On Efficient Key Agreement Protocols

Anish Mathuria *
Vipul Jain

Dhirubhai Ambani Institute of Information and
Communication Technology
Near Indroda Circle
Gandhinagar - 382 007
Gujarat, India

February 28, 2005

Abstract

A class of efficient key agreement protocols proposed by Boyd is examined. An attack is demonstrated on a round-optimal example protocol of this class, and a simple countermeasure is suggested. The whole class is known to be vulnerable to an attack proposed by Bauer, Berson and Feiertag. A new class of key agreement protocols without this vulnerability but having the same advantages in efficiency is identified, and a number of concrete protocols are proposed.

1 Introduction

Key establishment protocols are mechanisms that allow any two or more users to establish shared keys amongst themselves. There are two fundamental types of key establishment protocols [1]: *key transport* and *key agreement*. Key transport protocols are those in which a single entity is trusted to choose the key and securely transfer it to the other entities. Key agreement protocols are those in which each entity involved in the protocol provides an input to the equation that defines the key. Many of the key agreement protocols published in the literature employ public key cryptography and do not require an on-line server. Rueppel and van Oorschot [2] discuss several example protocols of this kind and compare their properties. We focus on key agreement protocols which employ shared key cryptography and require an on-line server. Protocols of this type have largely been ignored in the literature.

A fundamental property of any protocol for establishing session keys is the confidentiality of the key. Another important property is the freshness of the key. While the main task of the protocol designer is to ensure that the protocol achieves its security goals, it is always desirable to know whether a protocol that achieves its security goals is also efficient. An obvious efficiency measure is the number of message exchanges required between the parties involved in the protocol. In many protocols, several messages can be sent in parallel; these messages are said to belong to the same *round*. Gong [3] uses the minimum number of rounds needed by a protocol as another

*Contact author. E-mail: anish_mathuria@da-iict.org

efficiency measure. As noted by Hao, Clark and Jacob [4], communications efficiency issues such as the number of messages or rounds can be discussed at an abstract level, that is, without paying any attention to the algorithms used to provide cryptographic services. In contrast, computational efficiency issues are often discussed at the implementation level since they are dependent upon the choice of cryptographic algorithms.

In this paper we assume the usual scenario of two users who wish to establish a session key with the help of an on-line server using symmetric cryptography. Gong defined twelve classes of protocols of this sort and obtained lower bounds on the number of messages and the number of rounds for each class. He considered the use of two types of freshness identifiers, nonces and timestamps, to achieve key freshness. The bounds obtained by Gong suggest that protocols using nonces use more messages and rounds than those using timestamps. However, Boyd [5] showed that by a different design one can construct protocols using nonces for which the lower bounds on the number of messages and the number of rounds are the same as for protocols using timestamps. A protocol proposed by Kao and Chow [6] also lowers Gong's bounds for nonce-based protocols, but their protocol has the security disadvantage that one client can force reuse of an old key, a property easily avoided in Boyd's protocols. Recently, Crispo, Popescu and Tanenbaum [7] have used Boyd's design to devise a symmetric key based authentication infrastructure that enables key establishment within an unlimited population of users, without requiring the server to be on-line.

The protocols of Boyd and Kao and Chow have the property that if a user's long-term key is compromised, then an attacker can masquerade as that user even after the compromised key is replaced with a new one. By contrast, in protocols conforming to Gong's bounds such an attack is not possible once a new long-term key is installed. The first such attack on a published protocol was proposed by Bauer, Berson, and Feiertag [8] against the Needham-Schroeder protocol [9]. For lack of a better term, we will use the term *BBF attack* for any attack demonstrating that the protocol's security is not repaired even after a compromised long-term key is replaced. To avoid the BBF attack on Boyd's protocols, it is necessary to use a key revocation list containing keys that are known to be compromised, similar to the use of certificate revocation lists in public key infrastructures. In this paper we show that this constraint is incidental to the techniques used in those protocols, and we can develop protocols that have the same benefit in efficiency without the additional requirement of revocation lists to protect against the compromise of long-term keys. We also show an attack on an example protocol of Boyd that minimizes the number of rounds. The attack demonstrates that the protocol fails to achieve the key confirmation property which gives confidence about the possession of the key by intended users.

The rest of the paper is organized as follows. In Section 2 we review the basic ideas from Boyd's work [10, 5]. In Section 3 we review some concrete protocols proposed by him which lower Gong's bounds for protocols using nonces. In Section 4 we propose an attack on his round-optimal protocol, and strengthen the protocol against this attack. As already mentioned, the whole class of protocols proposed by Boyd is known to be vulnerable to the BBF attack. In Section 5 we propose alternative protocols that give protection against the BBF attack without needing key revocation lists. Section 6 concludes the paper.

2 Protocol Classes

We begin by recalling some of the concepts and definitions described by Boyd. The security properties that are essential to any key establishment protocol are described in the following definition.

Definition 1 ([10]) A protocol to establish a session key is *secure* if it is secure for all users involved. A protocol is secure for Alice if:

- Alice has acceptable assurance of who may have the key value.
- Alice has acceptable assurance that the key is fresh.

Each property in the above definition seems to naturally require that a user send a message to achieve it, or receive a message to achieve it. The sending or receiving of a message has to be integrated with cryptographic primitives in a secure manner. There are two possible ways of achieving assurance of recipient names.

Recipients by Imposition (RI) A trusted entity controls who gets the key. Alice is told the names of the intended recipients over an authentication channel from the trusted entity. The property of such a channel is that Alice can be sure that the names of the intended recipients were sent by the trusted entity.

Recipients by Choice (RC) Alice controls who gets the key. Alice generates keying material and sends it over a confidentiality channel to the intended recipients. The property of such a channel is that Alice can be sure that only the intended recipients will receive the keying material.

Key freshness can be achieved in one of the following two ways.

Freshness by Receipt (FR) A trusted entity controls freshness of the key. A fresh element is sent over an authentication channel from the trusted entity to prove freshness of the key.

Freshness by Input (FI) Alice controls freshness of the key. Alice generates a fresh input to the key generation process.

Overall, there are four different classes of protocols assuming Alice achieves assurance of recipient names and key freshness in the same way as Bob. Note that key agreement protocols lie in the protocol classes where freshness is achieved by input (FI). There are four such classes taking into account the possibility that Alice and Bob could achieve assurance of recipient names in distinct ways. The protocols of Boyd that lower Gong's bounds lie in the class where both Alice and Bob achieve assurance of recipient names by imposition. The new protocols proposed in Section 5 lie in the class where Alice achieves assurance of recipient names by choice and Bob achieves it by imposition (or vice-versa).

The following notations are used in the remainder of the paper. The notation $\{M\}_K$ denotes a string with the following properties.

1. The string can be used to recover M only with possession of K. (Confidentiality)
2. The string cannot be calculated from M without possession of K. (Integrity)

An encryption algorithm with a cryptographic checksum included would implement such a transformation. The next two notations correspond to transformations that satisfy only one of the two properties.

- $[M]_K$ denotes a string that satisfies integrity. A message authentication code (MAC) would be a typical implementation of such a transformation.
- $\langle M \rangle_K$ denotes a string that satisfies confidentiality. An encryption algorithm would be a typical implementation of such a transformation.

3 Existing Protocols

We assume two users A (Alice) and B (Bob) who share long-term keys K_{AS} and K_{BS} with a server S respectively. In what follows, N_A, N_B, and K_S are random values chosen by A, B, and S, respectively. The first example protocol proposed by Boyd is as follows [5, p. 4].

1. $A \rightarrow S$: A, B, N_A
2. $S \rightarrow B$: $\{A, B, K_S\}_{K_{AS}}$, $\{A, B, K_S\}_{K_{BS}}$, N_A
3. $B \rightarrow A$: $\{A, B, K_S\}_{K_{AS}}$, $[N_A]_{K_{AB}}$, N_B
4. $A \rightarrow B$: $[N_B]_{K_{AB}}$

Neither of the values N_A and N_B is kept secret. The value K_S is kept confidential to A and B. The session key K_{AB} is the result of the operation of a one-way function f of two variables,

$$K_{AB} = f(K_S, N_A \mid N_B),$$

where $N_A \mid N_B$ denotes the concatenation of N_A and N_B. The following two properties are essential to the security of the above protocol.

1. f should be such that it is infeasible to calculate $f(K_S, .)$ without knowledge of K_S.
2. f should be such that if one of the values N_A or N_B is fresh then it is infeasible to choose the other value so that the output of f is an old value.

The first property is needed to achieve assurance of recipient names. The second property is needed to achieve assurance of key freshness. A typical implementation of f would be a MAC algorithm.

The protocol above is more efficient in the number of messages than any protocol using nonces, where freshness is achieved by receipt, and where a handshake is included. The lower bounds Gong has found for nonce-based protocols of this class are 6 messages and 5 rounds if the two users jointly choose the key, or 5 messages and 4 rounds if the server chooses the key. The following example protocol, also due to Boyd, is similar in goal to the protocol above but requires only 3 rounds [5, p. 5].

1. $A \rightarrow S$: A, B
2. $A \rightarrow B$: A, N_A
3. $S \rightarrow B$: $\{A, B, K_S\}_{K_{BS}}$
4. $S \rightarrow A$: $\{A, B, K_S\}_{K_{AS}}$
5. $B \rightarrow A$: B, N_B
6. $B \rightarrow A$: $[N_A]_{K_{AB}}$
7. $A \rightarrow B$: $[N_B]_{K_{AB}}$

The session key is defined as before. This protocol can be executed in 3 rounds by sending the following sets of messages in parallel: messages 1 and 2, messages 3, 4, and 5, and messages 6 and 7.

The protocols described so far do not provide the forward secrecy property, that is, past session keys are compromised if long-term keys are compromised. Each of these protocols can be extended to provide forward secrecy. We take as example the following protocol [5, p. 6] in which the number of rounds remains the same as the second protocol above. In the description below, g is a generator of a suitable group in which the discrete logarithm problem is hard.

1. $A \rightarrow S$: A, B
2. $A \rightarrow B$: A, g^{N_A}
3. $S \rightarrow B$: $\{A, B, K_S\}_{K_{BS}}$
4. $S \rightarrow A$: $\{A, B, K_S\}_{K_{AS}}$
5. $B \rightarrow A$: B, g^{N_B}
6. $B \rightarrow A$: $[g^{N_A}]_{K_{AB}}$
7. $A \rightarrow B$: $[g^{N_B}]_{K_{AB}}$

The session key is the value $g^{N_A N_B K_S}$. It is obtained by A as $(g^{N_B})^{N_A K_S}$ and by B as $(g^{N_A})^{N_B K_S}$. Observe that knowledge of at least one of N_A or N_B and K_S is needed in order to obtain K_{AB} from the exchanged messages. The values N_A and N_B are destroyed by A and B respectively, after their time of use.

4 Attack

Key confirmation is a desirable goal in many key establishment protocols. It can be achieved by having each user send a message to the other that requires knowledge of the session key, usually called 'handshake' message. Since all the protocols above include handshake messages, it seems that key confirmation is a goal of these protocols. We demonstrate below an attack on the second example protocol to show that it does not provide key confirmation to A; the same attack is also applicable to the variant protocol which provides forward secrecy. In the following, C denotes an attacker who acts as B in the protocol.

1. $A \rightarrow S$: A, B
2. $A \rightarrow C$: A, N_A
3. $S \rightarrow C$: $\{A, B, K_S\}_{K_{BS}}$
4. $S \rightarrow A$: $\{A, B, K_S\}_{K_{AS}}$
5. $C \rightarrow A$: B, N_A
6. $A \rightarrow C$: $[N_A]_{K_{AB}}$

7. $C \rightarrow A$: $[N_A]_{K_{AB}}$

The first thing to note is that C obtains the value N_A in message 2. C is now able to send this value back to A in message 5 as B's nonce. C then waits for the handshake message sent by A, delaying the sending of the handshake message expected by A. The handshake message sent by A will be $[N_A]_{K_{AB}}$. After it receives this message, C can complete the protocol by simply replaying it to A. The attack shows that the protocol fails to assure A that B did actually obtain K_{AB}. Observe that C does not learn K_S and consequently cannot obtain K_{AB}. There is no failure of confidentiality or freshness of K_{AB} but there is loss of key confirmation.

Attacks which exploit the same pattern of weakness have been published in the literature; for example by Mitchell [11]. The usual remedy is to include identity fields in the handshake messages.

1. $A \rightarrow S$: A, B
2. $A \rightarrow B$: A, N_A
3. $S \rightarrow B$: $\{A, B, K_S\}_{K_{BS}}$
4. $S \rightarrow A$: $\{A, B, K_S\}_{K_{AS}}$
5. $B \rightarrow A$: B, N_B
6. $B \rightarrow A$: $[B, N_A]_{K_{AB}}$
7. $A \rightarrow B$: $[A, N_B]_{K_{AB}}$

The inclusion of identity fields ensures that the victim entity A cannot be used as an oracle by the attacker. Note this modification does not affect the number of rounds in the protocol.

The attack above does not apply to the first example protocol. The attack cannot succeed because in that protocol B sends his handshake message and nonce in the same message. The problem for the attacker now is that he cannot form the correct handshake message even though he can replay A's nonce.

5 New Protocols

It was noted by Boyd that his protocols are vulnerable to the BBF attack. To see this, suppose A's long-term key K_{AS} is compromised. As a consequence of this the attacker would gain access to K_S and thus be able to impersonate A to B. It would however be reasonable to expect that the protocol would become secure once the compromised key is replaced with a new one. This is not true for the protocols above. Even after a new key is installed, the attacker can continue to masquerade as A to B by simply replaying the value $\{K_S, A, B\}_{K_{BS}}$ containing a compromised K_S from an old run that took place before A's long-term key was replaced. Boyd suggested that the attack can be prevented by publishing a list of K_S values that are known to be compromised, similar to certificate revocation mechanisms used in public key infrastructure. However, this may be seen as an important disadvantage of his protocols since protocols where key freshness is achieved by receipt do not require such mechanisms. In this section we show that it is possible to avoid the use of key revocation lists without loss of security, while maintaining the level of efficiency achieved by Boyd's protocols.

The first protocol we propose is as follows.

1. $A \rightarrow S$: $\{B, g^{N_A}\}_{K_{AS}}$
2. $S \rightarrow B$: $\{A, g^{N_A}\}_{K_{BS}}$
3. $B \rightarrow A$: g^{N_B}, $[g^{N_A}]_{K_{AB}}$
4. $A \rightarrow B$: $[g^{N_B}]_{K_{AB}}$

As in Diffie-Hellman key exchange, the session key is $K_{AB} = g^{N_A N_B}$. Note that A sends her contribution g^{N_A} in a confidential way to B via S, so A achieves assurance of recipient names by choice. B obtains recipient information by imposition. Both A and B obtain freshness of K_{AB} by input. Thus from A's viewpoint the protocol lies in class RC/FI, and from B's viewpoint it lies in class RI/FI.

Similar to Boyd's first protocol, the protocol above has four messages. By rearranging the messages, we obtain the variant protocol below that can be executed in three rounds.

1. $A \rightarrow S$: $\{B, g^{N_A}\}_{K_{AS}}$
2. $A \rightarrow B$: A, B
3. $S \rightarrow B$: $\{A, g^{N_A}\}_{K_{BS}}$
4. $B \rightarrow A$: g^{N_B}
5. $A \rightarrow B$: $[g^{N_B}]_{K_{AB}}$
6. $B \rightarrow A$: $[g^{N_A}]_{K_{AB}}$

As before, the session key is $K_{AB} = g^{N_A N_B}$. A useful property of this protocol is that it requires one less message than the round-optimal protocol of Boyd's class. The attack described in Section 4 does not work against the protocol above because the attacker does not gain access to the value g^{N_A}. It therefore does not seem necessary to include the identity fields in the handshake messages.

Before examining the security of the proposed protocols against the BBF attack, we construct a 'split channel' protocol similar to the one constructed by Boyd. The advantage of this protocol is that it makes explicit distinction between those message elements that need confidentiality protection and those that need integrity protection, leading to greater flexibility in the choice of cryptographic algorithms.

1. $A \rightarrow S$: $\langle g^{N_A} \rangle_{K_{AS}}$, B, $[B, g^{N_A}]_{K_{AS}}$
2. $S \rightarrow B$: $\langle g^{N_A} \rangle_{K_{BS}}$, A, $[A, g^{N_A}]_{K_{BS}}$
3. $B \rightarrow A$: g^{N_B}, $[g^{N_A}]_{K_{AB}}$
4. $A \rightarrow B$: $[g^{N_B}]_{K_{AB}}$

5.1 Compromise of long-term keys

Let us consider the security of our proposed protocols against the BBF attack. There are two cases to consider.

Suppose that Alice's old key K_{AS} is compromised and subsequently replaced. Consider what an attacker may gain from this compromise. Compromise of K_{AS} reveals g^{N_A}, but does not reveal N_A. The attacker could replay the value $\{A, g^{N_A}\}_{K_{BS}}$ to B, but he cannot find the value of the session key since it requires knowledge of N_A. The attacker could also generate a new value of the form $\{B, g^{N_X}\}_{K_{AS}}$ and send it to S, but S will not accept this message since the value K_{AS} has been replaced with a new one.

Suppose that Bob's old key K_{BS} is compromised and subsequently replaced. The attacker cannot masquerade as B to A because it requires knowledge of g^{N_A}, which can be retrieved only with knowledge of B's new key or A's key.

5.2 Re-issuing of keys

Boyd's protocols have the feature that the values $\{A, B, K_S\}_{K_{AS}}$ and $\{A, B, K_S\}_{K_{BS}}$ generated by S are independent of the random inputs chosen by A and B. As a consequence of this it is possible to cache the value K_S and re-use it to establish new session keys without contacting S again. This feature may be obtained in our protocols by modifying the messages sent by S and B. Consider the following example protocol.

1. $A \rightarrow S$: $\{B, g^{N_A}\}_{K_{AS}}$
2. $S \rightarrow B$: $\{A, g^{N_A}\}_{K_{BS}}$, $\{A, B, K_S\}_{K_{AS}}$, $\{A, B, K_S\}_{K_{BS}}$
3. $B \rightarrow A$: g^{N_B}, $[g^{N_A}]_{K_{AB}}$, $\{A, B, K_S\}_{K_{AS}}$, $\{A, B, K_S\}_{K_{BS}}$
4. $A \rightarrow B$: $[g^{N_B}]_{K_{AB}}$

The session key is now defined as $K_{AB} = g^{N_A N_B K_S}$. Notice that Alice achieves assurance of recipient names in more than one way. Considering that her own key component is sent over a confidentiality channel, Alice achieves assurance of recipient names by choice. Considering that she receives information from the server on who else knows K_S, she achieves assurance of recipient names by imposition.

With the protocol above, the following protocol can be used to re-establish new keys without using S.

1. $A \rightarrow B$: $g^{N'_A}$, $\{A, B, K_S\}_{K_{BS}}$
2. $B \rightarrow A$: $g^{N'_B}$, $[B, g^{N'_A}]_{K'_{AB}}$
3. $A \rightarrow B$: $[A, g^{N'_B}]_{K'_{AB}}$

The new session key is defined as $K'_{AB} = g^{N'_A N'_B K_S}$, where $g^{N'_A}$ and $g^{N'_B}$ are new inputs chosen by A and B. It should be noted that one feature of the original protocol is lost, namely resistance to BBF attack without using key revocation lists. The view may therefore be taken that this particular class of protocols is not particularly attractive over Boyd's protocols.

6 Conclusions

In this paper we proposed new key agreement protocols that lower Gong's bounds for protocols using nonces. Our protocols do not require any additional mechanisms such as key revocation lists for protection against the BBF attack, unlike existing nonce-based protocols that lower Gong's bounds. The lesson to be learned from the attack on Boyd's protocol is that the security of any protocol must always be considered relative to all its goals. There are intuitive arguments for the security of the proposed protocols, but no proof has been given. There is wide recognition that proving protocols secure is a difficult task. Bellare and Rogaway [12] have pioneered a reductionist security proof methodology for protocols which is widely used today. Although a number of protocols exist for which security proofs are available, these protocols turn out to be less efficient in the number of messages and rounds than our protocols. The Bellare-Rogaway definition of security is violated if the session key is used within the protocol. Thus it rules out proofs of protocols that provide key confirmation. Furthermore, in their model there is no notion of installing a new long-term key after discovering a compromise; thus a security proof does not imply that the protocol is secure against the BBF attack. For example, Shin and Lee [13] have shown that two protocols proposed by Shoup and Rubin [14, 15], which have been proven secure and implemented, are vulnerable to the BBF attack.

References

[1] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[2] Rainer Rueppel and Paul C. van Oorschot. Modern key agreement techniques. *Computer Communications*, 17:458–465, 1994.

[3] Li Gong. Lower bounds on messages and rounds for network authentication protocols. In *1st ACM Conference on Computer & Communications Security*, pages 26–37. ACM Press, 1993.

[4] Chen Hao, John Clark, and Jeremy Jacob. Synthesising efficient and effective security protocols. In *IJCAR 2004 Workshop on Automated Reasoning for Security Protocol Analysis*. `http://wwww-users.cs.york.ac.yk/~jeremy/papers/ARSPA2004.pdf`.

[5] Colin Boyd. A class of flexible and efficient key management protocols. In *9th IEEE Computer Security Foundations Workshop*, pages 2–8. IEEE Press, 1996.

[6] I.-L. Kao and R. Chow. An efficient and secure authentication protocol using certified keys. *ACM Operating Systems Review*, 29(3):14–21, 1995.

[7] Bruno Crispo, Bogdan Popescu, and Andrew Tanenbaum. Symmetric key authentication services revisited. In *Proceedings of ACISP 2004*, volume 3108 of *LNCS*, pages 248–261. Springer-Verlag, 2004.

[8] R. Bauer, T. Berson, and R. Feiertag. A key distribution protocol using event markers. *ACM Transactions on Computer Systems*, 1(3):249–255, August 1983.

[9] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, December 1978.

[10] Colin Boyd. A framework for design of key establishment protocols. In *Proceedings of ACISP'96*, volume 1172 of *LNCS*, pages 146–157. Springer-Verlag, 1997.

[11] Chris J. Mitchell. Limitations of challenge-response entity authentication. *Electronics Letters*, 25:1195–1196, 1989.

[12] Mihir Bellare and Phillip Rogaway. Provably secure session key distribution – the three party case. In *27th ACM Symposium on Theory of Computing*, pages 57–66. ACM Press, 1995.

[13] Jun-Bum Shin and Kwang H. Lee. Bauer-Berson-Feiertag attack revisited. Technical Report 2002/146, Cryptology ePrint Archive, 2002. `http://eprint.iacr.org/2002/146/`.

[14] Victor Shoup and Aviel Rubin. Session key distribution using smart cards. In *Advances in Cryptology - Eurocrypt'96*, volume 1070 of *LNCS*, pages 321–331. Springer-Verlag, 1996.

[15] R. Jerdonek, P. Honeyman, K. Coffman, J. Rees, and K. Wheeler. Implementation of a provably secure, smartcard-based key distribution protocol. In *Proceedings of CARDIS '98*, volume 1820 of *LNCS*, pages 229–235. Springer-Verlag, 2000.