies and components.

References

[1] DoD Public Key Infrastructure Program Management Office, Key Recovery Policy for the United States Department of Defense, Version 3.0, http://www.defenselink.mil/nii/org/sio/ia/pki/DoD_KRP_V2.0_31May2002.pdf, Aug. 31, 2003.

[2] Computer Security Resource Center, National Institute of Standards and Technology, Key Recovery Examples, http://csrc.nist.gov/krdp/exa.html.

[3] Stallings, W., *Cryptography and Network Security*, Upper Saddle River, NJ: Pearson Education, Inc., 2003, pp. 145-164, 268-274.

[4] Wagner, N. R., The Laws of Cryptography with Java Code, http://www.cs.utsa.edu/~wagner/lawsbookcolor/laws.pdf, Jan. 24, 2002.

[5] Johnson, P., Java RSA Code, http://pajhome.org.uk/crypt/rsa/implementation.html, June 6, 2004.

[6] Schneier, B., Security Pitfalls in Cryptography, http://www.schneier.com/essay_028.html, 1998.

[7] Paine, T., Important Information About PGP & Encryption, http://proliberty.com/references/pgp/, Apr. 22, 1998.

[8] Abelson, H., Anderson, R., Bellovin, S. M., Benaloh, J., Blaze, M., Diffie, W., Gilmore, J., Neumann, P. G., Rivest, R. L., Schiller, J. I. and Schneier, B., The Risks of Key Recovery, Key Escrow, & Trusted Third Party Encryption, http://www.cdt.org/crypto/risks98/, 1998.

[9] Lim, S., Kang, S. and Sohn, J., "Modeling of Multiple Agent based Cryptographic Key Recovery Protocol," *Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC 2003)*, Las Vegas, NV, pp. 119-128, Dec. 2003.

[10] Numao, M., "A Secure Key Registration System Based on Proactive Secret-Sharing Scheme," *Proceedings of the Fourth International Symposium on Autonomous Decentralized Systems,* Tokyo, Japan, pp. 230-237, Mar. 1999.

[11] McConnell, B. W. and Appel, E. J., Interagency Working Group on Cryptography Policy, Enabling Privacy, Commerce, Security and Public Safety in the Global Information Infrastructure, http://www.cdt.org/crypto/clipper_III/clipper_III_draft.html, May 20, 1996.

[12] Al-Salqan, Y. Y., "Cryptographic Key Recovery," *Proceedings of the 6th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS '97)*, Tunis, Tunisia, pp. 34-37, Oct. 1997.

[13] Okamoto, T., "Threshold Key-Recovery Systems for RSA," *Proceedings of 1997 Security Protocols Workshop*, Paris, France, pp. 191-200, Apr. 1997.

[14] Walker, S. T., Lipner, S. B., Ellison, C. M. and Balenson, D. M., "Commercial Key Recovery," *Communications of the ACM*, vol. 39, no. 3, pp. 41-47, Mar. 1996.

[15] Neumann, P. G., "Crypto Key Management," *Communications of the ACM*, vol. 40, no. 8, p. 136, Aug. 1997.

[16] Denning, D. E. and Branstad, D. K., "A Taxonomy for Key Escrow Encryption Systems," *Communications of the ACM*, vol. 39, no. 3, pp. 34-40, Mar. 1996.

[17] http://www.nikosha.net, under "Brenda" and "Key Recovery Source Code", Feb. 27, 2005.