GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Security Essentials (Security 401)"
at http://www.giac.org/registration/gsec

Strategies for Ensuring Data Accessibility When Cryptographic Keys Are Lost

Arne Grimstrup

May 26, 2001

GIAC Security Essentials Practical V1.2e

Introduction

Private and public key cryptosystems are very effective in limiting access to those who hold the decryption keys. As long as the cryptosystem and keys have not been compromised, the data remains secure indefinitely. Unfortunately, the strength of the cryptosystem used also works against the user. If the cryptographic key is lost or forgotten, the data effectively becomes irretrievable to all.

This paper presents three strategies for preventing data loss due to lost encryption keys: user keys, key escrow and threshold schemes.

Problem

As any system administrator knows, users forget their passwords. Since they are prohibited from using the system, the organization usually suffers a small loss of productivity until the password is reset. In most cases, the cost of this failure is relatively small and is generally considered a nuisance. The recovery action is also quite simple; replace the value generated by the one-way function with one generated from a known text, give that known text to the user, and have the user login and choose a new password.

Access to encrypted data is controlled by giving the key to the authorized users. Since only the key holders can see the information, unauthorized access is prevented. The encrypted information also remains accessible as long as at least one of the authorized users remembers the key. Unfortunately, that group of authorized users does not remain stable over the time. People leave the organization, are transferred to other projects and departments, or no longer require access to the data. The key value will have to be changed each time a user loses the authority to access the data in order to maintain security. These key changes can overwhelm the users' ability to manage their keys and thereby increase the risk of key loss.

A forgotten or lost cryptographic key can be catastrophic to an organization. Since all of the security of the cryptosystem resides in the key, it cannot be replaced like a password. Thus, important data, such as accounting or medical records, that has been encrypted with that key becomes unaccessible. Without this information, the organization may not be able to continue operations resulting in severe financial losses until the data can be reconstitued from other sources.

Solution 1: User Keys

One approach to ensuring that encrypted data remains accessible is to use indirection. Instead of giving the authorized users the *master* or *file* key to the encrypted information, each user is given a unique *user* key with which they can retrieve the data. The master keys can then be stored in some secure place or a special archive user can be given access to the data via their own key.

Data access in a user-key system is a two-stage process. The user presents her key (UK) to the access manager. The access manager then verifies the key and retrieves the appropriate master key (MK) for the data requested. The master key is then passed to the file manager which decrypts the data and returns it to the user.

The Transparent Cryptographic File System(TCFS)[3] uses user keys to provide transparent access to encrypted data. The TCFS kernel module, with assistance from the *xattr* daemon, acts as both access and file manager. To access encrypted data, the user provides her password via the *tcfslogin* command to the kernel which retrieves the master key from a database. If the user is the owner of the secure file and the file resides on the local disk, the file is read and the decrypted data is passed to the user's application program. Remote data is transferred as cyphertext via NFS to the user's machine. Similarly, data is encrypted before being written out to the filesystem.

Because the master keys that are stored in the database are encrypted using the user's password as the key, they may be lost through inadvertent use of the *passwd* command. In order to ensure that master keys remain accessible, TCFS provides a replacement called *tcfspasswd* that updates the database entries in addition to the password file.

There are two main weaknesses in the TCFS approach: single point of failure and unnecessary duplication of files. Since all data access is dependent on the master key database, any corruption of key data or destruction of the database causes some or all of the files to become inaccessible. The master key database becomes a single point of failure and is inviting target for those who wish to harm the organization.

Also, since access to encrypted data is granted only to the file owners and files are encrypted on a per-user basis, any shared files must be duplicated. Maintaining synchronization of multiple copies of a file is difficult and requires considerable expenditure of resources.

A more flexible user-key approach was proposed by Chick and Tavares[4]. In their system, a small prime number for each service is chosen by a *trusted central authority*(CA) and a unique *service number* and *service key* is computed for each service. A user key is then computed using the small prime numbers for each service the user is allowed to access and is issued to the user. To access a service, the user simply computes the service key from his user key and the service number.

Chick and Tavares' approach relies heavily on the CA to manage the service and user keys, and is not suited to dynamic environments such as file systems. Halevi and Petrank[7] addressed these shortcomings through the use of lists of users, files and access authorizations. The CA still manages these lists, collectively referred to as the *authorization structure*, but tasks like adding files or access authorizations may be delegated to the users. This delegation allows new information and permissions to be added quickly and easily to the system, while eliminating the overhead of duplicate files.

Solution 2: Key Escrow

Another way of maintaining access to encrypted files is to leave a copy of the decryption key with a *trusted third party*(TTP). If the key is lost or the key holders are unavailable, the decryption key can be retrieved from the TTP and the data can then be decrypted. This approach, known as *key escrow*, can be useful in a number of situation such as those identified by Maher[8]. They include:

- recovering data after the loss of the decryption key;
- granting access to encrypted data by authorized users when a key holder is unavailable, and
- granting access to authorized users when the key is being withheld without just cause.

A major weakness in key-escrow systems are the TTPs themselves[1]. Once the key has been released, whether deliberately or accidentally, the encrypted data is no longer secure. Therefore TTPs, whether internal to the organization or external service providers, must flawlessly perform their duties in order to ensure the security of the keys entrusted to their care. The TTPs also become targets for social engineering or other attacks aimed at acquiring keys, so the security of user data effectively becomes equivalent to the strength of the TTP's key store security.

Blaze[2] addresses this weakness of key escrow by giving the TTP a tamper-resistant smartcard containing the decryption key rather than the key alone. When the escrow key is required, the smartcard is retrieved and inserted into a smartcard reader. The card then decrypts the cyphertext sent to the TTP without revealing the key and records the decryption operation in secure memory-based audit trail.

The use of a smartcard improves key-escrow security in several ways. Since the card resists tampering, it is more difficult for malicious third parties to retrieve the key, which limits the number of copies of that key in circulation. The user can verify that the TTP is acting in accordance with the terms of the escrow agreement by reviewing the audit trail. Finally, if supported by the hardware, the keys can be erased after a certain period of time, which limits the duration of the threat posed by lost or stolen smartcards.

Denning's taxonomy[6] gives an excellent overview of the necessary components of a key-escrow system. She also provides a detailed listing[5] of currently available research and commercial systems.

Solution 3: Threshold Schemes

The decryption key need not be maintained as a single unit. Shamir[10] describes an approach that partitions a key into n pieces. When k or more of the n pieces are brought together, the original key value can be computed very easily. Otherwise, attempting to compute the key from $k-1$ or less pieces gives no useful result. This is called a *(k,n)threshold scheme*.

Threshold schemes have a number of properties that are useful in key recovery situations. Up to $n-k$ pieces can be lost without losing access to encrypted data since only k are needed to recover the key. The pieces contain no useful clues about the original key value, so some pieces maybe given to external TTPs without compromising security. Depending on the number of pieces created, new people can be added to the recovery group very easily. Finally, because of the number of pieces required to gain access to encrypted data, it is more difficult to subvert the security system.

A controversial threshold scheme implementation is the Escrowed Encryption Standard(EES)[9]. The EES was designed to protect communication while preserving law enforcement authorities' ability to wiretap and was intended to be embedded in telephones and other communications devices. Key recovery in the EES is based on a (2,2) threshold scheme; the decryption key is split into two pieces and both are required in order to reconstitute it. In this particular system, the user would not control any of the pieces. Instead, the key pieces would be escrowed with two separate TTPs when the device was manufactured.

A variation on the (2,2) threshold scheme is Maher's CryptoBackup[8]. In CryptoBackup, the user, Alice, designates a TTP, Bob, to act her backup agent. Bob provides Alice with a digitally-signed *Master Key Vector*(MKV), which she installs in her encryption key generator.

Each time Alice generates a new key, the key generator also generates a *Backup Key Vector*(BKV) that Alice includes in the header of her message. If the key is lost, Alice sends the cyphertext and the BKV to Bob. Bob then decrypts the message using the MKV and BKV to reconstruct the original key. Bob has no knowledge of the original key and cannot gain access to the plaintext without the BKV. Unlike Shamir's approach, the original key is not known when the first piece, the MKV, is generated. Instead, the key and the BKV are generated using the MKV as the base value.

Conclusion

Encrypting data can prevent unauthorized access to sensitive information. Unfortunately, if the decryption key is lost, the strength of the cryptosystem used prevents authorized users from gaining access as well. To address the problem of ensuring access, we examined three key recovery strategies: user keys, key escrow and threshold schemes. Alone or in combination, these strategies can reduce the risk of accidental data loss due to a lost decryption key.

Bibliography

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

[2] M. Blaze. "Key Management in an Encrypting File System."
In *Proceedings of the USENIX Summer 1994 Technical Conference*, pages 27-35, Boston, MA, USA, 6-10 1994.

[3] G. Cattaneo, G. Persiano, A. Sorbo, A. Cozzolino, E. Mauriello, and R. Pisapia. "Design and Implementation of a Transparent Cryptographic File System for UNIX."
Available at: http://www.tcfs.it/, 1997.

[4] G. C. Chick and S. E. Tavares. "Flexible Access Control with Master Keys."
In G. Brassard, editor, *Proc. CRYPTO 89*, pages 316-323. Springer-Verlag, 1990.

[5] D. E. Denning. "Descriptions of Key Escrow Systems."
Available at: http://www.cosc.georgetown.edu/~denning/crypto/Appendix.html, 1997.

[6] D. E. Denning and D. K. Branstad. "A Taxonomy for Key Escrow Encryption Systems."
Communications of the ACM, 39(3):34-40, 1996.

[7] S. Halevi and E. Petrank. "Storing Classified Files."
Available at: http://citeseer.nj.nec.com/137200.html, 1995.

[8] D. Maher. "Crypto Backup and Key Escrow."
Communications of the ACM, 39(3):48-53, 1996.

[9] National Institute of Standards and Technology (NIST).
FIPS Publication 185: Escrowed Encryption Standard, February 9, 1994.

[10] A. Shamir. "How to Share a Secret."
Communications of the ACM, 22(11):612-613, 1979.