Comparing lower bounds on messages and rounds for two classes of key establishment protocols

Anish Mathuria

IBM Tokyo Research Laboratory, LAB-S77
1623-14, Shimotsuruma, Yamato-shi
Kanagawa-ken 242-0001, JAPAN
E-mail: anish@trl.ibm.co.jp

Abstract

An important requirement in designing protocols for key establishment is to provide assurance to protocol participants that a session key is fresh. This paper compares lower bounds on messages and rounds for two classes of protocols based on fundamentally different methods for achieving session key freshness.

1 Introduction

Security protocols using symmetric key cryptography for session key establishment between two or more users typically involve a server that shares a long-term key with each user. The seminal paper of Needham and Schroeder [6] introduced protocols of this kind in the literature. The fundamental aim of such protocols is to establish a fresh shared secret, known only to the users and possibly the server. Two common methods for ensuring session key freshness are the use of a challenge-response exchange and the use of timestamps. Both these methods require that each user receive an appropriate message that binds the session key together with some item which the user can verify to be fresh. However, as first explictly recognised by Boyd [2], it is possible to derive the session key as the result of a suitably chosen function so that its freshness is simply achieved by having each user provide a fresh input to the function, removing the need for the users to receive any messages in order to obtain reassurance of the freshness of the session key. The requirement of session key exclusivity can also be met by incorporating into the function a secret key input that controls the confidentiality, but not the freshness, of the derived session key. The above general design idea of Boyd results in a novel class of protocols having a number of attractive features. One of these features is the efficiency gain that can be achieved when compared to the class of protocols where session key freshness is achieved by using a challenge-response exchange. The efficiency achieved by Boyd's protocol class is greater with respect to the following two protocol metrics defined by Gong [3]: (1) number of messages and (2) number of rounds. Boyd presents two concrete protocols of the above class that overcome Gong's lower bounds on messages and rounds for protocols relying on challenge-response in the case where the server chooses the session key, and where a handshake using the session key is included. The goal of this paper is to investigate why greater efficiency is achieved in the protocol class introduced by Boyd, and to examine some of the trade-offs involved.

In the description of protocols below, we use the following notations, amongst others.

- $P \rightarrow Q : X$ represents a message exchange where principal P sends to another principal Q a data item X.

- $\{X\}_K$ represents X encrypted under K; this notation assumes that encryption provides both confidentiality and integrity.
- $[X]_K$ represents an integrity check value on X under K.

Following Gong [3], we informally define two metrics, the number of messages and the number of rounds, as follows. The *number of messages* is the total number of message exchanges required to complete the protocol. A *round* is a group of message exchanges that may be executed simultaneously; the *number of rounds* is the minimum number of rounds required to complete the protocol.

2 Boyd's message-efficient protocol

The first protocol given in Boyd's paper [2] allows two users A and B to establish a shared session key K_{AB}, with the help of a server S with whom they initially share keys K_{AS} and K_{BS}, respectively:

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. $A \rightarrow B$: $[N_B]_{K_{AB}}$

Figure 1: Boyd's first protocol

Here, N_A and N_B are random nonces generated by A and B, respectively; K_S is a key generated randomly by S. The session key K_{AB} is defined by:

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

where $N_A \mid N_B$ is the concatenation of N_A and N_B, and f is a suitably chosen function. A successful run of the protocol works as follows. In message 1, A generates N_A and sends it to S. Upon receiving this message, S generates K_S and creates two encrypted components, one for each of A and B. The component for A (B) contain K_S and the names A and B, encrypted under K_{AS} (K_{BS}). The server S sends both these components to B in message 2, along with A's nonce. B decrypts the component meant for it, generates N_B, and derives the session key. In message 3, B forwards the component meant for A, along with a value to convince A that it knows the session key. In the same message, B sends N_B to A. Finally, A derives the session key and returns a message to convince B that it knows the session key.

Notice that f must have the property that when the values of the two inputs are given, the result of f is easy to compute. It must also have some additional properties so as to maintain the confidentiality and freshness of the derived session key. The corresponding function properties can be summarised as follows:

1. The result of f cannot be feasibly computed without the knowledge of the value of the first input, whatever the value of the second input.
2. For any fixed value of the first input it is computationally infeasible to find two different values of the second input such that f results in the same output value.

The specific properties required of f are essentially similar to the standard desired properties of keyed hash functions or MACs [5].

The security of the protocol shown in Figure 1 is argued by Boyd. In particular, the following goal is achieved by this protocol:

> *At the end of a successful run A and B can both be sure that the session key is new, and is shared with the other.*

The protocol has one property that is worth noting for our purposes: S remains uninformed of the value of the session key shared by A and B during a normal run of the protocol. Since S does not receive any protocol message from which it can extract N_B, it does not normally obtain the session key value upon protocol execution. The design of the protocol tacitly indicates that it is not important that the session key value become known to S. It is more important that the session key become known to A and B, since it is assumed to be used for subsequent communication between them, without the help of S.

Consider the protocol class defined by Gong [3] that satisfies the following criteria:

- The server chooses the session key shared by A and B.
- The freshness of the session key is proved by using a challenge-response exchange between the server and each of A and B.
- A and B conduct a handshake under the session key to mutually confirm their possession of it.

For brevity we refer to protocol class defined above as NB+AH+SO, after Gong. It is informally proved by Gong that five messages is a lower bound for protocols of class NB+AH+SO. We may thus ask: What is the reason for the greater message efficiency achieved by Boyd's first protocol? To help answer this, we reproduce below a message-optimal protocol of class NB+AH+SO from Gong's paper [3]:

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

Figure 2: Message-optimal protocol of class NB+AH+SO

After execution of this protocol, A and B share a session key K_{AB} chosen newly by S. The handshake under K_{AB} additionally enables A and B to mutually confirm their possession of the session key. Notice that this protocol has the following property not shared by Boyd's first protocol: S knows the value of the session key. As is now explained, the lack of this property is central to the lowering of Gong's bound on messages by Boyd's first protocol. First, note that in Boyd's protocol the server S chooses the secret key input to the function defining the session key. Consider any protocol that is similar but that also leaves S in possession of the session key value. The following two assumptions are intuitively necessary for such a protocol:

1. A user (i.e., A or B) cannot send out a handshake message before receiving K_S and the other user's nonce.
2. The server must receive both users' nonces.

Following Gong [3], we assume that exactly one of the two users starts the protocol; thus, the other user and the server must first receive some message before sending out any message. On the basis of the assumptions made, we can informally prove that five is a lower bound on the number of messages

contained in the protocol. In the argument below, we refer to the user who starts the protocol as the initiator and to the other user as the responder.

Argument for lower bound: The protocol responder has to be notified of the start of the protocol before it can send out its nonce, which must be received by the server. The server has to distribute K_S in at least two messages. The message containing the last half of the handshake can be sent only after both users have received K_S. Thus, five messages is a lower bound.

The following is an example of a protocol that achieves this bound:

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

In this protocol, both A and B pass their nonces to the server S, and as a result S knows the session key. The cost of supporting this property is one extra message, as reflected by the lower bound of five messages obtained above. It is not difficult to see that without this property the lower bound is reduced by exactly one message to four messages; the reduced bound is achieved by the protocol shown in Figure 1.

By deleting the message parts comprising the session key handshake from Boyd's first protocol, we obtain the following protocol:

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_B

Figure 3: Boyd's first protocol with handshake removed

The protocol shown in Figure 3 is clearly message-optimal under Boyd's setting in the case where S chooses the secret key input to the function defining the session key, and where no handshake is included. Again, one more message is needed when it is assumed that S must know the session key value.

3 Boyd's round-efficient protocol

In the previous section, we looked at the improved message efficiency of Boyd's first protocol relative to protocols of class NB+AH+SO. As regards the number of rounds, Gong has informally proved that four rounds is a lower bound for the same class. The following example protocol of Gong achieves this bound [3]:

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

Figure 4: Round-optimal protocol of class NB+AH+SO

In the protocol shown in Figure 4, messages contained in the same round are grouped together and solid lines are used to indicate the beginning of each new round. We will henceforward use this notational device of Gong [4] to indicate the intended round structure of a given protocol.

Boyd has shown that his first protocol, appearing in Figure 1, can be rearranged to obtain another protocol that contains only three rounds, less than the minimum of four rounds required for class NB+AH+SO:

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.	$A \rightarrow B$:	$[N_B]_{K_{AB}}$

Figure 5: Boyd's second protocol

The protocol shown in Figure 5 is clearly round-optimal under Boyd's setting for the case where S chooses the secret key input to the function defining the session key, and where a handshake is included. As was the case in Boyd's first protocol, execution of the above protocol does not leave S in possession of the session key value. If we add the following message to the second round of the protocol shown in Figure 5:

$$5'. \quad B \rightarrow S: \quad B,\ N_B,\ N_A$$

then the resulting protocol contains the same number of rounds as the original but additionally provides S with the session key value. Thus, the lowering of Gong's bound on rounds by Boyd's second protocol is independent of whether the protocol leaves S in possession of the session key value. The protocol shown in Figure 5 is more round-efficient than, but functionally similar to, the round-optimal protocol of class NB+AH+SO shown in Figure 4. It achieves greater efficiency in the number of rounds because, unlike the latter protocol, it does not depend on a nonce handshake between each user and the server in order to prove session key freshness.

Consider the class of protocols that is defined similarly to the class NB+AH+SO except that it does not include a session key handshake. For brevity we refer to this protocol class as NB+AO+SO, after Gong [3]. It is informally proved by Gong that three rounds is a lower bound for protocol of class NB+AO+SO. As an example, we reproduce below a round-optimal protocol of the above class from Gong's paper [3]:

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

Figure 6: Round-optimal protocol of class NB+AO+SO

Clearly, in Boyd's setting two rounds is optimal for protocols where S chooses the secret input to the function defining the session key, when no handshake is included. The following protocol, which is

obtained from Boyd's second protocol by deleting the handshake messages, achieves this bound:

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

Figure 7: Boyd's second protocol with handshake removed

Notice that this protocol requires only two rounds, less than the minimum of three rounds required for class NB+AO+SO.

4 Discussion

Freshness by input		
lower bound	session key recipients	
msg/round	both users	both users + server
no handshake	3/2	4/2
handshake	4/3	5/3

Freshness by challenge-response	
lower bound	session key recipients
msg/round	both users + server
NB+AO+SO	4/3
NB+AH+SO	5/4

Table 1: Comparison of lower bounds on messages and rounds

Table 1 compares the proven lower bounds on messages and rounds under Boyd's setting with bounds for two specific classes of protocols where the session key is chosen by the server, and where session key freshness is achieved by using challenge-response.

Boyd's first protocol saves one message over any message-optimal protocol of class NB+AH+SO. If we remove the handshake from the former protocol, the resulting protocol saves one message over any message-optimal protocol of class NB+AO+SO. In both cases, the message efficiency gained in Boyd's setting is nullified under the assumption that the server must know the session key value. Clearly, this assumption is not necessary when one considers the minimal set of users to whom the session key needs to be conveyed for use subsequent to the protocol. However, the extent to which the above protocols with reduced bounds on messages can be seen to provide efficient alternatives to protocols that use challenge-response may depend on other factors. For example, in Boyd's setting, each user must provide an input into the equation that defines the session key so as to obtain confidence that the key value is fresh. A well-known drawback of this form of key generation, commonly called *key agreement* [1], is that the resulting key may be unsuitable for use in situations where there are quality requirements placed on key generation other than generating good random numbers. In general, such requirements are easily incorporated into protocols that rely on the server to generate the session key.

A distinctive aspect of Boyd's setting is that the freshness of the server's input into the equation that defines the session key is not required in order to assure the users of the freshness of the derived session key. This allows more round-efficient protocols to be designed. Boyd's second protocol saves one round

over any round-optimal protocol of class NB+AH+SO. If we remove the handshake from the former protocol, the resulting protocol saves one round over any round-optimal protocol of class NB+AO+SO. In both cases, better round-efficiency is achieved under Boyd's setting, regardless of whether the server is provided with the session key value.

The following table summarises the bounds found by Gong [3] for two specific classes of protocols where the session key is chosen by the server, and where session key freshness is achieved by using timestamps. The class denoted as TB+AO+SO (TB+AH+SO) is defined similarly to the class NB+AO+SO (NB+AH+SO) except that it uses timestamps instead of random nonces.

Freshness by timestamps	
lower bound msg/round	session key recipients both users + server
TB+AO+SO	3/2
TB+AH+SO	4/3

Table 2: Lower bounds for timestamp-based classes

If we compare the bounds given in Tables 1 and 2, it is evident that the method of achieving freshness by input can provide the same advantage in the number of messages and rounds as that provided by the use of timestamps over the use of challenge-response. As Boyd has noted [1], this method is also more favourable than using timestamps because it does not depend on clocks. However, other factors may need to be considered when choosing between protocols where freshness is achieved by input and protocols where freshness is achieved by timestamps. For example, the class TB+AO+SO enjoys the optimality property that it is possible to construct a protocol which achieves the corresponding lower bounds on messages and rounds simultaneously. We reproduce below an example from Gong's paper [3] of a protocol having this property:

1.	$A \rightarrow S$:	A, B
2.	$S \rightarrow A$:	$\{B, K_{AB}, T_S\}_{K_{AS}}$
3.	$S \rightarrow B$:	$\{A, K_{AB}, T_S\}_{K_{BS}}$

Figure 8: An optimal protocol of class TB+AO+SO

Here, T_S is a timestamp generated by S. In this protocol, A is convinced of the freshness of the session key, provided that the timestamp in the message encrypted under K_{AS} is within a reasonable interval of A's local time; and analogously for B. Consider the class of protocols in Boyd's setting where the server S chooses the secret key input to the function defining the session key, and where no handshake is included. The lower bounds on messages and rounds for this class under the assumption that S goes uninformed of the session key value are the same as for class TB+AO+SO. However, in the class where freshness is achieved by input it appears impossible to construct a protocol that achieves the corresponding lower bounds on messages and rounds simultaneously. An informal argument in support of this claim is given below. It follows the style of reasoning used by Gong [4] to argue similarly impossible protocols for several other protocol classes where session key freshness is achieved by using challenge-response or timestamps.

Argument for the above claim: Suppose that the two lower bounds of three messages and two rounds are achieved by the same protocol. A two-round protocol must include at least three messages, one each corresponding to the following events:

1. S being notified of protocol start during the first round.

2. A receiving K_S during the second round.

3. B receiving K_S during the second round.

Suppose B is notified in the second round. Then at least one more round is needed for B to send N_B to A. Suppose B is notified in the first round. Then a fourth message is needed for this purpose.

5 Conclusions

We have given some results on lower bounds on messages and rounds for a class of protocols introduced by Boyd where session key freshness is achieved by input. We have also compared the similarities and differences between these bounds and some bounds published by Gong for protocols where session key freshness is achieved by using either challenge-response or timestamps. The results of our comparison can be of use to designers of protocols of practical systems. Where greater efficiency is achieved by trading off some protocol property, it is important to understand the issues involved in order to select efficient solutions.

Acknowledgements

Colin Boyd read drafts of this paper and made constructive comments. An anonymous reviewer suggested improvements to a draft. Michael McDonald provided editorial assistance.

References

[1] C. Boyd. Towards a Classification of Key Agreement Protocols. In *Proceedings of the 8th IEEE Computer Security Foundations Workshop*, pages 38–43, County Kerry, Ireland, June 1995.

[2] C. Boyd. A Class of Flexible and Efficient Key Management Protocols. In *Proceedings of the 9th IEEE Computer Security Foundations Workshop*, pages 2–8, County Kerry, Ireland, June 1996.

[3] L. Gong. Lower Bounds on Messages and Rounds for Network Authentication Protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 26–37, Fairfax, Virginia, November 1993.

[4] L. Gong. Efficient network authentication protocols: lower bounds and optimal implementations. *Distributed Computing*, 9(3):131–145, 1995. Springer-Verlag.

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

[6] R. Needham and M. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, 21(12):993–999, 1978.