Key Recovery Scheme Interoperability – a protocol for mechanism negotiation

Konstantinos Rantos and Chris J. Mitchell

Information Security Group,
Royal Holloway, University of London,
Egham, Surrey TW20 0EX, UK.
{K.Rantos, C.Mitchell}@rhul.ac.uk

Abstract. This paper investigates interoperability problems arising from the use of dissimilar key recovery mechanisms in encrypted communications. The components that can cause interoperability problems are identified and a protocol is proposed where two communicating entities can negotiate the key recovery mechanism(s) to be used. The ultimate goal is to provide the entities a means to agree either on a mutually acceptable KRM or on different, yet interoperable, mechanisms of their choice.

Keywords: Key recovery, interoperability, negotiation protocol

1 Introduction

As business increases its use of encryption for data confidentiality, the threats arising from the lack of access to decryption keys grow [2]. Although transient keys typically should not be retained, there is a potential need to access these keys during their lifetime, i.e. during the communication session (or afterwards if the company logs any communications). Corporations might want to access encrypted communications to check for malicious software or to track leakage of sensitive information.

Key recovery mechanisms (KRMs) address this problem [3, 10] by providing the means to recover decryption keys. They can be divided into two types: *key escrow* and *key encapsulation* mechanisms. A *key escrow* mechanism, [8], is a method of *key recovery* (KR) where the secret or private keys, key parts or key-related information are stored by one or more key escrow agents. In a *key encapsulation* mechanism, keys, key parts, or key-related information are enclosed in a KR block, which is typically attached to the data and encrypted specifically for the *key recovery agent* (KRA). Here the terms *key escrow agent* and *key recovery agent* are considered synonymous, and refer to the trusted entity which responds to key recovery requests and potentially holds users' key-related material.

The variety of KRMs so far proposed, in conjunction with the lack of a standard for KRMs, means that interoperability problems are likely to arise from the use of dissimilar KRMs in encrypted communications [11]. Interoperability,

as in [5], means the ability of entity A, using KRM KRM_A, to establish a KR-enabled cryptographic association with entity B, using KRM KRM_B. Entities deploying dissimilar KRMs may not know whether the remote party can deal with their KRM's demands. This may force them to avoid key recovery, with associated increased risks. Note that, even if KRMs were standardised, interoperability problems may still arise; standards tend to provide a variety of sound but not necessarily interoperable mechanisms.

This paper addresses these interoperability problems. The components that can cause interoperability problems are identified and a protocol is proposed that, to a great extent, overcomes the interoperability problems. The protocol offers communicating parties the ability to agree either on a mutually acceptable KRM or on different, yet interoperable, mechanisms for their encrypted communications.

2 Key recovery enabled communications

In the context of communications between entities A and B, we give two scenarios where KRM use can affect the establishment of a cryptographic association.

1. Entities A and B make use of KRMs KRM_A and KRM_B respectively, which might be identical, compatible or dissimilar mechanisms. In the case of identical or compatible mechanisms the two entities are not expected to face any problems. Problems, however, might arise if the two entities make use of dissimilar mechanisms. They are unlikely to be able to establish a secure communication while using their respective KRMs, as this would typically demand each entity to fulfil the requirements of the peer's KRM.
2. Entity A uses KRM_A while B does not use KR. The issues that arise here are whether B will be able to cope with KRM_A's needs, and whether A will be able to generate valid KR information. For the two entities to be able to communicate, assuming that A manages to generate valid KR information, B should at least be aware that A makes use of a KRM. This is important as B should not discard incoming traffic because of unrecognised KR fields that B cannot interpret. Another potential problem is whether A's policy will permit the acceptance of incoming traffic that does not make use of KR. If A operates within a corporate environment this requirement is likely to be crucial, as the company might want to check incoming data for malicious software before they reach their destination.

Communicating entities wanting KR functionality for encrypted communications without the above problems must use interoperable KRMs. Also, any deployed cryptographic mechanisms with embedded KR functionality should be compatible with cryptographic products not using KR. These requirements will ensure that neither of the above scenarios will prevent the establishment of a secure session.

3 Factors that can affect interoperability

Many KRMs, especially key escrow mechanisms, demand the use of a specific mechanism for session key generation, and as such they can be considered as part of the key establishment protocol. This restriction is a cause of KRM interoperability problems. A KRM with this property demands compatibility of the underlying key establishment protocols, a requirement that is not always fulfilled. Key escrow mechanisms suffer more from this problem, as most require the use of a specific key establishment protocol. By contrast, key encapsulation schemes appear to be more adaptable in this respect, since they simply wrap the generated data encryption key under the KRA's public encryption key (and hence potentially work with any key establishment protocol).

Flexibility of key encapsulation mechanisms with respect to the underlying key establishment protocols does not always rule out interoperability problems. Interoperability very much depends on what additional requirements exist. For example, problems arise if the recipient of encrypted data needs to validate KR information, or the receiver relies on the sender to generate KR information. These needs will typically demand interaction between the two parties during KR information generation/verification. If either parties' mechanism cannot meet the peer's demands, interoperability problems are likely to arise. This problem is not restricted to key encapsulation schemes. In the case of key escrow mechanisms, a requirement for participation of both entities in generating KR information will have the same effect. We can therefore divide KRMs into two classes, depending on their communications requirements during KR information generation/verification.

1. KRMs where each entity generates KR information for its own use only, without peer assistance. If neither party requires verification of the peer's KR information prior to decryption, interoperability issues become of minor importance and the parties will be able to use their respective KRMs.
2. KRMs that require interaction between the two entities for the generation/verification of KR information. Interaction might be needed in the following cases.
   - Exchange of data is required for KR information generation.
   - The sender generates KR information both for his own and the peer's needs. This is particularly relevant to single-message communications.
   - Either party wishes, e.g. for policy reasons, to verify the KR information generated by the peer.

   In situations like these interoperability is an issue that must be dealt with; otherwise, it is likely to lead to a failure to establish secure communications.

In summary, the two factors that are likely to affect the interoperability of KRMs in encrypted communication sessions are:

1. the KRMs' dependence on the underlying key establishment protocol, and
2. the interaction requirements between the communicating parties for the generation and/or verification of KR information.

4 Interoperable mechanisms

Based on the above analysis, a mechanism which is neither dependent on the underlying key establishment protocol nor needs any interaction with the peer for generation or verification of KR information, will always be interoperable with a KRM with the same requirements. The two mechanisms can work independently regardless of the underlying key establishment protocol. A mechanism with these requirements can also inter-operate with one that is dependent on an underlying key establishment protocol which both entities can deal with, as long as it does not require interaction with the peer for KR information generation/verification.

Interoperability problems are likely to arise in the following cases (we assume that the communicating parties can deal with all possible underlying key establishment protocols):

1. Both KRMs use specific key establishment mechanisms regardless of interaction requirements. In this case the interoperability of the KRMs depends on the compatibility of the underlying cryptographic mechanisms.
2. At least one KRM demands peer participation in KR information generation/verification. For example, if the policy demands that the KR information for the receiver should be generated by the sender, then the sender must be able to handle the receiver's mechanism. Otherwise, it is likely that establishment of secure communications will fail.

The chances of interoperability problems are therefore considerable, and a solution is needed. However, due to the significant differences in the characteristics of existing KRMs it is unlikely that a single model, such as the one proposed by the Key Recovery Alliance in [5] (see also [11]), can apply to all of them. This latter model mainly addresses problems arising from the transmission of KR information in proprietary formats, and suggests as a solution the use of a wrapper, namely the "Common Key Recovery Block". However, as described in [9], it fails to achieve one of its main objectives, which is to offer the ability for validation of KR information by the peer, and does not deal with the situation where KR information cannot be generated because of the use of dissimilar mechanisms.

A different approach is described in this paper, which requires the entities to be able to deal with more than one KRM. This enables some of the difficulties described above to be avoided.

5 A KRM negotiation protocol

We now describe a protocol designed to enable two communicating parties to negotiate the KRM to be used in an encrypted communication session. Its main objective is to deal with situations where the parties wish to make use of different, non-interoperable, KRMs.

A similar protocol specifically designed to allow the negotiation of KRMs using the Internet Security Association and Key Management Protocol (ISAKMP)

[7] is described in [1]. This model, however, adopts the mechanism described in [5] for the transmission of KR information, which, as mentioned in the previous section, has been shown to have problems. A more generalised model is described here that considers the different requirements of various mechanisms and the additional requirements that might arise regarding the exchange of cryptographic certificates. Moreover, the proposed protocol can be used to provide key recovery functionality in the application layer, in contrast to the mechanism proposed by the Key Recovery Alliance which targets the IP layer.

Note that in the protocol description we refer to the two parties as 'Client' and 'Server'; this is so as to follow the client-server model terminology as closely as possible.

5.1 The proposed scheme

The protocol consists of the following steps (messages in Small Caps are optional; more detailed descriptions of the exchanged messages are given in the next section).

Client		Server
ClientHello	⟶	
		ServerHello
		Certificate
		CertificateRequest
	⟵	ServerHelloDone
Certificate		
KRParameters		
Finished	⟶	
	⟵	Finished

The client first sends the *ClientHello* message, to which the server responds with the *ServerHello*. With these two messages the two entities exchange the parameters necessary for KRM negotiation. After the *ServerHello* (if required by the selected KRM(s)) the server sends the *Certificate* message containing the appropriate certificates, requests client certificates with the *CertificateRequest*, and, finally, sends the *ServerHelloDone* message. The client responds with the optional *Certificate* message, containing the certificates specified in the *CertificateRequest*, the optional *KRParameters* message, containing any additional information required by the selected KRMs, and, finally, the *Finished* message. The server verifies the received *Finished* message and, if successful, responds with a similar *Finished* message. On receipt, the client verifies it and, if successful, the negotiation terminates successfully.

5.2 Exchanged messages

In the following sections the exchanged messages are described in detail.

Client Hello The client, as previously mentioned, initiates the protocol by sending the first message of the negotiation protocol (*ClientHello*). The *ClientHello* contains a list of KRMs (from a complete list of mechanisms the protocol supports) that the client is willing to use, in decreasing order of preference. A default mechanism that all parties are assumed to be able to use can be included in the list.

With the *ClientHello* the client must also inform the server whether he wants to resume a previous session by including the identifier in the appropriate field. If this field is empty a new session id should be assigned by the server.

Server Hello If the client does not request the resumption of a previous session, or if the server wants to initiate a new one, the server must assign a new session id, which will be sent with the *ServerHello* message. Otherwise, the server will respond with the session id included in the *ClientHello* and proceed with the *Finished* message.

If the server initiates a new session, he indicates the mechanism that he wants the client to use, and the mechanism that the server will use. The two mechanisms, if not identical, must be interoperable. For this purpose, a list of all possible matches of interoperable mechanisms has to be kept by both entities. If an acceptable match is not found in the list, the server can either terminate the negotiation protocol unsuccessfully, or choose the default mechanism if this is in the list sent by the client. Otherwise the server drops the session.

If the selection of the mechanism for both entities is a KRM that can itself handle the exchange of certificates and related KR parameters, the two parties can terminate the negotiation protocol and leave this KRM to take charge. To achieve this the server will send a *Finish* message (after the *ServerHello*) to indicate that control is now to be passed to the negotiated KRM(s).

Finally, within the *ServerHello* the server also includes the *KRParameters* field, which carries any additional information that the client has to possess to be able to deal with the server's KRM.

Certificate and KRM related information exchange Depending on the selection of the KRM, and if the server has not sent a *Finish* message, the server proceeds with the *Certificate* message. This message is optional and contains the required certificates (for the chosen mechanisms) for the generation and/or verification of KR information. Following that, and depending on the requirements of the chosen KRM(s), the server can also send a request for the corresponding client's certificates using the *CertificateRequest*. The purpose of the *CertificateRequest* is to give the client a list of specific types of certificates needed by the server, and a list of certification authorities trusted by the server. After the *CertificateRequest* the server sends the *ServerHelloDone* message, which indicates that the server has completed his *Hello* messages.

On receipt of the *ServerHelloDone*, if the client has received a *CertificateRequest* he responds with his *Certificate* message, which contains the requested certificates, assuming that he is in possession of the appropriate ones. Further, the

client sends in the optional *KRParameters* message any additional information required by the selected for the client KRM. Note that the corresponding *KRParameters* for the server's KRM is sent as part of the *ServerHello* message.

Finish messages If the client is satisfied with the current selection of mechanisms he sends the *Finish* message, which indicates that the client is willing to proceed with the current selection of mechanisms. Subsequently, the client waits for the corresponding server's *Finish* message, whose receipt indicates successful execution of the protocol.

5.3 Protecting the integrity of the *Hello* messages

After the execution of the above protocol the two entities are not sure whether any of the exchanged messages have been altered during transmission by an adversary, as the specified protocol includes no proper integrity checks. Moreover, neither of the communicating parties authenticates the other. Assuming, however, that the KRMs that can be negotiated are sound, the protocol does not introduce any vulnerabilities to the secrecy of the session key. The only attack that an adversary can mount against the protocol is to alter the *Hello* messages exchanged between the two entities in an attempt to downgrade the negotiated KRM(s) to one(s) that the attacker regards as weaker. Such an attack will only force the two entities to make use of less favourable mechanisms.

To avoid such problems we propose enhancing the previously proposed protocol. These enhancements provide the following security services.

- Integrity of the exchanged messages.
- Assurance that the *Hello* messages exchanged are not a replay from a previous session.

Additionally, the mechanism provides mutual authentication of the communicating parties. Note, however, that mutual authentication is not a requirement for the negotiation protocol. It is a property derived from the use of digital signatures. The modifications proposed are as follows.

- The client generates a random value $randC$, which he sends to the server with the *ClientHello* message.
- The server generates a random value $randS$, which he sends to the client with the *ServerHello* message.
- The client's *Finish* message becomes

$$S_C(ClientHello \parallel ServerHello \parallel randC \parallel randS)$$

  and the server's *Finish* message becomes

$$S_S(ServerHello \parallel ClientHello \parallel randS \parallel randC)$$

  where $S_U(M)$ is U's signature on data M and "$\parallel$" denotes concatenation.

The rest of the messages remain as previously defined. On receipt of the respective *Finish* messages the two entities check the signatures and if either of the two verification checks fails the protocol terminates unsuccessfully (this indicates that at least one of the *ClientHello*, *ServerHello* might have been altered during transmission). The modified protocol deals with the threat of modification of the exchanged *Hello* messages by an adversary. The generated random values prevent against replay attacks, i.e. where an adversary uses old exchanged messages to subvert the protocol. The cost of this countermeasure, however, is the introduction of signatures which have to be supported by an appropriate public key infrastructure. In practice the two variants could co-exist, and an extra field in the *Hello* messages could be used to indicate which variants of the protocol are supported.

6 Properties and Discussion

The proposed protocol offers the communicating parties a means of negotiating the KRMs to be used for their encrypted communications. Given that this negotiation will affect the selection of the key establishment protocol, execution of the proposed protocol must take place before the establishment of any session keys. It might also be the case that the two entities are obliged to use a specific key establishment protocol. This will simply restrict the number of mechanisms that the two entities will be able to negotiate.

The negotiating entities will be able to choose different KRMs as long as there are no conflicts between the underlying key establishment mechanisms. Therefore, the choice of the key recovery mechanism(s) will only be affected by the compatibility of the underlying key establishment protocols. If these are compatible, the two parties will be able to use the negotiated KRMs, overcoming efficiently any interoperability problems that the two parties would have otherwise faced.

Finally, note that in order to achieve the degree of agreement needed, the KRM negotiation process and the KRMs to be negotiated need to be subject of a standardisation process of some type (e.g. via the IETF). This standard will need to include agreed identifiers for a large set of KRMs.

7 Conclusions

The introduction of a large number of KRMs and their use in encrypted communications is likely to lead to interoperability problems between KR-enabled encryption products. In this paper the factors that can cause interoperability problems have been identified. Following a different approach to the single model solution proposed by the Key Recovery Alliance, a protocol has been proposed that gives communicating entities the means to negotiate the KRMs to be used. The parties can make use of different, yet interoperable, KRMs matching their needs for the specific communication session.

References

1. Balenson, D., Markham, T.: ISAKMP key recovery extensions. Computers & Security, **19(1)** (2000) 91–99.
2. Denning, D.E.: Information Warfare and Security. Addison Wesley, (1998).
3. Denning, D.E., Branstad, D.K.: A taxonomy of key escrow encryption systems. Communications of the ACM, **39(3)** (1996) 34–40.
4. Dierks, T., Allen, C.: The TLS protocol, Version 1.0. RFC 2246 (1999).
5. Gupta, S.: A common key recovery block format: Promoting interoperability between dissimilar key recovery mechanisms. Computers & Security, **19(1)** (2000) 41–47.
6. Kennedy, J., Matyas Jr., S.M., Zunic, N.: Key recovery functional model. Computers & Security, **19(1)** (2000) 31–36.
7. Maughan, D., Schertler, M., Turner, J.: Internet security association and key management protocol (ISAKMP). RFC 2408.
8. National Institute of Standards and Technology: Requirements for key recovery products. Available at http://csrc.nist.gov/keyrecovery/ (1998).
9. Rantos, K., Mitchell, C.: Remarks on KRA's key recovery block format. Electronics Letters, **35** (1999) 632–634.
10. Smith, M., van Oorschot, P., Willett, M.: Cryptographic information recovery using key recovery. Computers & Security, **19(1)** (2000) 21–27.
11. Williams, C., Zunic, N.: Global interoperability for key recovery. Computers & Security, **19(1)** (2000) 48–55.