The Royal Holloway TTP-based key escrow scheme

Chris J. Mitchell

Information Security Group
Royal Holloway, University of London

8th June 1996

ABSTRACT

In this paper we describe the Royal Holloway key escrow scheme, which provides a solution to the problem of managing cryptographic keys for end-to-end encryption in a way that meets legal requirements for warranted interception.

1. Introduction

There has been much recent discussion on the question of how to meet users' requirements for security services, such as confidentiality and authentication, whilst at the same time meeting legitimate requirements of government agencies for access to communications; a survey of recent work can be found in an article by Denning and Branstad, [1]. The discussion has been largely prompted by the US government's Clipper proposals [2], as well as the increasing use of electronic means for transferring commercially sensitive data. On the one hand, users want the ability to communicate securely with other users, wherever they may be, and on the other hand, governments have requirements to intercept traffic in order to combat crime and protect national security. Clearly, for any scheme to be acceptable on a wide basis, it must provide the service that users want, as well as meeting legal requirements in the territories it serves.

It seems likely that solutions to this 'key escrow' problem will be based on the use of trusted third parties (TTPs) from which users can obtain keys for encrypting their data or providing other security services. Law enforcement agencies' requirements will be met by obtaining relevant keys from a TTP within their jurisdiction, so they can decrypt the communications that they are authorised to intercept.

We describe here a mechanism (the 'Royal Holloway' (RHUL) scheme) that will enable TTPs to perform the dual rôle of providing users with key management services and providing law enforcement agencies with warranted access to a particular user's communications. Unlike other proposals, the mechanism allows users to update their keys according to their own internal security policies. This mechanism has previously been described in [3,4].

We go on to list typical application requirements for such a scheme, and consider how well the proposed mechanism meets these requirements. It is important to note that the scheme described here has been designed to establish keys for providing end-to-end confidentiality services, and not for integrity, origin authentication or non-repudiation services; the appropriateness of the mechanism for providing these services is a matter for further study.

2. The Mechanism

The RHUL mechanism is based on Diffie-Hellman key exchange [5]. To simplify our description we consider the mechanism in relation to one-way communication (such as e-mail). Adapting the scheme for two-way communication is very straightforward. We consider a pair of users *A* and *B*, where *A* wants to send *B* a confidential message and needs a session key to encrypt it. We suppose that *A* and *B* have associated TTPs *TA* and *TB* respectively, where *TA* and *TB* are distinct. Since this scheme is intended to provide warranted access to user communications via the TTPs, we assume that each TTP is within the jurisdiction of an intercepting authority, and each TTP operates subject to the regulations of that authority (such TTPs will probably operate within the terms of a licence).

2.1 Initial requirements

Before use of the mechanism, *TA* and *TB* need to agree certain parameters, and exchange some information.

- *TA* and *TB* must agree between them values g and p. These values may be different for each pair of communicating TTPs, and must have the usual properties required for Diffie-Hellman key exchange, namely that g must be a primitive element modulo p, where p is a large prime. These values must be passed to any clients of *TA* and *TB* who wish to communicate securely with a client of the other TTP.

- *TA* and *TB* must agree on the use of a digital signature algorithm. They must also each choose their own signature key/verification key pair, and exchange verification keys in a reliable way. Any user *B* wishing to receive a message from a user *A*, with TTP *TA*, must be equipped with a trusted copy of *TA*'s verification key (typically this would be provided by *TB*, perhaps using a signed certificate).

- *TA* and *TB* must agree a secret key $K(TA,TB)$ and a Diffie-Hellman key generating function *f*. This function *f* shall take as input the shared secret key and the name of any user, and generate for that user a private integer *b* satisfying $1 < b < p-1$ (which will be a 'private receive key' assigned to that user–see immediately below). The secret key $K(TA,TB)$ might be generated by a higher-level Diffie-Hellman exchange between the TTPs, or by any other method.

Given that *B* is to be given the means to receive secure messages from *A*, before use of the mechanism *A* and *B* must be provided with certain cryptographic parameters by their TTPs.

- Using the function *f*, the secret key $K(TA,TB)$ and the name of *B*, both *TA* and *TB* generate the private integer *b* satisfying $1 < b < p$-1 (as described above). This key is known as *B*'s *private receive key*. The corresponding *public receive key* for *B* is set equal to $g^b \bmod p$. The private receive key *b* for *B* needs to be securely transferred from *TB* to *B* (like other transfers discussed here, this can be performed 'off-line'). Note that *B* can derive its public receive key from *b* by computing $g^b \bmod p$. Note also that this key can be used by *B* to receive secure messages from any client of *TA*; however, a different key pair will need to be generated if secure messages need to be received from clients of another TTP.

- *A* must be equipped with a *send key pair*, for use when sending confidential messages to clients of *TB* (in fact this key pair could be used with many other TTPs, as long as they share the values *g* and *p*). *TA* randomly generates a *private send key* for *A*, denoted *a* (where $1 < a < p$-1). *A*'s *public send key* is then set to $g^a \bmod p$. *TA* then signs a copy of *A*'s public send key concatenated with the name of *A* using its private signature key, yielding a certificate for *A*'s public send key. The certificate is passed to *A* with a copy of *A*'s private send key *a* (this must be done using a secure channel between *A* and the TTP).

  In principle *A* could generate the private send key *a* him/herself, and then only pass its *public* send key to *TA* (by some reliable means which does not need to preserve secrecy). *TA* would then sign a copy of *A*'s public send key concatenated with *A*'s name to yield a certificate for *A*'s public send key, and then pass it back to *A*. The key escrow system would still work even though *A*'s TTP might not know *A*'s private send key. However, as discussed below, the system works in a more flexible way if *A*'s TTP does know *A*'s private send key, while giving *TA* *A*'s private send key does not give *TA* access to any more encrypted messages than if *TA* did not have this key.

- *A* must also be given a copy of *B*'s public receive key. *B*'s private receive key *b* can be computed by *TA* using *f*, the name of *B*, and the key $K(TA,TB)$. *TA* can then compute *B*'s public receive key g^b, which can then be transferred in a reliable way from *TA* to *A*.

2.2 The mechanism itself

As we have seen, before use of the mechanism, A has the following information:

- *A*'s own private send key *a*;
- a certificate for *A*'s own public send key ($g^a \bmod p$), signed by *TA*;
- the public receive key ($g^b \bmod p$) for user *B*, and;
- the parameters *g* and *p*.

This information can be used to generate a shared key $g^{ab} \bmod p$ for encrypting a message to be sent from *A* to *B*. This key can be used as a session key, or better, as a key-encryption key (KEK). The KEK can be used to encrypt a suitable session key. This latter approach has a number of advantages:

- it would facilitate the sending of email to multiple recipients, since the message can be encrypted once under a random session key, and this session key can then be distributed to each recipient by encrypting it using the KEK, and
- it allows the use of a new key for each message.

User *A* then sends the following information to user *B*:

- a message encrypted using the session key (either $g^{ab} \bmod p$ or a key encrypted using $g^{ab} \bmod p$),

- *A*'s public send key (g^a mod p) signed by *TA*, and
- the public receive key (g^b mod p) for user *B*

The received public receive key g^b mod p allows *B* to find its corresponding private receive key *b* (there will be a different receive key for each TTP with whose users *B* communicates). *B* can then generate the (secret) session key g^{ab} mod p by raising *A*'s public receive key (g^a mod p) to the power of *B*'s own private receive key *b*, and thus can decrypt the message.

A diagrammatic representation of the scheme is given in Figure 1.

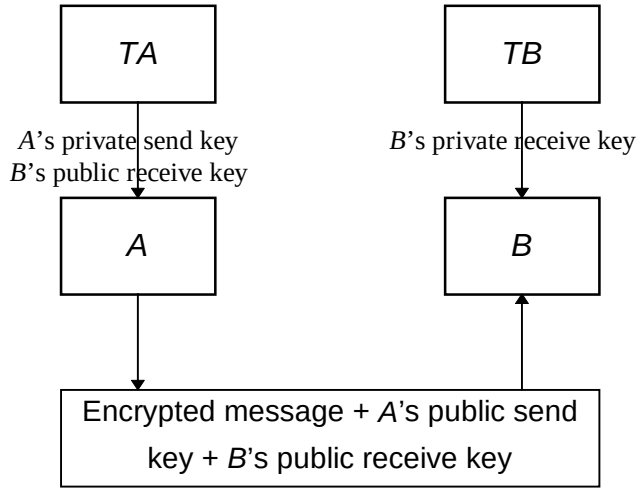

Figure 1: Use of the TTP scheme for one-way encrypted communication

2.3 Warranted interception

If a warrant exists for legal interception of a message, an intercepting authority can retrieve either the private send key of the 'sending user' or the private receive key of the 'receiving user' from the TTP in its jurisdiction, and use this in conjunction with the public receive key of the 'receiving user' or the public send key of the 'sending user', respectively, to recover the key for the encryption. An intercepting authority never has to deal with TTPs outside its jurisdiction, and no TTPs outside its jurisdiction need know what is going on.

More specifically, suppose user *A* (served by TTP *TA*) has sent a message to *B* (served by TTP *TB*). There are two cases to consider, depending on whether *TA* or *TB* is required to provide access to the encrypted message.

First suppose *TA* has to provide access to the message. There are two ways in which *TA* could recover the shared key g^{ab} mod p. It can combine either:

- *B*'s private receive key *b* (generated from *K*(*TA*,*TB*) and the name of *B* using the function *f*), with
- *A*'s public send key (g^a mod p), sent with the message,

or

- *A*'s private send key *a*, with
- *B*'s public receive key (g^b mod p), sent with the message.

Second suppose *TB* has to provide access to the message. Because *TB* will not have access to *A*'s private send key, there is only one way *TB* can get the shared key g^{ab} mod p, namely by combining

- *B*'s private receive key *b* (generated from *K*(*TA*,*TB*) and the name of *B* using the function *f*), and
- *A*'s public send key (g^a mod p), sent with the message.

When presented with appropriate authorisation (e.g. a warrant), there are two ways for the TTP to provide access to communications. The TTP could pass the appropriate keys to the intercepting authority, and then take no further part in the process, or the TTP could use its escrowed key(s) to decipher messages supplied by the intercepting authority, without revealing the keys themselves.

In order to assess the relative merits of these different approaches to providing warranted interception, we need to consider four possible situations (where the first two appear to be the most likely scenarios). We use notation from our discussion immediately above.

1. *TA* is warranted to provide access to all outgoing communications from a user *A* for which it acts.
2. *TB* is warranted to provide access to all incoming communications to a user *B* for which it acts,
3. *TA* is warranted to provide access to all incoming communications (from users for which it acts) to a user *B* for which it does **not** act,
4. *TB* is warranted to provide access to all outgoing communications (to users for which it acts) from a user *A* for which it does **not** act,

We first suppose that the TTP is required to provide keys to the intercepting authority. In case (1) it is sufficient for *TA* to provide the private send key(s) for *A*, and divulging these keys to the intercepting authority will not reveal information about any traffic not being sent by the user covered by the warrant. Note that, if *TA* did not possess the private send key for *A*, then it would be much more difficult for *TA* to provide warranted access to all *A*'s messages (it would be necessary for TTP to supply the session key for each individual recipient). In case (2), *TB* can supply *B*'s private receive keys for each of the other relevant TTPs; as before, divulging these keys to the intercepting authority will not reveal any information about any traffic not being sent to *B*. In case (3), *TA* can supply *B*'s private receive key (which it can work out), again without revealing any information not covered by the warrant. Case (4) is the most problematic, since *TB* will not have access to *A*'s private send key. In this case (which is less likely than cases (1) or (2)), all *TB* can do is provide the intercepting authority with the key g^{ab} mod p for every other user *B* which *A* sends messages to. Thus in all but one, relatively unlikely, case, the TTP can very easily provide exactly the key which will enable the intercepting authority to access the identified user's communications, without providing access to any communications which the intercepting authority is not entitled to.

The second approach to providing warranted access, i.e. having the TTP decipher messages 'on demand', avoids any of the problems we have just discussed. However, the main disadvantage of this approach is the increased communication required between the TTP and the intercepting authority, and the potential delay in accessing enciphered information.

Ultimately, the exact way in which TTPs provide warranted access to communications is a political matter, and may vary from domain to domain. The purpose of the above discussion is to show what options are available, and consider their relative merits.

2.4 Properties of the mechanism

We next observe a few significant properties of the proposed mechanism.

- First note that a user can change his/her *send key pair* at any time. A user simply asks his/her TTP to generate a new key pair, which is passed by the TTP to the user (with a signed certificate).
- No directories are needed to make the system work. An entity sending a message needs only get the public receive key for the recipient from his/her own TTP, who can generate this information from the name of the recipient and knowledge of the recipient's TTP. A recipient of an enciphered message will, with the information sent with the message, possess all data necessary to get the session key, without further reference to any third parties.
- Although receive key pairs are apparently fixed, by including the year (or month and year) within the scope of the key generating function *f*, receive key pairs can automatically be updated at regular intervals.

2.5 Possible methods of attack

We next briefly consider what approaches might be used to attack the scheme. First observe that the scheme is based on the Diffie-Hellman key exchange scheme, which has withstood detailed scrutiny over a period of time. The only means of attack on Diffie-Hellman of relevance here would appear to be the Burmester attack, [6]. As discussed in [4], this attack is not a threat in practice. In any case, given that a TTP always checks that a user possesses the private key corresponding to their public key before generating a certificate, which is already accepted practice for Certification Authorities, then the Burmester attack does not apply.

3. A brief analysis

Clearly, the definition and agreement of a set of requirements acceptable across a broad set of countries is largely a political process. However, we can give a set of typical or likely requirements on which to base an analysis of the suitability of the scheme.

Use of the scheme should provide visible benefits for the user. The design and operation of the scheme means that TTPs can offer their services to users on a commercial basis. By signing up to a licensed TTP, a user can communicate securely with every user of every TTP with whom his TTP has an agreement. The user can potentially choose from a number of TTPs in his home country, thus increasing trust in the TTP.

The scheme should allow national and international operation. The scheme achieves this by allowing users in any country, where an appropriate TTP resides, to communicate securely. It also ensures that the intercepting authority can obtain the required keys from a TTP within its jurisdiction.

Details of the scheme should be public. This has been achieved for the scheme by [3].

The scheme should be based on well known techniques, and Diffie-Hellman certainly qualifies.

All forms of electronic communication should be supported. The scheme can easily be adapted to cover two-way communication such as voice telephony.

The scheme should be compatible with laws and regulations on interception, as well as on the use, export and sale of cryptographic mechanisms. This matter is the subject of further study, but no problems have yet been identified.

Access must be provided to the subject's incoming and outgoing communication, where a warrant is held. This is clearly achieved for the scheme, as the subject's TTP will be able to provide the appropriate session keys.

The scheme should support a variety of encryption algorithms, in hardware and software. As the scheme deals solely with key management, any suitable encryption algorithm can be used, as long as it is available to users of the scheme (wherever they reside) and to the relevant interception authority. This could be achieved by using a standard list of algorithms, e.g. the ISO register.

An entity with a warrant should not be able to fabricate false evidence. This is particularly applicable in countries where intercepted communications are admissible as evidence in court. The scheme as it stands does not meet this requirement, but the provision of digital signatures as an additional service by the TTP will allow it to be met.

Where possible, users should be able to update keys according to their own internal policies. The scheme allows users to have new send key pairs as often as wished. Receive keys, which are a fixed function of the TTPs' shared key and the user's identity, are more permanent, and change only if the TTPs' shared key or the user's identity changes. However, as we have noted, if there is a requirement for receive keys to change at regular intervals, a date stamp could be included in the scope of the key generating function *f*. This would have the advantage that any private receive key provided to an intercepting authority would have a limited validity period, so that the warranted interception capability would only last for a certain time before needing to be renewed.

Abuse by either side should be detectable by the other. We believe that this is the case for this scheme, although abuse by collusion between the two sides may still be possible. The main disincentive to such abuse may be the 'shrink-wrapped' provision of the software, which could be bundled in with, say, an email system or other telecommunications software.

Users should not have to communicate with TTPs other than their own. The only communication required is with a user's own TTP.

On-line communication between TTPs should not be required. The independent generation of the receive keys means that no such communication is required.

4. References

[1] D.E. Denning and D.K. Branstad, *A taxonomy for key escrow encryption systems*. Communications of the ACM **39 No. 3** (March 1996) 34-40.

[2] National Institute of Standards and Technology, *FIPS Publication 185: Escrowed encryption standard*. February 1994.

[3] N. Jefferies, C. Mitchell and M. Walker, *A proposed architecture for Trusted Third Party services*. In: *Cryptography: Policy and Algorithms (International Conference, Brisbane, Australia, July 1995)*, Springer-Verlag, Berlin, 1996, pp.98-104.

[4] N. Jefferies, C. Mitchell and M. Walker, *Combining TTP-based key management with key escrow*. Submitted to the *Journal of Computer Security*.

[5] W. Diffie and M.E. Hellman, *New directions in cryptography*. IEEE Transactions in Information Theory **IT-22** (1976) 644-655.

[6] M. Burmester, *On the risk of opening distributed keys*. In: *Advances in Cryptology - CRYPTO '94*, Springer-Verlag, Berlin (1994) pp. 308-317.