Key Recovery for Archived Data using Smart Cards

Konstantinos Rantos* and Chris J. Mitchell
Information Security Group,
Royal Holloway, University of London,
Egham, Surrey TW20 0EX, UK.
{K.Rantos, C.Mitchell}@rhbnc.ac.uk

Abstract

Although many key recovery mechanisms have been proposed, the majority of them have been designed in the context of use with communicated data. The different requirements, however, that surround communicated and archived data make most of these mechanisms inappropriate for use on archived data. This paper investigates the business need for key recovery for encrypted archived data, identifies the requirements a key recovery mechanism should fulfil, and proposes a scheme where keys used for stored data encryption can easily be recovered.

Keywords: key recovery, smart cards, archived data, rogue user attacks, business requirements.

1 Introduction

With the development of cryptography and its growing use in protecting communicated and archived data, a critical issue has evolved concerning the loss of decryption keys. Assuming the use of secure mechanisms, loss of

*This author's work is supported by the European Commission (TMR Marie Curie Research and Training Grant ERBFMBICT983274).

keys means that decryption is infeasible, resulting in inaccessibility of data. Corporations will find such situations unacceptable, especially if the inaccessible data hold potentially valuable information. *Key recovery mechanisms* (KRMs) can be an efficient countermeasure to this threat. KRMs are mechanisms that allow authorised parties, under certain conditions, to retrieve the cryptographic keys used for data confidentiality, with the ultimate goal of recovering the cleartext data. An introduction to how these mechanisms work, and descriptions of existing KRMs and their properties, can be found in [1, 2, 3, 4, 5, 6].

The KRMs proposed so far have mainly been designed to provide *key recovery* (KR) for encrypted communication sessions. However, the different requirements for KRMs for communicated and archived data [7] demand the distinction between these two types of application. This paper examines the requirements that a KRM should fulfil when deployed for encrypted archived data in a corporate environment. Further to this analysis, a mechanism is proposed that meets these needs. More specifically, the mechanism gives authorised parties the ability to recover keys used for encrypting archived data with limited storage and processing requirements while preventing rogue user attacks. Unlike most KRMs, each user will be able to recover the keys used on the files he encrypted without the intervention of the *key recovery agent* (KRA), i.e. the trusted entity that assists in the recovery of keys, by preserving the appropriate key material.

2 The need for KR for encrypted archived data

The encryption of archived data typically involves a single entity, which stores and retrieves data at distinct points in time [7]. Unless complex key generation techniques giving a third party access to the generated key material are used, the entity that does the encryption is likely to be the only one in possession of the key material needed to decrypt the data. As a result, there are many circumstances that can result in the loss or inaccessibility of keys. These circumstances include both deliberate actions and accidents.

Deliberate actions originate both from outsiders and insiders. Experience seems to show that attacks are more likely to come from insiders than outsiders [8, 9], and hence the threat to the company from its employees cannot be ignored when enforcing security in a corporate environment. One of the dangers that a company might face is a disgruntled employee being the only holder of the keys used to decrypt business information. When requested to hand over the decryption keys, e.g. on termination of employment, he might refuse to do so, leaving the company with the potentially infeasible task of retrieving encrypted data without having the decryption key.

There are also certain accidental situations which might have the same unacceptable result. Consider, for instance, an employee that is away on vacation when the company requires the keys for the decryption of some of his data. Requesting the employee to reveal the password over an insecure telephone link, subject to eavesdropping, bears the risk of accidentally revealing it to unauthorised third parties. Under these conditions there should be an alternative means for the company to gain access to the keys necessary for decryption.

When keys are held in a storage device, e.g. a hard disk or a smart card, accessible by a password, the possibility exists that the device is destroyed, malfunctions, or is lost (not unlikely in the case of smart cards). The stored keys would then become inaccessible and, if a mechanism that allows recovery of the keys kept in the device does not exist, the data would be lost.

All these undesirable situations imply that the corporation, as the legitimate owner of all the company data, should have access to data decryption keys when necessary. KRMs can help overcome this problem, since they provide a means to access the key material necessary to recover the data decryption keys. This access will typically only be given to authorised entities, acting in accordance with a defined corporate security policy.

Although key recovery can be used as a countermeasure to the above threats, it can also be used to encourage employees to use encryption. Unless they are sure that data they encrypt can be easily recovered even if they lose the decryption keys, employees will be reluctant to use encryption, hence leaving their data unencrypted even though that information needs to be protected. Note that this paper does not attempt a detailed analysis of the various classes of KRMs; instead see [1] and [2].

3 Using existing KRMs with archived data

Previously proposed KRMs were mainly designed to provide KR functionality for encrypted communications. When these mechanisms are applied to archived data (especially key encapsulation schemes [2, 6]), they suffer from the absence of a second party that can verify the generated KR information. As a result, they become particularly vulnerable to rogue user attacks, where a rogue user can tamper with (alter or delete) the KR information during or after its generation, making it unusable to third parties.

An obvious countermeasure to this attack is to have an on-line agent which will contribute to the generation of all data encryption keys, while having direct access to, and keeping a backup of, all the generated keys. Alternatively, users could be required to escrow their master key or the file encrypting keys with a central agent. These solutions, however, typically demand a high-powered on-line server that may be involved in the key generation process, and its unavailability would prevent use of encryption. Furthermore, the administrative costs involved, including the security mechanisms needed for the protection of this information and the storage needs of all the escrowed keys, may make this solution quite unattractive, especially in small and medium-sized enterprises.

Another problem with most existing KRMs is that they do not offer the user who performed the encryption the ability to recover his keys unaided. That is, every time the user wants to recover a key previously used for encrypting data, he has to contact his agent, who will recover the required keys on the user's behalf. This problem arises from the fact that the intention of the majority of the proposed mechanisms was to give KRAs access to suspected encrypted communicated data. So the design was focused on giving access to the on-line agent, where the communication channel already exists, rather than the user himself. As a result, applying these mechanisms to encrypted archived data introduces extra communications costs, due to the required agent's interaction, and demands an on-line agent with the consequences mentioned above.

A key recovery mechanism that would be specifically designed for use with encrypted archived data while overcoming the aforementioned problems should typically satisfy the following requirements:

1. The KRA should have the ability to recover the required keys, even when the user tampers with the generated KR information.

2. The mechanism should give the user the ability to recover his keys unaided, i.e. without the KRA's intervention. This will ensure that the user has continuous access to his keys while avoiding the need to keep

a backup of them locally (an approach that introduces new threats to the secrecy of the keys).

Further to these requirements, it will be an advantage if the KRM makes no demands for an on-line server or storage of users' key related material.

A KRM designed for use with archived data was proposed by Maher, [10]. This mechanism, however, suffers from the problem that a user can tamper with the generated key recovery information and prevent key recovery by the agent. Moreover, the user cannot recover his keys without the agent's participation, a property that either necessitates the use of backup copies of keys or relies on the KRA's active support to access encrypted data.

Another scheme is proposed here that overcomes this problem. More specifically, for the proposed KRM a user does not require his agent's intervention to recover his keys. Further, it is not vulnerable to rogue user attacks on the generated KR information, and has no requirement for generated keys to be escrowed with the agent. Therefore, the proposed mechanism avoids both the vulnerability to rogue user attacks of key encapsulation mechanisms, and the requirement for storage and protection of user key material of key escrow mechanisms.

As far as rogue user attacks are concerned, for the purposes of this paper we assume that a rogue user, trying to disable authorised KR by his associated KRA, may tamper with the generated KR information by either altering or deleting it, or may even prevent its generation. However, we assume that the user will leave the encrypted data unchanged so that he can recover it when necessary (if the encrypted data is modified or destroyed, then no KRM can deal with the situation). For instance, the rogue user might simply detach the KR information from the encrypted file, which he will retain but not hand to the KRA. Through possession of this information the user can recover the required key, but the KRA cannot. Finally, note that no KRM can prevent a rogue user from deploying his own cryptographic infrastructure, and hence, such rogue user attacks are not considered in this paper.

4 A new KRM for encrypted archived data

We now describe the new mechanism for adding KR functionality to the encryption process for archived data.

Three entities are involved in the proposed mechanism: the *user* who encrypts the data, the KRA which assists in the management of keys, and the *authorised entity* AE which is the entity authorised by the corporate policy to have access to users' data.

Whenever a user wants to encrypt a file or message, instead of generating a random key for data encryption, he uses the proposed mechanism for key generation, which will also allow later recovery of the generated key. For this mechanism, which necessitates the use of a smart card by each user, the following requirements must be satisfied:

1. The KRA and the user's card share a *message authentication code* (MAC) function $f1$, a one-way hash function h, and a key generating function $f2$ (this could potentially be a one-way hash function). $f2$ is used to generate the key K_{AC} that will be used by the MAC function $f1$, i.e. $K_{AC} = f2(K_M, id_A)$, where K_M is the KRA's master key and id_A is user's A identity. K_{AC} should be stored on the user's card, typically during the card's personalisation, while the user must not have access to it to prevent rogue user attacks.

2. The KRA, user, and authorised entity share a key generating function $f3$. As with $f2$, $f3$ can be a one-way hash function.

3. The user's card and the AE share a secret key K_{AM} which is generated as a function of K_A and a secret master key K_{AE} that the authorised entity possesses, i.e. $K_{AM} = f3(K_{AE}, K_A)$. K_A is a master key specific to user A, which is generated as a function of KRA's key K_M and the user's identity id_A, i.e. $K_A = f3(K_M, id_A)$. As with K_{AC}, K_{AM} should also be stored on the user's card during the card's personalisation.

4. The user has access to a random number generator. The generated random numbers are used to ensure key freshness so that even a single file will not be encrypted with the same key more than once.

5. The KRA administers a file consisting of indexes binding a unique file identifier with a random value generated for the specific file. The integrity of this file and must be preserved.

6. All the entities trust the device where the encryption takes place, i.e. the user's PC or a server. If this is not the case then encryption has to

take place on the card, although there are certain performance limitations associated with this approach.

When user A wants to encrypt a file, a session key K_S has to be generated using the following protocol.

1. A generates a random value *RAND* either on his card or on the PC and using his card computes a MAC on *RAND* and the unique identifier *fileid* of the file to be encrypted, i.e. $MAC_1 = f1_{K_{AC}}(RAND, fileid)$. A then sends the following message to the KRA,

   $$\text{A} \xrightarrow{\quad idA \parallel RAND \parallel fileid \parallel MAC_1 \quad} \text{KRA}$$

   where $\parallel$ denotes concatenation.

2. Upon receipt of the message the KRA uses the received user's identity id_A and the master key K_M to compute the key K_{AC}. The KRA then recomputes the message authentication code MAC_1 using the received values *RAND* and *fileid* and checks the result against the received MAC_1. If the check succeeds the KRA adds an entry to the index file consisting of the received *fileid* and the random value *RAND*, indexed by the user who sent it. The KRA then computes a message authentication code MAC_2 on the hash of values *RAND* and *fileid*, i.e. $MAC_2 = f1_{K_{AC}}(h(RAND, fileid))$ and sends it back to the user's card.

   $$\text{A} \xleftarrow{\quad MAC_2 \quad} \text{KRA}$$

   This tells the card that the KRA has successfully registered the received random value *RAND* for the file identified by *fileid*.

3. As soon as the card receives MAC_2 it recomputes it using the values *RAND* and *fileid* that it sent to the KRA, and checks it against the received MAC_2. (Note that the computation of MAC_2 could take place while the card waits for the KRA's response.) If the check succeeds, the card uses the secret key K_{AM} and the random value *RAND* to generate the session key K_S as

   $$K_S = f3(K_{AM}, RAND)$$

which is passed to the PC for the encryption process. The file is encrypted using K_S and a key recovery field *KRF* is attached to it. The *KRF* consists of the random value *RAND* (the KRA's identity should also be included if there are multiple KRAs), i.e.

$$KRF = \{RAND\}$$

Should an emergency access situation arise, the authorised entity will request from the KRA the key K_A that corresponds to the user that performed the encryption. Having K_A and using the master key K_{AE} and the function $f3$ the authorised entity computes the corresponding user's key K_{AM}, i.e. $K_{AM} = f3(K_{AE}, K_A)$. Using K_{AM} and the value *RAND* attached to the file, the authorised entity can successfully recover the required key and the target data.

5 Properties and security analysis

With the proposed scheme, there is no need for interaction with the agent in everyday user access to the encrypted data, a property that simplifies the key recovery process for the user. More specifically, when a user wants to access a key that he has previously used to encrypt archived data, the agent need not participate in this process. The user is able to recover the keys using his smart card, which can recompute the target key K_S using the value *RAND* attached to the file.

The majority of KRMs lack such a feature; the user's KRA is typically the only entity that can recover the key. With the proposed scheme the user will be required to contact his agent only if he has lost his card, in which case only the authorised entity *AE* can recover the user's keys. This property typically eliminates the requirement for an on-line agent for the recovery of keys (for the purposes of user everyday access to encrypted data) and avoids the related communications overhead. Moreover, the user does not have to back-up or archive the generated keys for his own needs, thus avoiding certain problems associated with such an approach. Further properties of the proposed mechanism include:

1. The proposed mechanism is not vulnerable to rogue user attacks, as even if a rogue user deletes the generated *KRF* the KRA has the means

to recover the requested key. Using just the index file and the identity of the file and the user, the KRA has all the needed values to compute the required key.

2. The KRA does not have to store or protect any of the user generated keys, thus avoiding certain problems that key escrow mechanisms face, e.g. protection from unauthorised access to the escrowed material. The only requirement, apart from the protection of the secret value K_M, is protection of the index file from unauthorised modification.

3. The mechanism benefits from the separation of the KRA from the authorised entity AE in that the KRA does not have access to users' generated session keys. The only entities that can recover the session keys are the users and the authorised entity AE. This allows the corporation to outsource the management of the KRM without endangering the confidentiality of the corporate data.

4. Dispersion of key material, a countermeasure that makes attacks on key recovery mechanisms more difficult, is properly enforced with the use of both K_M and K_{AE} for the computation of K_{AM} and, therefore, the generation of K_S. Even if K_M or K_{AE} is compromised an adversary cannot gain access to the users' keys. The attacker has to know both K_M and K_{AE} to be able to recover users' keys.

5. The random value $RAND$ can be either generated on the card or on the user's PC and passed to the card. The security of the mechanism, however, does not rely on the randomness of this value, since it is only used to ensure freshness of the generated key. As a result, $RAND$ can be generated on the PC to reduce the number of power consuming procedures that take place on the card.

6 Conclusions

In this paper, the possible dangers to a corporation arising from an inability to access keys used for encrypting stored data have been considered. The requirements for a KRM used as a countermeasure have been identified and a new mechanism that fulfils them has been proposed. More specifically, the mechanism is not vulnerable to rogue user attacks, unlike many existing KRMs when used for encrypted archived data, while it offers the user the ability to recover his keys without his agent's intervention, a feature that eliminates any communication overheads. It has the additional benefit that it does not require the direct escrow of any user generated keys, avoiding the costs introduced by the demand for protection and administration of these keys.

Acknowledgements

The authors would like to thank Keith Martin and the anonymous referees for their helpful comments.

References

[1] D.E. Denning and D.K. Branstad. A taxonomy of key escrow encryption systems. *Communications of the ACM*, **39(3)**:34–40, March 1996.

[2] M. Smith, P. van Oorschot, and M. Willett. Cryptographic information recovery using key recovery. *Computers & Security*, **19(1)**:21–27, 2000.

[3] J. Kennedy, S.M. Matyas Jr., and N. Zunic. Key recovery functional model. *Computers & Security*, **19(1)**:31–36, 2000.

[4] IBM SecureWay. Towards a framework based solution to cryptographic key recovery. http://www-4.ibm.com/software/security/keyworks/library/.

[5] N. Jefferies, C. Mitchell, and M. Walker. A proposed architecture for trusted third parties. In E. Dawson and J. Golic, editors, *Cryptography: Policy and Algorithms — Proceedings: International Conference, Brisbane, Australia*, pages 98–104. Springer-Verlag (LNCS 1029), Berlin 1996.

[6] S.T. Walker, S.B. Lipner, C.M. Ellison, and D.M. Balenson. Commercial key recovery. *Communications of the ACM*, **39(3)**:41–47, March 1996.

[7] K. Rantos and C.J. Mitchell. Matching key recovery mechanisms to business requirements. Submitted.

[8] D.E. Denning. *Information Warfare and Security.* Addison Wesley, 1998.

[9] M.R. Smith. *Commonsense Computer Security.* McGraw-Hill, 1994.

[10] D.P. Maher. Crypto backup and key escrow. *Communications of the ACM*, **39(3)**:48–53, March 1996.