Enforcing Traceability in Software

Colin Boyd

Information Security Research Centre, School of Data Communications,
Queensland University of Technology, Brisbane 4001, Australia.
boyd@fit.qut.edu.au

Abstract. Traceability is a property of a communications protocol that ensures that the origin and/or destination of messages can be identified. The aims of this paper are twofold. Firstly the aims of traceable communications protocols are reviewed and compared with the available mechanisms to ensure compliance. These are compared with the methods used to ensure compliance in escrow schemes, the context in which traceability has usually arisen. Secondly a new communications architecture is proposed which provides traceability robustly, while preserving user control over other security services.

1 Introduction

There has been much recent interest in notions of *compliance* in cryptographic systems: how can users be provided with adequate security services in a controlled manner that does not allow them to abuse those services for unapproved purposes. The area in which compliance has been most studied recently is in escrow schemes [8]. Here the important issue is to allow users to employ cryptosystems for traditional confidentiality and integrity purposes, while ensuring that decryption is possible by authorised parties. More recently compliance has been an issue in electronic payment schemes, especially electronic cash [4,6]. In this context the compromise is between anonymity of user actions (a defining property of cash) and the need for financial and government bodies to regulate the flow and exchange of currency.

The Escrowed Encryption Standard [12], published by the US National Institute of Standards and Technology, defines the scheme underlying a family of devices including 'Clipper'. There are two, essentially independent, issues involved.

Escrow is the mechanism whereby a *copy* of an individual's secret key is stored in a specified manner.

Traceability is the mechanism that allows the owner of the scheme to *identify* the sender and/or recipient of a particular encrypted message.

In the Clipper device neither escrow nor traceability are implemented using cryptographic methods alone. Escrow is performed by a procedural (physical)

method at the time of manufacture of the device; traceability works by an algorithm that relies on the tamper-proof property of the device (although it includes a cryptographic algorithm for integrity).

It is natural to consider the possibility of implementing both escrow and traceability in software. There are a number of potential advantages of this. One is that there is reduced trust required in the operators of the system and the integrity of tamper-proofing mechanisms. Another is that if algorithms and protocols are made public they are more likely to gain the confidence of the users of the system. Key escrow can be implemented in software using *verifiable secret sharing* which is a subject of continuing research [16].

The problem of enforcing traceability in software has received less attention but recently Desmedt [9] considered whether it is possible also to devise cryptosystems with traceability properties. Although the scheme of Desmedt works perfectly if the users act correctly, when users abuse the system it is possible for them to avoid traceability while still exploiting the system. Knudsen and Pedersen [17] have shown that in at least three different scenarios users are able to produce messages which may be sent to one user but are indistinguishable in their tracing properties from those sent to a different user.

The first purpose of this paper is to consider what it means to provide a scheme that ensures compliance, especially in the context of software implementation. This is complemented by a consideration of the different security mechanisms that have been used to ensure compliance. The second purpose is to propose a specific new scheme to implement traceability. In order to avoid the sorts of attack described by Knudsen and Pedersen a different architecture is required and so use if made of a third party through whom all messages must be routed. However, the basic properties of Desmedt's scheme are retained; in particular users are not required to reveal their secret keys to any party but traceability of messages is enforced in a robust fashion. Thus there is no assumption of a *trusted* third party in contrast to some escrow schemes. It should be noted that this paper is concerned only with traceability; escrow is not directly addressed at all.

2 Background

2.1 Desmedt's Scheme

The idea is to use the ElGamal encryption algorithm [11] (or any variation of it) where the prime modulus p is chosen so that $p-1$ is divisible by the product of several primes. Each user i is allocated a different base value g_i which generates a unique subgroup of $\mathbf{Z}_p^*$. The value g_i forms part of that user's public key, which is then certified.

Each user then chooses a secret and public key pair in the same manner as ElGamal: the secret key of user U_i is s_i, chosen randomly in the range $1 \leq s_i \leq p-1$, while the corresponding public key is $y_i = g_i^{s_i}$ (all arithmetic is done in the field $\mathbf{Z}_p$). The public key of user U_i is the pair (g_i, y_i) while all users share the modulus p. To encrypt a message for user U_i the sender:

- chooses a random number r in the range $1 \leq r \leq p-1$ such that $(r, p-1) = 1$;
- calculates $C = My_i^r$ and $R = g_i^r$;
- sends the pair (R, C) to U_i.

The recipient can decrypt in the usual manner for ElGamal by first calculating $(g_i^r)^{s_i}$, inverting this value and multiplying the result by C.

Although the C value in a ciphertext can be any value, the R component is restricted to the subgroup of Z_p^* generated by the g_i value of the recipient. Traceability is based on the idea that each user has a different subgroup in which the R values of ciphertexts sent to that user lie. Although these subgroups have a small overlap, for large values of p and q_i there is a negligible probability that a ciphertext of one user will lie in the group of another.

In order to trace the recipient of a particular message the authority need only find which subgroup the R value lies in. This may be efficiently achieved by the authority by calculating the order of R. A limitation of the method is that the authority needs to keep the factors of $p-1$ secret, and in order to avoid an attacker factorising $p-1$ these must all be large. As a consequence Desmedt suggests that p needs to have around 10000 bits if a large number of users is to be accommodated.

2.2 Attacks of Knudsen and Pedersen

Desmedt provides a convincing argument (although he does not claim a proof) that a malicious user who cannot solve the Diffie-Hellman or factorisation problems will be unable to send a message that can be decrypted by the intended recipient but which cannot be traced. This argument relies on the assumption that the recipient will decrypt the received message in the exact manner intended. Knudsen and Pedersen [17] have pointed out three attacks which, although they violate this assumption, rely on relatively simple measures. The basis of these attacks is reviewed here, together with a fourth attack.

Conspiring Receivers In this attack two users i and j must conspire prior to choosing their private keys. They choose the same value $s_i = s_j$ for their private keys. They broadcast this information and as a consequence any sender can use the 'wrong' key $(g_i g_j, y_i y_j)$ to send a message to either i or j but which will not be traceable to either. In this attack i and j do not need to know whether or not the sender has used the correct key or the wrong key to encrypt.

Juggling Components The second and third of Knudsen and Pedersen's attacks involve changing the purpose of each of the components sent. One of these requires the sender to construct two encrypted messages where only the second component of each is used by the recipient, while the second flips the two components so that the wrong one is used in tracing.

Random Multiplier A fourth attack, which was already hinted at in the details of the two attacks of Knudsen and Pedersen which juggle components, is to use a random multiplier to hide the first component. The sender (or

the receiver) chooses a random value $m \in \mathbf{Z}_p$ and broadcasts it. The sender constructs (R, C) as normal but instead sends (mR, C). The recipient simply divides the first component by m before proceeding to the usual decryption algorithm. With high probability mR is not in the subgroup of the receiver.

Apart from the specific attacks mentioned, Knudsen and Pedersen also pointed out more general concerns which may be applicable to a wide range of similar protocols. In particular they point out that abusers of the system may simply use the public key of the receiver in a different algorithm altogether, thereby avoiding the possibility for tracing. They show, for example, that the public keys could be used in Diffie-Hellman key agreement.

3 Compliance Mechanisms

All schemes for key escrow and traceability need some methods to ensure that their users comply with the procedures for allowing encrypted communications to be traced and/or recovered. This is the case whether the scheme is intended for use as a backup method in a commercial environment or to enable monitoring by law enforcement or national security agencies[1]. Methods that have been suggested to ensure compliance may be divided into four classes as follows.

Tamperproof Hardware This method is the original one used by the Clipper proposal. Because the user is only able to use the scheme by employing the hardware implementation, use of the chosen mechanisms is unavoidable. The hardware is programmed to check on receipt of a message that the correct procedure has been followed. Although Blaze has shown [3] that the Clipper implementation can potentially be attacked, the basic principle of a using tamperproof hardware has not been undermined.

The major problem with relying on tamperproof hardware is that it seems to require also that the algorithms used must be kept secret. If this were not the case then users could engage software implementations modified in such a way as to avoid the tracing and escrow procedures. Or if not, then there is no necessity for the tamperproof hardware whose only purpose can be to hide certain features from the user. The arguments with respect to the undesirability, or otherwise, of unpublished cryptographic algorithms have been well rehearsed in recent years.

Tamperproof Software While the concept of tamperproof hardware is well established it is widely understood that compiled software can readily be reverse compiled to find the original algorithm. For this reason the idea of tamperproof software seems non-sensical. However there are situations where the effort to de-compile may not be worthwhile compared with the gains. A prime example would be commercial office software with key escrow mechanisms embedded. For almost all users the effort required to de-compile

[1] Traceability on its own has other applications such as resource monitoring which may be worth exploring. Such applications will also require compliance measures.

the software and re-compile it with alterations which sidestep the escrow mechanisms would be far beyond their capabilities or concerns. Methods to render software tamper resistant have been considered by Aucsmith [1].

Third Party Interactions A number of researchers have identified the possibility of using a third party service to interact with users and ensure that procedures have been properly followed. The first paper suggesting this seems to have been by Beth *et al.* [2] in which they propose a variation on Diffie-Hellman key exchange where the third party participates in choosing the session key. A number of potential weaknesses of the scheme were detailed by Horster, Michels and Petersen [14] and they proposed a revised scheme. A significantly different scheme, but still requiring the third-party to participate in the key exchange, is the Yaksha system of Ganesan [13]. A scheme of Jefferies, Mitchell and Walker [15] uses *trusted* third parties who choose the secret keys of the users.

Although the solution proposed in this paper uses third party interaction, there are a number of significant drawbacks of the above schemes that are avoided.

- The server must *interact* (that is both send and receive messages) with both parties before processing of any communication can begin.
- The server must hold security relevant information for each participant who may wish to communicate.
- Even in a store-and-forward application (such as electronic mail) the sender needs to find an on-line server before sending the message.

Certification Procedures Desmedt's scheme of 1995 does not fit into any of the above categories. Although its purpose is explicitly to allow software implementation it does not do this by assuming tamperproof software. Instead the aim is to devise a public key scheme such that a user's certified public key can only be used in a way that allows tracing of the identity of the recipient. Since a user must rely on public key certificates to make use of such a scheme it may be regarded that the certification procedure (including generation and allocation of public keys) is itself the method of ensuring traceability.

The difficulty of making such a scheme robust has already been alluded to above. The resourcefulness of attackers is impossible to anticipate and while a formal statement of security is lacking the precise achievement of Desmedt's scheme is impossible to assess.

The scheme to be described below compromises by making practical choices amongst the above schemes while avoiding their worst disadvantages. Thus software and hardware tamperproof assumptions are avoided. Desmedt's scheme is made robust by adding third party interaction, but in such a way that it is far less intrusive and much less difficult to implement than other proposals.

4 Security Services for Compliance

It is worthwhile to be as precise as possible about what security properties are achievable by any scheme. In general we would like to say that users of

the scheme should not be able to avoid traceability. A difficulty here is what defines a *user* of the scheme. For example, attackers who are assumed to have authenticated channels between them can broadcast keys on those channels for a totally different public key cryptosystem and thus avoid the scheme completely. Such interaction would not be reasonably thought of as breaking the scheme since it is not the aim to stop users colluding to devise their own security mechanisms.

In a scheme using tamperproofing to ensure compliance it is obvious to define the scheme to be *in use* if the device is used without alteration and is relied upon for security. In a scheme that relies on certification procedures for compliance, *use* of the scheme must imply utilisation of the public key that is certified. There are only two ways that a public key can be used in cryptography: it may be used to *encrypt* messages for confidentiality, or it may be used to *verify* authentication/integrity of messages. Essentially we would like users to make use of the public key in either of these two ways only if their communications are traceable.

Definition 1. A public key scheme is said to be *in use* by an entity if that entity is using the public key either:

- to provide confidentiality of messages sent, or
- to verify authentication of messages received

and that confidentiality or authentication fails if and only if the private key corresponding to the public key used is compromised.

It can be seen that in all the attacks of Knudsen and Pedersen described in section 2.2 the scheme of Desmedt is in use by this definition, while traceability is avoided. For example, the conspiring receivers attack uses a legitimate public key to provide confidentiality of user data, as do the juggling component attacks.

5 A New Software Traceability Scheme

In this section a new scheme for software traceability is proposed which guarantees traceability when the scheme is in use. The basic idea is related to Desmedt's scheme in using certification procedures to ensure compliance but also incorporates third party involvement. However many of the drawbacks of other schemes which use third parties are avoided. In particular the following features are achieved.

- Users are unable to make use of the scheme while avoiding traceability. This includes the use of the public key infrastructure with other algorithms.
- The third party, or any other party, is at no time given the secret key of any user.
- Although the third party needs to *process* every message, this processing is *not interactive.* For example when *sending* an email message no third party involvement is required.

– Users cannot avoid tracing by hiding their communications.

The basic idea relies on splitting the private key of the user with a third party. This is an application of multiple key ciphers [5] and to this extent is similar to the Yaksha system [13]. Third party processing is involved which depends on the identity of the recipient. To this extent it is more complex than Desmedt's scheme. However, this makes attacks much more difficult and all the ones so far proposed are ruled out. A further advantage is that a (relatively) small prime modulus is now possible making implementation much more efficient. The basic scheme is appropriate for use with key escrow schemes based on interactive key sharing schemes.

5.1 System Parameters

The scheme ensures compliance through the use of certification procedures. The certification authority and the third party need not be located together, but they share common secrets and so can be considered as the same from a security viewpoint. The parameters include a universal large prime p and a generator g of $\mathbf{Z}_p^*$. Encryption is with a variation of the ElGamal algorithm. As usual the length of p must be sufficient to make the discrete logarithm problem in $\mathbf{Z}_p$ infeasible. In order to prevent cheating by the authority the prime p should be chosen so that $q = (p-1)/2$ is also prime. The reason for this will be seen below.

The general idea is simply to force senders to include the name of the receiver with the message. As opposed to the methods using hardware or software tamperproofing to achieve this, here the constraint is that the third party T will be unable to help in the decryption without it. To achieve this T must use a different secret value b_A with each user A. In order to ease the burden on the third party it is convenient to make b_A a function of the identity A; this means that T does not need to store secrets for each user. However, because b_A must be a secret value, the function used should also depend on a secret value. An obvious choice is to have

$$b_A = h(K_T, A)$$

where h is a keyed hash function or Message Authentication Code (MAC) [18] and K_T is a secret value known only by T. The function h should have the following properties.

Collision Resistance This prevents an deliberate choice of two identities A and B for which the values b_A and b_B are equal.

Resistant to Forgery Without knowledge of K_T it should be infeasible to find the value of b_A for any identity A.

To simplify the security arguments it is desirable to have the property $(b_A, p-1) = 1$. This can be ensured by suitable choice of the length of p and the output length of h and by appending a single 1 to the output of h to ensure that it is odd. (Note that because $(p-1)/2 = q$, if $b_A < p-1$ then the only possible values for $(b_A, p-1)$ are 1, 2, and q.)

5.2 User Registration

User registration proceeds as follows.

1. User A chooses a secret value s_A, with $(s_A, p-1) = 1$.
2. The authority calculates b_A and gives A the value $\alpha = g^{b_A}$ (all calculations take place in $\mathbf{Z}_p$).
3. A checks that $\alpha^2 \neq 1$ and $\alpha^q \neq 1$, and if so gives the authority g^{s_A}.
4. Instead of certifying this value as A's public key (as for normal ElGamal encryption), the key $P_A = g^{s_A b_A}$ becomes Alice's public key that is certified by the authority for use in the system
5. The authority returns P_A and the certificate to A.
6. A checks that $P_A = \alpha^{s_A}$ and if so accepts the public key.

It is important that in this procedure the user A is able to be sure that the authority acts correctly. In particular it should not be possible for A to accept a certificate for a public key which can be decrypted by the authority alone. In other words, if A acts correctly and accepts the public key P_A it should not be possible for the authority to obtain the effective private key, even if the authority has not followed the protocol by choosing α in some different, unknown, way. This will not be possible as long as the well-known Diffie-Hellman protocol [10] is secure in the sense that the shared secret can only be found with knowledge of at least one of the secret inputs.

Lemma 2. *Suppose that A executes the protocol faithfully and accepts P_A. Then if the Diffie-Hellman protocol is secure in $\mathbf{Z}_p$ the authority cannot find the effective private key which is the discrete logarithm of P_A.*

Proof Since A accepts the public key, it follows from step 6, that $P_A = \alpha^{s_A}$. Let us write $P_A = g^{y s_A}$, where *a priori* it may or may not be the case that $y = b_A$. From step 5 it follows that the authority is able to calculate P_A from knowledge of g^{s_A} and $\alpha = g^y$. Now P_A is exactly the Diffie-Hellman key corresponding to the inputs s_A and y. Therefore, if the Diffie-Hellman protocol is secure, the authority must know the value of y. The effective private key is then the value $s_A y \bmod p-1$. By step 3, $(y, p-1) = 1$ and so y has an inverse modulo $p-1$. Therefore knowledge of $s_A y \bmod p-1$ enables the authority to find s_A. Thus if the authority can find the effective private key it can also take discrete logarithms in $\mathbf{Z}_p$ (and therefore break the Diffie-Hellman protocol). □

5.3 Message Processing

When a user B wishes to send a message to A he proceeds as follows.

1. B obtains A's certificate and encrypts his message m using the public key and ElGamal encryption to form the message (R, C) where $R = g^r$ for a random exponent r and $C = P_A^r m$.

2. B sends the encrypted message to the third party T (possibly via A) along with the identity of A.
3. T calculates b_A and the value (R^{b_A}, C) and sends this to A.
4. When A receives (R^{b_A}, C) she recovers the message by calculating $C/R^{b_A s_A}$.

5.4 Security

Lemma 3. *If the scheme is used for encryption in the sense of definition 1, then messages are traceable.*

Proof Suppose to the contrary that a certain message is not traceable. Then either it is not sent via T, or it is sent via T with a different identity from the intended receiver. In either of these cases the correct private key will not be used. This contradicts the definition of use of the scheme since if the private key is not used the security service cannot rely upon it. □

In the new scheme all the attacks of Knudsen and Pedersen are avoided. The conspiring receivers attack fails because the conspirers will be unable to find a user identity A which will provide the appropriate b_A value. Juggling components and the random multiplier attack are also hopeless because again the attacker cannot know what is the right identity to send with the altered message.

It is worth noting that an 'attack' is possible in which user A broadcasts the value α used in the registration phase. If other users then encrypt using α as A's public key, A can decrypt messages which have not been processed by T. However, this means that the public key P_A is *not in use* in the sense defined above. Furthermore, a user who encrypts a message using α as public key has no idea who may be able to read the message unless α was sent to that user along a pre-existing authenticated channel from A. In other words, this attack only allows users to bypass traceability in the case that a secure channel already existed to A when, as discussed earlier, it is not possible to ensure compliance.

In addition to the above properties it might also be noted that there is a significant extra power in providing traceability. If the scheme is in use then communications cannot be hidden by steganography or any other hiding of the communications channel. This is a property not even provided by the tamper-proof hardware mechanism.

5.5 Preventing Unauthorised Tracing

Sending the identity in plaintext has the potential disadvantage that any eavesdropper is able to trace the recipient as well as T. It could be argued that this is not the concern of the current scheme. However, it is likely to be deemed unacceptable and luckily a simple additional mechanism may be used to encrypt information for the third party use. The solution is to have an ElGamal public key $P_T = g^{b_T}$ which can be used to encrypt the identity of the sender when it is

sent to T with the message. The value of P_T must be publicly available for all users.

Instead of sending (R, C) together with the identity of A in plaintext, B must instead encrypt the identifier A using ElGamal encryption with P_T as public key. (In order to avoid leaking information it is worthwhile also to include some random padding in the plaintext to be encrypted.) On receipt of the message, T first extracts the identity of A and uses this to find b_A. As before, if the message is badly formed then the plaintext is not recoverable by A. The probabilistic nature of ElGamal encryption will prevent different messages to the same recipient being linked.

6 Discussion

6.1 Implementation Complexity

The most obvious disadvantage of the schemes presented here, in comparison with Desmedt's scheme, is that the third party must be available in order for decryption to take place. This is clearly a considerable overhead. It may be observed that several other schemes require such an overhead [2,13,14]. Furthermore, in distinction to all these schemes the sender does not need to interact with the third party, but only route the message via the third party. In a store-and-forward application in particular, this is much simpler since there is no real-time requirement for interaction. Note also that in an application using real-time communications the public key encryption would be used for exchange of a session key and subsequent communication will not have to be routed via T.

6.2 Scaling

A possible drawback of the scheme is the difficulty of scaling up to a widely distributed system. One way to implement the schemes would be to have trusted parties implemented widely, with each user being registered with a 'home' third party. By using a hierarchy of certificates, with each third party signature being certified globally, this is no impediment to global communication. Since most messages will be routed to the home location, there will be little extra communication costs, while computational requirements for traceability are distributed.

Another method to alleviate the problems of scaling is to have users obtain separate public keys and certificates from third parties that operate in each of the domains that the user frequents. For example, the certificate used could be based on the electronic mail domain of the user. The beauty of this arrangement is that the user can have different public keys and certificates while keeping her secret key always the same. This is because each third party generates a new b_A value using the domain secret T and can interact with the user to obtain the correct public key while the user's secret is the same.

Acknowledgements

I am very grateful to Ed Dawson of QUT, and to Wenbo Mao of Hewlett Packard, for many constructive critical comments.

References

1. D. Aucsmith, "Tamper Resistant Software - An Implementation", *Information Hiding*, pp.317-333, Springer-Verlag, 1996.
2. T. Beth, H.-J. Knobloch, M. Otten, G.J. Simmons, P. Wichmann, "Towards Acceptable Key Escrow Systems", *Proceedings of the 2nd ACM Conference on Communications and Computer Security*, 1994, pp.51-58.
3. M. Blaze, "Protocol Failure in the Escrowed Encryption Standard", *2nd ACM Conference on Computer and Communications Security*, 1994, pp.59-67.
4. E. Brickell, P. Gemmell, D. Kravitz, "Trustee-based Tracing Extensions to Anonymous Cash and the Making of Anonymous Change", *1995 SIAM-ACM Symposium on Discrete Algorithms*, pp.457-466.
5. C. Boyd, "Some Applications of Multiple Key Ciphers", *Advances in Cryptology - Eurocrypt 88*, Springer-Verlag 1988, pp.455-467.
6. J. Camenisch, J.-M. Piveteau, M. Stadler, "An Efficient Fair Payment System", *3rd ACM Conference on Computer and Communications Security*, ACM Press 1996, pp.88-94.
7. E. Dawson, J. Golić (Eds.), "Cryptography: Policy and Algorithms", Springer-Verlag, 1996.
8. D. Denning and D. Branstad, "A Taxonomy for Key Escrow Encryption Systems", *Communications of the ACM*, 39,3 March 1996, pp.34-40.
9. Y. Desmedt, "Securing Traceability of Ciphertexts – Towards a Secure Software Key Escrow System", *Advances in Cryptology - Eurocrypt '95*, Springer-Verlag 1995, pp.147-157.
10. W. Diffie and M. Hellman, "New Directions in Cryptography", *IEEE Transactions on Information Theory*, 22, 6, pp.644-654, 1976.
11. T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", *IEEE Transaction on Information Theory*, IT-31, 4, pp.469-472, 1985.
12. FIPS PUB 185, "Escrowed Encryption Standard", US Department of Commerce/National Institute of Standard and Technology, February 1994.
13. R. Ganesan, "The Yaksha System", *Communications of the ACM*, 39,3 March 1996, pp.55-60.
14. P. Horster, M. Michels, H. Petersen, "A New Key Escrow System with Active Investigator", Technical Report TR-95-4-F, University of Technology, Chemnitz-Zwickau, April 1995.
15. N. Jefferies, C. Mitchell, M. Walker, "A Proposed Architecture for Trusted Third Party Services", *Cryptography: Policy and Algorithms*, LNCS 1029, Springer-Verlag, 1996, pp.98-104.
16. J. Kilian, T. Leighton, "Fair Cryptosystems, Revisited" *Advances in Cryptology - Crypto 95*, Springer-Verlag, 1995, pp.208-221.
17. L. Knudsen and T. Pedersen, "On the Difficulty of Software Key Escrow", Advances in Cryptology - Eurocrypt 96, Springer-Verlag 1996, pp.237-244.
18. B. Preneel, P. C. van Oorschot, "MDx-MAC and Building Fast MACs from Hash Functions, *Advances in Cryptology - Crypto 95*, Springer-Verlag, 1995, pp.1-14.