Practical solutions to key escrow and regulatory aspects[1]

Nigel Jefferies, Chris Mitchell[2], Michael Walker

Information Security Group
Royal Holloway, University of London

8th September 1996

ABSTRACT

In this paper we propose a novel solution to the problem of managing cryptographic keys for end-to-end encryption, in a way that meets legal requirements for warranted interception. Also included are a discussion of what might constitute a reasonable set of requirements for international provision of such services, an analysis of the cryptographic properties of the scheme, consideration of how it might operate in practice, and a generalisation of the scheme to provide for 'split escrow' (i.e. allowing a user to distribute trust over several TTPs).

[1] Some of the results in this paper have previously been described in conference presentations made at the *Cryptography Policy and Algorithms Conference*, QUT, Australia, July 1995 and the *Public Key Infrastructure Invitational Workshop*, MITRE, Va, USA, September 1995.

[2] Chris Mitchell is currently at: Europay International, Chaussée de Tervuren 198A, B-1410 Waterloo, Belgium.

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 the legal requirements in the territories it serves.

To create a platform that can be used to provide user services, it is anticipated that solutions will be based on the use of trusted third parties (TTPs) from which users can obtain the necessary cryptographic keys with which to encrypt their data or make use of other security services. Law enforcement agencies' requirements will be focused on the need to obtain the relevant keys from a TTP within their jurisdiction, so that they can decrypt precisely those communications that they are authorised to intercept.

In this paper we propose a novel mechanism 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. Moreover, it provides a framework for Diffie-Hellman key establishment which obviates the need for directories.

After briefly considering possible attacks on the mechanism, we 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. We conclude by considering possible variants of the basic method, including a scheme allowing 'split escrow', and also how other proposed schemes for using TTPs in this way relate to the described method.

This paper was produced as part of the UK DTI/EPSRC-funded LINK PCP project 'Third-Generation Systems Security Studies' (3GS3). Participants in this project are Vodafone Ltd, GPT Ltd and Royal Holloway, University of London. The authors would like to acknowledge the valuable comments and suggestions of their colleagues in the 3GS3 project. We would also like to thank Burmester (and, independently, Anderson) for pointing out the possible relevance of Burmester's attack, [4], which we discuss in Section 2.5 below.

2. The Mechanism

The proposed mechanism is based upon the Diffie-Hellman algorithm for key exchange [3]. In order to simplify our description, we consider the mechanism only in relation to one-way communication (such as e-mail). The adaptation of the scheme for two-way communication is very straightforward.

More specifically we present the mechanism in the context of a pair of users *A* and *B*, where *A* wishes to send *B* a confidential message and needs to be provided with a session key to protect it. We suppose that *A* and *B* have associated TTPs *TA* and *TB* respectively, where *TA* and *TB* are distinct. Note that, since this scheme is intended to provide warranted access to user communications via the TTPs, we assume that each TTP is located within the jurisdiction of some intercepting authority, and that each TTP operates subject to the regulations of that authority (typically one might expect such TTPs to operate within the terms of some kind of licence).

2.1 Initial requirements

Prior to use of the mechanism, *TA* and *TB* need to agree a number of parameters, and exchange certain information.

- Every pair of TTPs whose users wish to communicate securely 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 operation of the Diffie-Hellman key exchange mechanism, namely that *g* must be a primitive element modulo *p*, where *p* is a large integer (satisfying certain properties). These values will need to be passed to any client users of *TA* and *TB* who wish to communicate securely with a client of the other TTP.

- Every pair of TTPs whose users wish to communicate securely 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 associated TTP *TA*, must be equipped with a trusted copy of *TA*'s verification key (typically this would be provided by their own TTP *TB*, perhaps by means of a signed certificate).

- Every pair of TTPs whose users wish to communicate securely 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 itself be generated by a higher-level Diffie-Hellman exchange between the TTPs, or by any other bilaterally agreed method.

Given that *B* is to be provided with the means to receive a secure message from *A*, prior to use of the mechanism *A* and *B* need to be provided with certain cryptographic parameters by their respective 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 the other transfers discussed here, this can be performed 'off-line'). Note that *B* will be able to derive its public receive key from *b* simply by computing $g^b \bmod p$. Note also that this key can be used by *B* to receive secure messages from any user associated with *TA*; however, a different key pair will need to be generated if secure messages need to be received from users associated with another TTP.

- *A* must be equipped with a *send key pair*, for use when sending confidential messages to users associated with TTP *TB* (in fact this key pair could be used with many, perhaps all, other TTPs, as long as they share the values *g* and *p*). *A*'s TTP randomly generates a *private send key* for *A*, denoted *a* (where $1 < a < p-1$). *A*'s *public send key* is then set equal 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; this yields a certificate for *A*'s public send key. The signed certificate is then passed to *A*, together with a copy of *A*'s private send key *a* (this must be done using a secure channel between *A* and the TTP).

  In fact, in principle at least, *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 the name of *A* to yield a certificate for *A*'s public send key, which would then be passed 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 we discuss in more detail below, the key escrow system works in a more flexible way if *A*'s TTP has access to *A*'s private send key, while giving the TTP the private send key of *A* does not give the TTP access to any more encrypted messages than if the TTP did not have access to this key.

- *A* must also be equipped with 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 as g^b, which can then be transferred in a reliable way from *TA* to *A*.

2.2 The mechanism itself

As we have seen, prior to use of the mechanism, *A* possesses the following information:

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

This information can be employed to generate a shared key g^{ab} mod p for protecting the confidentiality of a message to be sent from *A* to *B*. This key can be used as a session key, or, even better, as a key-encryption key (KEK). The KEK would then be used to encrypt a suitable session key. This latter approach has a number of advantages. For example:

- 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*:

- the message encrypted using the session key (either g^{ab} mod p or a key encrypted using g^{ab} mod 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*

Once received, the public receive key g^b mod p allows user B to find its corresponding private receive key b (there will be a different receive key for each TTP with whose users *B* communicates). User *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 received message.

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

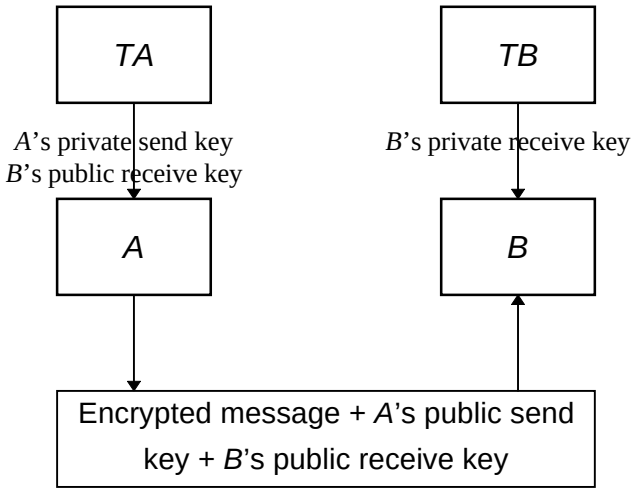

Figure 1: Use of the TTP scheme for one-way encrypted communication

2.3 Warranted interception

Should there be a warrant for legal interception of this communication, 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 trusted third party within 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 find the session key for the encryption. There is no requirement for the intercepting authority to deal with any TTPs outside of its jurisdiction, or for any TTPs outside of its jurisdiction to know what is going on.

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

First suppose *TA* is required to provide access to this message. There are two ways in which *TA* could recover the shared key g^{ab} mod *p*, namely it can combine either:

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

Second suppose that *TB* is required to provide access to this message. Then, because *TB* will not have access to *A*'s private send key, there is only one way in which *TB* could recover 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 key generating function *f*), and
- *A*'s public send key (g^a mod *p*), sent with the message.

When presented with the appropriate authorising information (e.g. a warrant), there are then two possible ways for the TTP to use this information to provide warranted access to communications. The TTP could pass the appropriate keys to the intercepting authority, and then take no further part in the interception process, or the TTP could use its escrowed key(s) to decipher messages presented to it 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 first need to consider four possible situations (where the first two correspond to what seem to be the most likely scenarios). We use notation corresponding to our discussion immediately above.

1. TTP *TA* is warranted to provide access to all outgoing communications from a user *A* for which it acts.
2. TTP *TB* is warranted to provide access to all incoming communications to a user *B* for which it acts,
3. TTP *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. TTP *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 rather less likely than cases (1) or (2)), the only thing that *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 gain access to 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 amount of communication required between the TTP and the intercepting authority, and the potential delay in accessing enciphered information.

In the final analysis, the exact way in which the TTPs provide warranted access to communications is a political matter, and may even vary from domain to domain. The purpose of the above discussion is to show what options are available, and consider their technical advantages and disadvantages.

2.4 Properties of the mechanism

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

- First note that a user can arrange for his/her *send key pair* to be changed at any time. A user simply requests his/her TTP to generate a new key pair for him/herself, which is then passed by the TTP to the user (along with a signed certificate).
- No directories are required to make the system work. An entity wishing to send a message only needs to obtain the public receive key for the intended recipient from his/her own TTP, who can generate this information merely from the name of the recipient and the identity of the recipient's TTP. A recipient of an enciphered message will, given the information contained in the message, possess all the data necessary to obtain the session key, without further reference to any third parties.
- Whilst, given the description above, receive key pairs are apparently fixed, by including the year (or month and year) within the scope of the key generating function *f*, all receive key pairs can automatically be updated at regular intervals. We discuss an option of this type in more detail in Section 4.1 below.

2.5 Possible methods of attack

We conclude this discussion of the mechanism by considering 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, [4], which we now discuss.

The basic idea of the Burmester attack (in the context of the scheme presented here) is as follows. Suppose user *A* has public send key g^a, and user *B* has public receive key g^b mod *p*. Then the key to be used to protect messages sent from *A* to *B* will be g^{ab} mod *p*. Now suppose that a third user, *C* say, manages to persuade *B* that its public send key is g^a mod *p* (we consider below ways in which this might occur). This means that the session key used for encrypting messages sent from *C* to *B* is g^{ab} mod *p*. *C* next claims to have temporarily lost its copy of the session key, and asks *B* to supply a replacement copy (by some secure means). *B*, believing that *C* is entitled to a copy of this key, complies, and now *C* has the means to decipher all the traffic sent from *A* to *B*.

As discussed in Burmester's paper, [4], there are many ways in which such an attack can be avoided, even without considering how *C* manages to persuade *B* that g^a mod *p* is really *C*'s public send key (we return to this latter point in a moment). First and foremost, *B* should never divulge the session key g^{ab} mod *p*, even if *B* believes that *C* is really entitled to it (since if *C* is really entitled to it, then why cannot *C* recompute it for him/herself?). Second, in a practical implementation users should be prevented from directly accessing session keys, thereby preventing them from divulging secrets accidentally or deliberately. This is probably a good idea in most practical security systems.

We now return to the key part of the above discussion, namely the means by which *C* manages to persuade *B* that *A*'s key is really *C*'s key (even though *C* does not know the private key *a*). There would appear to be three main ways in which this might occur. We consider each of them, and show that in each case either simple measures can prevent such attacks, or that the attack does not apply.

1. The first possibility is that *C* persuades *A*'s TTP (*TA*) that *C*'s public send key is g^a mod *p*, and gets *TA* to sign a certificate to this effect. To avoid this, a TTP must always get proof that a user possesses the private key corresponding to a public key before signing a certificate to the effect that this public key belongs to the user. This could be done be asking the user to utilise the private key to sign some data which could then be checked using the public key. This is in any case already accepted as good practice for the operation of a certification service.

2. The second possibility is that *A*'s TTP (*TA*) colludes with *C* and generates a certificate to the effect that *C*'s public send key is g^a mod *p*. Whilst this may be possible, *TA* already has the means to read *A*'s messages, and hence gains nothing by such an attack! The whole point of any key management system based on TTPs is that a user must trust the TTP they appoint to act on their behalf. We are thus entitled to ignore this case.

3. The third possibility is that *C* persuades some other TTP (*TC* say) that *C*'s public send key is g^a mod *p*, and gets *TC* to sign a certificate to this effect. However the above attack will no longer work in this case, since *B*'s private receive keys for users *C* and *A* will be different, since they are served by different TTPs. This should be clear by observing that every user's private receive key is computed as a function of a key shared by their respective TTPs. Hence, although *A* and *C* apparently share a *send key*, the session key for protecting messages sent from *A* to *B* will be different from the session key used for protecting messages sent from *C* to *B*, and hence *B*, however co-operative he/she might be, cannot reveal any useful information to *C*.

The above analysis shows that given that, before generating a certificate, a TTP always checks that a user possesses the private key corresponding to the public key which they claim as their own, then the Burmester attack does not apply.

3. A Typical Set of Requirements on a Trusted Third Party Scheme

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 proposed mechanism.

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

The scheme should allow national and international operation. The proposed 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 is achieved for the proposed scheme by the publication of this paper!

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

All forms of electronic communication should be supported. The proposed scheme can easily be adapted to include 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 proposed 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 proposed 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. The best way to achieve this may be to use a standard list of algorithms, such as 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 proposed 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 proposed scheme allows a user to have new send key pairs generated as often as wished. The receive keys, which are generated deterministically based on 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 already noted, if there is a requirement for receive keys to be changed at regular intervals, a date stamp could be included within 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 only a limited period of validity, meaning that the warranted interception capability could only last for a certain time period before needing to be renewed.

Abuse by either side should be detectable by the other. We believe that this is the case for the proposed 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 we would expect to 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 in the proposed scheme is with the user's own TTP.

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

4. Two variations on the basic mechanism

The proposed scheme can almost certainly be modified in many ways. We briefly present two such modifications, and consider their associated advantages and disadvantages.

4.1 Time bounding of TTP keys

As we have already discussed, in the scheme described above a user's receive key pair is apparently fixed, since it is generated as a deterministic function of a secret key shared by two TTPs and the name of the receiving user. We now describe one way in which this problem can be overcome though the use of date-stamps.

- The modified system requires the use of two key-generating functions *f* and *g* instead of one (although *f* and *g* might be the same function).
- Whenever user *A* requests his/her TTP *TA* to supply a copy of the public receive key of entity *B*, *TA* first computes *B*'s *permanent* private receive key *b* using *f* with inputs:

    1. the identity of *B*,
    2. the secret key shared by *TA* and *TB*,

    just as before. *TA* then uses *g* with the two inputs *b* and a current date-stamp to generate *today*'s private receive key for *B*, which we denote b'. *TA* then passes *B*'s public receive key for today ($g^{b'}$ mod *p*) to *A*.

- *A* uses *B*'s public receive key for today (in conjunction with *A*'s private send key) to compute a session key $g^{ab'}$ which is used to encrypt the message to *B*. *A* also sends a copy of today's public receive key with the encrypted message, along with the current date-stamp. Ideally *TA* should bind the date-stamp to *B*'s public receive key for today using a digital signature.
- As before, *B* will have been equipped with its own permanent private receive key *b* by its TTP *TB*. *B* can then use the function *g* to compute b' from *b* and the current date, and hence compute $g^{ab'}$, and thus decrypt the message.

The main advantages of this modified scheme are as follows.

- Every user's receive key pair will automatically change every day.
- Time-bounding of warrants could be enforced by only providing receive key pairs for the days specified in the warrant to an intercepting authority.

The main disadvantage is as follows.

- The TTP will need to pass several key pairs to an intercepting authority to provide access to communications for a period of time exceeding one day.

4.2 A scheme allowing split escrow

In an environment where commercial TTPs will be looking to offer additional services to their users, it is possible that some users will want the extra reassurance offered by having their keys shared between a number of independent TTPs. The proposed protocol is easily adaptable to provide this feature. For instance, the ideas of Micali [5] for adding secret sharing on top of existing schemes could be adopted. We now propose another solution which has the advantage of reducing the number of key pairs that message originators need hold.

4.2.1 Operation of the modified mechanism

For this scheme to operate, and unlike Micali's scheme, [5], every user *A* (with identity ID_A) will have a distinct modulus p_A and base g_A, used to secure all messages originated by *A*. These are computed as deterministic functions of ID_A, e.g. $p_A = F_p(ID_A)$ and $g_A = F_g(ID_A)$, where F_p and F_g are universally agreed functions.

Now suppose that each user subscribes to a set of TTPs, which the user is prepared to trust collectively but perhaps not individually. Thus, as previously, in the context of the situation where user *A* wishes to send a secret message to user *B*, we suppose that *A* subscribes to the set of TTPs $\{X_i\}$, and that *B* subscribes to the set of TTPs $\{Y_j\}$. Each of the TTPs X_i will need to know the identities of all members of the sets $\{X_i\}$ and $\{Y_j\}$, and each pair of TTPs (X_i,Y_j) share a secret key which we denote by K_{ij}. Observe that each of the TTPs in the set $\{X_i\}$ will need to be within the jurisdiction of the intercepting authority within which *A* resides, and similarly for $\{Y_i\}$.

Before *A* can send a secret message to *B*, two key pairs will need to be established, as we now describe.

- ***A* will need to be equipped with a send key pair**. Each of the TTPs X_i generates a part of *A*'s private send key; denote the part generated by X_i as S_{Ai}. Each X_i also computes their part of *A*'s public send key as

$$P_{Ai} = g_A^{S_{Ai}}.$$

  Each TTP X_i now signs a concatenation of the name of *A* with their part of *A*'s public key, and passes the resulting certificate, denoted $<P_{Ai}>$, together with S_{Ai} to A. A can now compute his/her private send key as

$$S_A = \sum_i S_{Ai},$$

and *A*'s public send key will be

$$P_A = g_A^{S_A} = \prod_i g_A^{S_{Ai}} .$$

- ***B* will need to be equipped with a receive key pair (and *A* will need to be given a copy of *B*'s public receive key)**. Each pair of TTPs (X_i,Y_j) now use a function *f*, taking as input their shared secret key K_{ij} and the identity of *B* (ID_B), to generate a part of *B*'s private receive key which we call S_{Bij}. I.e.

  $$S_{Bij} = f(K_{ij},ID_B).$$

  For each *j*, the values of S_{Bij} will be sent (securely) from TTP Y_j to *B*. *B*'s private receive key for use when receiving messages from clients of X_i will then be

  $$S_{Bi} = \sum_j S_{Bij} .$$

  Each of the TTPs X_i will perform the same calculation as *B*, and will also compute

  $$P_{Bi} = g_A^{S_{Bi}} .$$

  Each of these values will then be passed to *A* who can then calculate *B*'s public receive key as

  $$P_B = \prod_i P_{Bi} .$$

  It is important to note that the private receive key 'components' S_{Bi} are independent of the identity of *A*, but the public receive keys will vary depending on who the sender is.

The session key to be used to secure messages sent from A to B will then be

$$g_A^{S_A S_B} = (P_B)^{S_A} = (P_A)^{S_B}$$

and hence can be calculated by both *A* and *B*. The enciphered message will then be sent accompanied by the certified pieces of A's public send key <P_{Ai}> together with *B*'s public receive key P_B.

Key escrow will now operate in an exactly analogous way to that described in Section 2.3 above, with the exception that each TTP within a domain will now be required to supply their parts of either the send private key or the receive private key.

4.2.2 Properties of the modified mechanism

This modified version of the mechanism has the following properties.

- This modified scheme has a possible advantage over the scheme described in Section 2, even when each user only has one TTP. In this modified scheme users need only store one private receive key for each other TTP in the system, and only one private send key. The storage requirement for receive keys is thus unchanged, and the storage requirement for send keys is potentially reduced (depending on the number of different values of *g* and *p* used in the basic scheme).

- It is not possible for a user *A* to choose their own 'modulus' p_A and 'base' g_A, because *A* might include hidden structure within them which would allow access to other users' private receive keys. For example, p_A might be chosen so that some polynomial with small coefficients has a root modulo p_A, thereby facilitating the use of the Number Field Sieve. In addition, p_A must be prime, as its deterministic construction might otherwise compromise its factorisation.

- A user should probably not be permitted to change his/her modulus p_A, since that would potentially provide access to several different public keys computed from the same private key using different moduli. This might, at least in principle, lead to an attack on the private key using the Chinese Remainder Theorem.

- A possible alternative to deterministic generation of moduli is for a TTP to generate (and sign) a modulus p_A for *A*'s use. This would then need to be passed to each of the TTPs X_i, and also with each message sent by *A*. Such a scheme has the risk that it might be possible for a user to obtain several different moduli, and hence deterministic generation is probably preferable.
- *A* need not use the same set of TTPs all the time. For example, given that contacting a long list of TTPs may be expensive and/or inconvenient, the sender may choose to use the full set of TTPs only for particularly sensitive messages.
- It would be preferable to allow users to include a single public key with their message, signed by all of their TTPs, rather than the individual components specified above; apart from anything else this would reduce the communications overhead. However, it is not clear whether this can be done without increasing the possibility of 'cheating' the escrow system; it would also probably require the TTPs to communicate amongst themselves, which is not necessary with the current scheme.

5. Options and Other Issues

5.1 Trusting TTPs

The receiving party must trust the sending party's TTP, in order to verify the sending party's public key, and also because the sender's TTP can generate the receiver's private key. However, this trust only concerns communications between the receiver and senders belonging to that TTP.

There may also be a need for a certification hierarchy to identify a common point of trust for different TTPs; alternatively, all TTPs could manage their inter-relationships by bilateral agreement.

5.2 The Choice of Values

There has been considerable discussion in the literature on the benefits of using a composite modulus for Diffie-Hellman. This, and other matters such as the length of the modulus p and the primitive element g, are beyond the scope of this paper.

5.3 Commercial Value

The proposed scheme relies entirely on its perceived value to users in order to be taken up. Service providers will want to recover the cost of setting up the service from their customers. Therefore the scheme must be able to provide value-added end-to-end services that users want. Further investigation is required to assess the level of demand for services such as:

- end-to-end encryption;
- end-to-end authentication;
- non-repudiation of sent messages;
- message integrity.

Given that users will be paying for these services, they will expect a sufficient level of security. In the event of security failure with financial impact on the user, he will expect to be able to recover this, either via his insurers or from the organisation running the TTP. This makes running a TTP a potentially expensive business, unless the financial risks run by the TTP can be adequately protected against. If TTPs are not commercially viable, then the scheme will not be viable.

5.4 Combined two-way Session Key

The two-way version of the proposed scheme provides two keys for communication: one for each direction. These could be combined to form a single session key, or just one of the keys could be used. The advantages and disadvantages of this are a matter for further study.

6. Other Published Schemes

We conclude this paper by briefly indicating how two other key establishment schemes relate to the scheme described above.

6.1 The Goss Scheme

A scheme designed by Goss has been patented in the US [6]. In this scheme, a shared secret key is established by combining two Diffie-Hellman exponentiations using fixed and varying (per session) parameters. At first sight, this appears to correspond to the receive and send keys in the proposed scheme. However, the Goss scheme uses a universal modulus and primitive element. If x and x' are A's fixed and variant keys, and y and y' are B's, then the shared key is calculated as

$$\alpha^{xy'} \oplus \alpha^{x'y}$$

This could be viewed as a variant of the proposed two-way protocol whereby a universal modulus and primitive element are used and the two keys are combined by XOR-ing them.

6.2 Yacobi Scheme

The scheme of Yacobi [7] is almost identical to the Goss one, but uses a composite modulus, and combines the session keys by modular multiplication rather than XOR-ing.

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

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

[5] S. Micali, *Fair cryptosystems*. MIT Technical Report **MIT/LCS/TR-579.b**, November 1993.

[6] U.S. Patent 4956863, *Cryptographic method and apparatus for public key exchange with authentication*. Granted 11th September 1990.

[7] Y. Yacobi, *A key distribution paradox*. In *Advances in Cryptology - CRYPTO 90*, Springer-Verlag, Berlin (1991) pp. 268-273.