Chapter 17

TIMESTAMPS FOR NETWORK AUTHENTICATION PROTOCOLS REVISITED

Chandana Gamage
Peninsula School of Computing and Information Technology
Monash University, McMahons Road, Frankston, Vic 3199, Australia
chandag@pscit.monash.edu.au

Jussipekka Leiwo
Vrije Universiteit, Department of mathematics and computer science
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
leiwo@cs.vu.nl

Yuliang Zheng
Peninsula School of Computing and Information Technology
Monash University, McMahons Road, Frankston, Vic 3199, Australia
yuliang@pscit.monash.edu.au

Abstract

In network security protocols, the freshness and uniqueness of a particular protocol-run provide a strong defense against replay attacks. These two properties are incorporated into a protocol interaction by the use of time-variant parameters such as nonces, random numbers, sequence numbers and timestamps. Many of the network authentication protocols can be classified into two main streams by the type of time-variant parameter they use; nonce or timestamp. Network authentication protocols based on timestamps are always more efficient in terms of numbers of rounds to their nonce-based counterparts. However, regardless of this favorable operational characteristic, most widely used protocols favor nonce-based schemes over timestamps. The main reason for this neglect of timestamp-based schemes was the difficulty of establishing synchronized clocks over a widely distributed network infrastructure spanning multiple administrative domains. Another reason was the need to develop protocols that are efficient in terms of number of messages at the cost of number of rounds to optimize the bandwidth use of standard network links.

Availability of new technology such as global positioning system (GPS) signaling for distributed clock synchronization and the need for round efficient network authentication protocols to support secure communication in high bandwidth/high speed networks warrant a fresh look at timestamp-based network authentication protocols under new high performance distributed computing infrastructure. In this paper, we discuss the usefulness of time-based network authentication protocols for secure high speed networks and use of GPS to overcome the main difficulty in timestamp-based schemes. We then review several important protocol optimization research works and select a simple timestamp-based authentication protocol with an optimum lower bound for protocol round efficiency and formally prove its security properties under BAN logic constraints. Then the selected protocols efficiency is analytically studied and the results are compared to data from a simulation experiment.

Keywords:

Network security, Authentication protocols, Timestamps, GPS, High speed networks

1. INTRODUCTION

One of the important developments in the area of high speed distributed computing is gigabit networks (such as vBNS[1], NGI[2] and I2[3])which provide the speed and capacity required by advanced applications. These high speed networks are characterized by the high *bandwidth* $\times$ *delay* property. In data communication networks, the total delay in sending a packet from one node to another consists of two basic components:

1. The *transmission delay*, which is the time for all the bits of the data packet to arrive at the destination node and
2. The *propagation delay*, which is the time for the first bit of the data packet to reach the destination.

As the data rate of networks increase, the transmission delay which is a property of the data rate (bandwidth), correspondingly decreases. However, the propagation delay which is a property of the distance a

[1]very high performance Backbone Network Service `http://www.vbns.net/`
[2]Next Generation Internet `http://www.ngi.gov/`
[3]Internet2 `http://www.internet2.edu/`

data packet must travel and speed of light remains constant. Therefore, the overall delay for packet transfer in gigabit networks is increasingly dominated by propagation delay.

Consequently, these delay properties have a significant impact on how high performance protocols for gigabit networks are developed:

- The size of a data packet becomes an insignificant factor at gigabit rates due to low transmission delay. In the design of network protocols, this feature emerge in the form of large packet headers that carry more control information (possibly redundant) and mapping of application protocol data units (PDUs) into network PDUs without segmenting.
- The number of sequential rounds as well as the number of individual packets (messages) in a protocol becomes a dominant issue due to relatively high propagation delay. As an example, network protocol designs try to adapt to this situation by using forward error correction through increased payload redundancy rather than error recovery through retransmission.

As we enhance network protocols with security functionality or extend network protocol stacks with security service layers, it is important to consider above two factors more carefully. This will ensure that implanting of network security function in high performance protocols does not significantly reduce the protocol efficiency and thus their practical value.

In network security protocols, the freshness and uniqueness of a particular protocol-run provide a strong defense against replay attacks. These two properties are incorporated into a protocol interaction by the use of time-variant parameters such as nonces, random numbers, sequence numbers and timestamps. Many of the network authentication protocols can be classified into two main streams by the type of time-variant parameter they use; nonce or timestamp. In the work presented in this paper, we demonstrate that both in theory and practice, network authentication protocols based on timestamps to be always more efficient in terms of number of rounds to their nonce-based counterparts.

Structure of the paper. In section 2., we formulate a security model for use in high speed network environment, where timestamp-based protocols are most likely to provide performance gains. In section 3., we review different protocol performance improvement schemes to establish the efficiency ranking of timestamp-based protocols. This is followed by the security analysis of a selected protocol using BAN logic in section

4.. The performance of selected protocol schemes were experimentally evaluated in section 5. and final observations and results are given in section 6..

The main contribution of the work presented in this paper is to both analytically and experimentally demonstrate the possibility of using timestamp-based network authentication and key distribution protocols in high performance distributed computing infrastructure.

2. DESIGN CRITERIA FOR KEY DISTRIBUTION MODELS

Key distribution is the process in which participating entities ultimately share a secret key for cryptographic computations. We limit our discussion to the sharing of a key between two parties only. The problem of group key distribution is not discussed here. The solutions to the problem of secure key distribution can be broadly classified between schemes based on symmetric key cryptography and public key cryptography. Key distribution schemes can be further classified as key agreement or key transport. For public-key cryptography based models, the key transport problem is defined as one entity selecting the key material and securely distributing it over a network to another entity (for example, using public key encryption) and if the process includes assurances on participants identities (for example, using digital signatures), it is called authenticated key transport [22]. For both key agreement and key transport schemes based on public-key techniques, the end-point entities need to share only authenticated information (public keys) in advance [8, 16] and the major mechanism for this are the digital certification authorities structured into a hierarchy as shown in figure 17.1.

Network entity authentication and key distribution protocols based on symmetric key cryptography and long-term secrets shared with a trusted server were introduced by Needham and Schroeder [17]. In an important study of network authentication protocol efficiencies done by Gong [10, 11], the authentication protocols were classified according to following criteria:

1. Freshness of the protocol run: using timestamps (with synchronized clocks) or nonces.

2. Secret key generation : by the trusted server, by only one of the clients or by the participation of both clients.

3. Authentication (key distribution) only or authentication with handshake (key distribution and key confirmation).

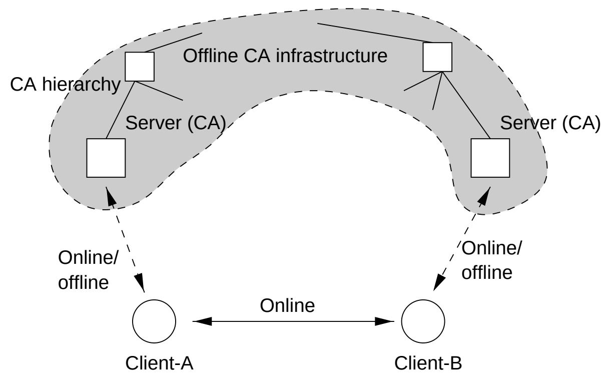

Figure 17.1 Session key distribution using public key certificates. The interaction with CA may be online or offline depending on the need for certificate verification

The efficiency of an authentication protocol was measured against two metrics:

1. The total *number of messages* exchanges in a successful protocol run.

2. The total *number of rounds* in a successful run. A round is set of message exchanges between participating entities that can occur simultaneously.

For entity authentication and key distribution protocols for high speed networks, we suggest following criteria as most crucial:

1. Symmetric key cryptography based solutions that allow faster computations relative to asymmetric (or public) key based schemes.

2. Server assisted key management including key generation to allow use of dedicated hardware and also to facilitate certification of key servers (see section 2.1).

3. Minimum overall delay due to key distribution protocol run (see section 2.2).

4. Mutual or partial confirmation of session key delivery as part of the authentication protocol as this is a common secure application requirement.

2.1 SESSION KEY GENERATION BY A TRUSTED SERVER

It is not feasible to certify every network entity as capable of generating cryptographically secure key material. There are two practical solutions to this problem:

1. Key material generation by trusted servers where the servers can be setup and certified through an off-line procedure.
2. Configuration of network entities with a plug-in cryptographic module (for example, a smart card) to ensure the cryptographic strength of generated key material. The cryptographic modules can be constructed according to standardized specifications and certified.

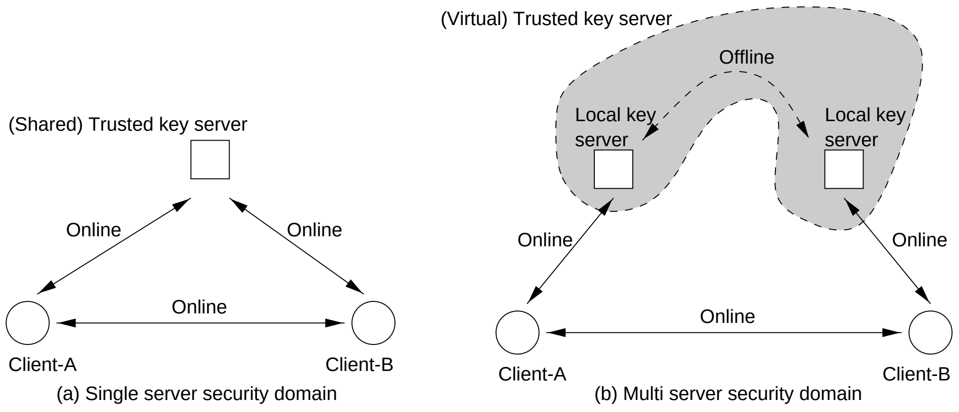
(Shared) Trusted key server
Online
Online
Online
Client-A
Client-B
(a) Single server security domain
(Virtual) Trusted key server
Offline
Local key server
Local key server
Online
Online
Online
Client-A
Client-B
(b) Multi server security domain

Figure 17.2 Session key distribution by a trusted key server with shared secret keys. In the case of virtual key server, the complexity of key management within the server group is hidden to the client nodes

As the number of trusted servers required for secure communication is much less than the actual number of network entities that would require client services, use of trusted servers is a practical solution. The infrastructure requirement in this case is analogous to routers and hosts. For this solution, either trusted servers can be individually setup or selected servers can be configured with cryptographic modules. Apart from the direct use as a certified key server, a trusted server can provide a wide array of other symmetric key management services, such as

- Key backup for short term during a protocol run.

- Key archival for long term storage to comply with legislative or business rules.
- Key recovery from backup for non-compromised key loss while in use or from archives for others needs (for example, dispute resolution).
- Off-loading of computational workload of key generation from general network entities to dedicated servers. This will reduce processing delays at hosts.

Following from the need for a wide array of key management services and reduction of processor load at network nodes, we suggest the use of a trusted server for key generation. Two possible configurations for setting up a trusted server is illustrated in figure 17.2.

2.2 NUMBER OF ROUNDS IN A PROTOCOL RUN

In high speed networks, the total message transfer latency consists of transmission delay at network entities and propagation delay over high speed links (for example, fiber optic). The transmission delay component can be further reduced by increasing the processing capacity at network entities using techniques such as faster processors, hardware implementation of critical-path code and parallelization. However, the propagation delay is bounded by the physical properties of the transmission media and limited by the speed of light for high speed links. In this situation, it is important for authentication protocols for high speed network communication to reduce the number of rounds even at the cost of increased number of messages as the number of rounds determine the overall delay for secure communication.

An evaluation of network authentication protocols based on the number of rounds in a successful run reveals that protocols based on timestamps for freshness are more efficient than those based on nonces. For similar network authentication protocols that differ only in their scheme for freshness guarantee (time or nonce), the transmission delay of timestamp-based protocol is at least one complete round less than its nonce-based counterpart.

The main argument against the use of timestamp-based key distribution protocols was the need for synchronized clocks (for example, see [9]). Until recently, setting up of synchronized clocks and periodic re-synchronization was a complicated and resource consuming process. However, the availability of globally distributed clock synchronization infrastructure such as global positioning system (GPS) signaling [3, pages

433-437] make cost and complexity arguments against key distribution protocols using time synchronization less forceful. A GPS based network time protocol (NTP) server in a LAN (see figure 17.3) can provide clock synchronization in the millisecond range.

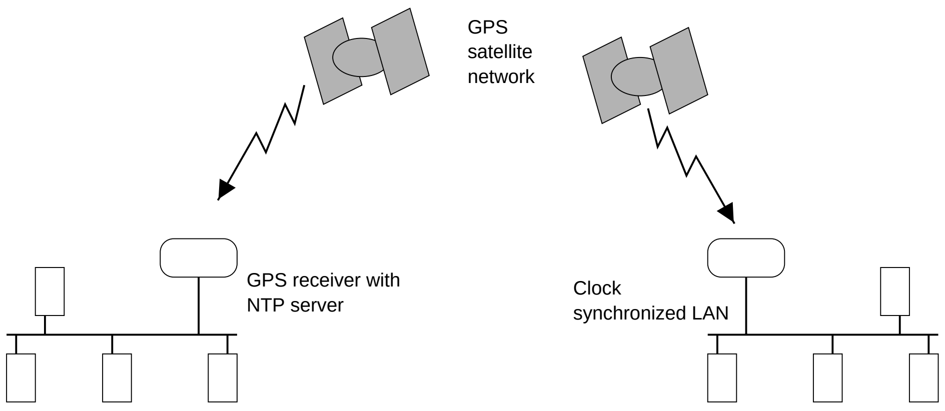

Figure 17.3 GPS based distributed clock synchronization for wide area networks

Our main objective in presenting the above discussion is to justify the use of time-based authentication protocols that make trade-offs in favor of reducing the number of rounds in a successful protocol run for use in high speed network environments.

3. ANALYSIS OF KEY DISTRIBUTION PROTOCOL EFFICIENCIES

The optimum lower bound for a key distribution protocol using timestamps for freshness has only two messages and two rounds. An example scheme with this optimality is Yahalom's Wide-Mouthed-Frog protocol [5]. Although, that protocol is optimal in both messages and rounds, the secret key is generated entirely by user A. The trusted server S is used simply to create a secure tunnel for key distribution. Thus, the Wide-Mouthed-Frog protocol uses a weaker security model that requires one end-point to implicitly trust the key generation capability of the other end-point. The next best lower bound for an improved security model has three messages and two rounds. An example of this scheme is Denning and Sacco [7] protocol. In their protocol security model, the secret key is generated by the trusted server S.

3.1 REVIEW OF AUTHENTICATED KEY DISTRIBUTION PROTOCOLS DESIGNED FOR ROUND EFFICIENCY

As described in section 2., the *security model* for authentication in high speed networks that we prefer should use a trusted server S for key generation and also provide a handshake phase for confirmation of mutual authentication and key distribution. Next we review some of the important research literature that had a significant focus on protocol performance metrics, performance measurement and techniques to improve performance. Only those protocol schemes that map into the security model described earlier are reviewed here. We use a standard notation devised by Gong [11] to illustrate the protocol steps in a scheme which separate concurrently run protocol steps (rounds) by a horizontal line. Each encrypted message (shown as $\{\ldots\}_{key}$) is formatted such that it indicates the identities of sender and receiver (shown as A,B,etc). If a key is included in the message, we also indicates the identities of the parties for which the key is intended to comply with the explicitness principle as described by Anderson and Needham [1, 2]. T_x and N_x denote a timestamp and a nonce generated by entity with identity X respectively.

Gong's clock-based round-efficient protocol (1993). In [10], an informal proof for round efficiency of clock-based protocols was given with an example protocol scheme similar to Kerberos [19]. In the scheme shown below, the session key K_{ab} is generated by S and the cryptographic security (that is, strength) of the session key depends on S.

$$\begin{array}{llcll}
1. & A & \rightarrow & S & : \text{A,B} \\
\hline
2. & S & \rightarrow & A & : \{\text{S,A,A,B},K_{ab},T_s\}_{K_{as}} \\
3. & S & \rightarrow & B & : \{\text{S,B,A,B},K_{ab},T_s\}_{K_{bs}} \\
\hline
4. & A & \rightarrow & B & : \{\text{A,B},T_a\}_{K_{ab}} \\
5. & B & \rightarrow & A & : \{\text{B,A},T_b\}_{K_{ab}}
\end{array}$$

Gong's nonce-based round-efficient protocol (1993). This is the nonce-based counterpart of the above protocol and an informal proof for round efficiency was given in [10] with an example protocol scheme derived from a protocol by Kehne et al. [13]. As before, the session key K_{ab} is generated by S.

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,B,K_{ab},$N_a\}_{K_{as}}$,N_b
4.	S	$\rightarrow$	B	:	$\{$S,B,A,B,K_{ab},$N_a\}_{K_{bs}}$
5.	A	$\rightarrow$	B	:	$\{$A,B,$N_b\}_{K_{ab}}$
6.	B	$\rightarrow$	A	:	$\{$B,A,$N_a\}_{K_{ab}}$

Kao-Chow nonce-based round-efficient protocol (1995). A scheme described by Kao and Chow in [12] makes careful use of *uncertified keys* in session key distribution protocols to achieve both round and message optimal efficiency. The design principles for the scheme are based on the authentication protocols by Kehne, et.al [13] (that strengthen authentication goals) and Neuman and Stubblebine [18] (that reduce number of messages). In this protocol A does not receive any message directly from the trusted server S. The shared session key K_{ab} is generated by S.

1.	A	$\rightarrow$	S	:	A,B,N_a
2.	S	$\rightarrow$	B	:	$\{$S,A,A,B,K_{ab},$N_a\}_{K_{as}}$, $\{$S,B,A,B,K_{ab},$N_a\}_{K_{bs}}$
3.	B	$\rightarrow$	A	:	$\{$S,A,A,B,K_{ab},$N_a\}_{K_{as}}$, $\{$B,A,$N_a\}_{K_{ab}}$,N_b
4.	A	$\rightarrow$	B	:	$\{$A,B,$N_b\}_{K_{ab}}$

Boyd's round-efficient protocol (1996). For key distribution protocols using either timestamps or nonces, the freshness of the key is determined by each user receiving a message in which a timestamp or a nonce value is bound to the key being distributed. In the authentication protocol design concept described by Boyd in [4], he noted that the same effect can be achieved by each user giving a fresh input value to a suitably chosen function. This removes the need for a message exchange to determine the freshness. Also, by giving a shared secret value as input to the function, the output value will be known only to those parties that contributed the inputs. The cryptographic function required in Boyd's scheme must have the properties of a keyed-hash function or MAC (for an example description on fast MAC construction see [20]). In the protocol scheme shown below, the session key K_{ab} is the output of MAC function keyed with shared secret K_s which is generated by S. The trusted server S does not know the session key K_{ab} as it did not receive the nonce values N_a and N_b which is concatenated and given as input to the MAC function. The protocol can allow S to compute the session key without increasing the number of rounds by having B transmit the pair of values N_a and N_b to S in round three at the cost of an extra message [14].

1.	A	$\rightarrow$	B	:	A,B,N_a
2.	A	$\rightarrow$	S	:	A,B
3.	S	$\rightarrow$	A	:	$\{$S,A,A,B,$K_s\}_{K_{as}}$
4.	S	$\rightarrow$	B	:	$\{$S,B,A,B,$K_s\}_{K_{bs}}$
5.	B	$\rightarrow$	A	:	B,A,N_b
A and B compute $K_{ab} = f(K_s, N_a \mid N_b)$					
6.	A	$\rightarrow$	B	:	$[$A,B,$N_b]_{K_{ab}}$
7.	B	$\rightarrow$	A	:	$[$B,A,$N_a]_{K_{ab}}$

3.2 KEY DISTRIBUTION PROTOCOL PERFORMANCE COMPARISON

The techniques used in above nonce-based protocol schemes to improve efficiency and further lower the bounds on number of rounds can be summarized as follows:

1. Designing protocol schemes with number of rounds closer to the three stages of authenticated key distribution (initiation, transport and handshake).
2. Use of uncertified keys *within* a protocol run.
3. Deriving session key freshness by using input from end-point entities.
4. Limiting a trusted server's knowledge about the session key.

The efficiency comparison shown in table 17.1 clearly show that above techniques have failed to improve the performance of nonce-based protocols above the performance of timestamp-based schemes. We use this analysis to forward our argument that timestamp-based network authentication and key distribution protocols are a mechanism to improve performance of secure applications in high speed networks.

Another factor which can affect both the performance and the security of these symmetric key protocol schemes is message encryption. The encoding of a message M with a secret key k as $\{M\}_k$ should preferably be done using CBC mode to prevent guessing-and-modifying type of attacks, especially for timestamps. For example, the OFB mode which encode as $\{M\}_k = G(k) \oplus M$, where G is a suitable pseudo-random function, is unsuitable for authentication message encryption. However,

Table 17.1 Analytical performance comparison of authenticated key distribution Protocols

Round Efficient Authenticated Key Distribution Protocols

Protocol	*Rounds*	*Messages*
Gong - using nonces	4	6
Kao-Chow - using nonces	4	4
Boyd - using nonces	3	7
Gong - using timestamps	3	5

Message Efficient Authenticated Key Distribution Protocols

Protocol	*Rounds*	*Messages*
Gong - using nonces	4	5
Kao-Chow - using nonces	4	4
Boyd - using nonces	4	4
Gong - using timestamps	4	4

it could be strengthen by modifying the encryption as $\{M\}_k = G(k) \oplus (M|f(k, M))$.

4. FORMALIZED SECURITY ANALYSIS OF A TIMESTAMP BASED KEY DISTRIBUTION PROTOCOL

The timestamp-based network authentication and session key distribution protocol that we prefer to use in high speed network environments is the example protocol given by Gong for round efficiency. As the original description of the protocol assumed its efficiency only if the protocol is proven to have the desired security properties, in this section we provide formal arguments on the protocol security. The formal analysis of a network security protocol using BAN logic [5] involves following steps: (1) Converting original protocol statements to their idealized form. (2) Determining the assumptions about the initial state of the system. (3) Representation of the state of the system after executing each statement as logical assertions by attaching logical formulas to each statement. (4) Application of logical postulates to assumptions and assertions. The result of this exercise is to obtain the beliefs held by the parties involved in the protocol. This four-step procedure may be repeatedly applied for newly discovered assumptions and for further refinement of protocol statements in idealized form.

In the following analysis of the authenticated key distribution protocol with confirmation, We use the same notation used in original BAN logic paper [5]. P and Q are parties involved in a protocol. K is a cryptographic key. X is a logical formula involving possibly P, Q, K and other entities. In this setting, an n step protocol system can be represented in the following format:

$$[assumptions]S_1[assertion_1]S_2 \ldots S_{n-1}[assertion_{n-1}]S_n[conclusions]$$

where each statement S_i is of the form $\quad P \to Q : X$ with $P \neq Q$

Idealized protocol statements. The first step in the formal analysis of the key distribution protocol using BAN logic is to convert the message exchanges (see section 3.1 and also figure 17.4) to idealized forms.

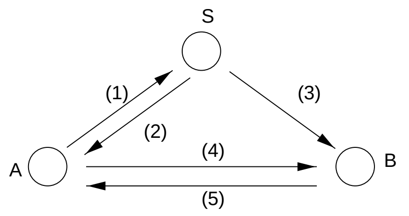

Figure 17.4 Session key distribution using a trusted server with key confirmation

1. The first message is in clear-text and therefore provides no security guarantees.

2. The second message contains the identities of A and B together with shared session key K_s and a timestamp T_s encrypted by K_{as}. Thus, A knows that the key K_s is for communication between the two parties and the timestamp guarantees its freshness.

   $\{(A \overset{K_s}{\leftrightarrow} B), \sharp(A \overset{K_s}{\leftrightarrow} B)\}_{K_{as}}$

3. The security guarantees provided to B by the third message is similar to above.

   $\{(A \overset{K_s}{\leftrightarrow} B), \sharp(A \overset{K_s}{\leftrightarrow} B)\}_{K_{bs}}$

4. As T_s was encrypted by a shared secret key K_{as}, A knows that $S \mid\!\sim T_s$. This time data is used by A to calculate the message

acceptance time window. The idealized form of the message also encodes the fact that the fourth message is being sent as a result of receiving previous message two.

$\{\sharp(A \overset{K_s}{\leftrightarrow} B), (A \overset{K_s}{\leftrightarrow} B), (S \mid\!\sim T_s)\}_{K_s}$ from A

5. The security guarantees provided to B by the fifth message is similar to above. We explicitly identify the sender of the message to differentiate between the last two messages in their idealized form.

   $\{\sharp(A \overset{K_s}{\leftrightarrow} B), (A \overset{K_s}{\leftrightarrow} B), (S \mid\!\sim T_s)\}_{K_s}$ from B

Assumptions about the protocol system. The second step is to determine the initial assumptions about the system which is shown in table 17.2.

Table 17.2 Initial assumptions of the protocol system

Beliefs on security of shared keys:	
1. $A \mid\equiv A \overset{K_{as}}{\leftrightarrow} S$	
2. $S \mid\equiv A \overset{K_{as}}{\leftrightarrow} S$	
3. $B \mid\equiv B \overset{K_{bs}}{\leftrightarrow} S$	
4. $S \mid\equiv B \overset{K_{bs}}{\leftrightarrow} S$	
5. $S \mid\equiv A \overset{K_s}{\leftrightarrow} B$	This statement indicates servers knowledge about a secret key that will finally be shared between A and B.
Beliefs on server's ability to generate secure shared keys:	
1. $A \mid\equiv (S \mid\Rightarrow A \overset{K}{\leftrightarrow} B)$	
2. $B \mid\equiv (S \mid\Rightarrow A \overset{K}{\leftrightarrow} B)$	
Beliefs on server's ability to honestly forward messages to another participant:	
1. $A \mid\equiv (S \mid\Rightarrow (B \mid\!\sim X))$	
2. $B \mid\equiv (S \mid\Rightarrow (A \mid\!\sim X))$	
Beliefs on correctness (that is, freshness) of timestamps by their recipients when the timestamp originates from a trusted server:	
1. $A \mid\equiv \sharp(T_s)$	
2. $B \mid\equiv \sharp(T_s)$	

Logical postulates. The third step is to apply the postulates of the BAN logic and the annotation rules to the available formulas to prove beliefs on the protocol outcome. At the beginning of the protocol, A sees the complete message sent by S.

$$A \triangleleft \{(A \overset{K_s}{\leftrightarrow} B), \sharp(A \overset{K_s}{\leftrightarrow} B)\}_{K_{as}} \tag{17.1}$$

After decrypting the above message A can convince herself of the shared keys freshness and the fact that K_s behaves as a nonce. As the message was encrypted under the long-term shared secret key K_{as}, A is convinced that S has seen the clear-text message. Now, we apply the nonce-verification postulate as follows

$$\frac{A \models \sharp(A \overset{K_s}{\leftrightarrow} B), A \models S \mid\sim (A \overset{K_s}{\leftrightarrow} B)}{A \models S \models (A \overset{K_s}{\leftrightarrow} B)} \tag{17.2}$$

Next, we apply the jurisdiction postulate to infer the following.

$$\frac{A \models S \Rightarrow (A \overset{K_s}{\leftrightarrow} B), A \models S \models (A \overset{K_s}{\leftrightarrow} B)}{A \models (A \overset{K_s}{\leftrightarrow} B)} \tag{17.3}$$

Due to the symmetry of the protocol statements, we can obtain similar beliefs for B by applying the nonce-verification and jurisdiction postulates.

$$B \models (A \overset{K_s}{\leftrightarrow} B) \tag{17.4}$$

After B receives the message in protocol statement four from A, B is convinced of the existence of A, the possession of the session key by A and the message was sent recently (within the acceptance time window allowed by synchronized clocks).

$$\frac{B \models \sharp(A \overset{K_s}{\leftrightarrow} B), B \models A \mid\sim (A \overset{K_s}{\leftrightarrow} B)}{B \models A \models (A \overset{K_s}{\leftrightarrow} B)} \tag{17.5}$$

As before, exploiting the symmetry of the protocol statements, we can infer following belief.

$$A \models B \models (A \overset{K_s}{\leftrightarrow} B) \tag{17.6}$$

Final beliefs. The final result of the formal protocol analysis is the derivation of following beliefs.

1. $A \models (A \overset{K_s}{\leftrightarrow} B)$
2. $B \models (A \overset{K_s}{\leftrightarrow} B)$
3. $A \models B \models (A \overset{K_s}{\leftrightarrow} B)$
4. $A \models B \models (A \overset{K_s}{\leftrightarrow} B)$

Summary of analysis. The formal analysis provides strong security arguments for the timestamp-based authenticated key distribution protocol. Before any data transmission occurs, the end-point principals are confident of each others exclusive knowledge of the shared secret. If the secret session key is computed as $K_{ab} = f(K_s, T_a|T_b)$ using a suitable keyed hash function f, the freshness of K_{ab} will be guaranteed by both users A and B. The user B is convinced of the key freshness of K_s if the timestamp T_a is within its acceptance window. The final construction of the session key K_{ab} ensure that none of the participants can force the use of an old session key even by collusion. The final formulation of the session key K_{ab} using timestamps ensure the key freshness irrespective of guarantees provided by trusted server S. As none of the participants make any assumptions about the freshness of the session key, the protocol is not susceptible to the Denning and Sacco type of attack [7] on the Needham-Schroeder protocol [17] whereby an attacker has unlimited time to find an old session key and reuse it. In the protocol analyzed here, a session key being transmitted is always bound to a timestamp to explicitly prevent this type of attack.

5. SIMULATION OF TIMESTAMP BASED KEY DISTRIBUTION PROTOCOLS

The simulation of network authentication protocols that we have studied in section 3. was done on a 10Mbit/s switched LAN consisting of single CPU (Intel PII 333MHz with 64MB of RAM) machines running Sun Solaris operating system. The objective of the simulation was to obtain timing values of the protocol runs for the purpose of efficiency comparison. The measurements were taken as follows:

1. Time for protocol run completion at each host Alice (t_A), Bob (t_B) and the trusted Server (t_S).
2. Time for full completion of the protocol run: $t_{run} \max\{t_A, t_B, t_S\}$ (maximum time taken by the hosts participating in the scheme).

Additionally, above measurements were taken with null cryptographic computations to determine the communication overhead of the protocols. These values were used to check the accuracy of the timing measurements taken for the communication part of the full protocol runs. The simulation program used the Berkeley socket interface [23] to the operating systems network subsystem and executed as user-space processes. The first version of the program which used connection-oriented TCP streams introduced excessive network overhead to the timing measurements and made it difficult to gather data for any meaningful efficiency comparison of different protocols. Thereafter, the simulation program was modified to use unreliable UDP datagrams for protocol message transfer.

Multi-thread deadlock on uni-processor machines. The initial simulation program using unreliable UDP datagrams for communication suffered from a deadlock problem due to the hardware environment we have used. The message transmitting send() call is non-blocking while the message receiving recv() call blocks by running in a closed-loop until a data packet is available at its designated network port address. Although the programs were multi-threaded, the hosts that executed them were single processor machines. Therefore, due to the relatively low latency of the switched LAN, during the acknowledgment phase of the protocol run one of the end-point hosts had its recv() thread suspended while the send() thread is executing. If the host received a message from the other end-point at this time, the data packet was lost due to the unreliable nature of datagram transfer. After, the recv() thread returns to the run state it indefinitely blocks waiting for the lost acknowledgment message. The deadlock was removed by serializing the usually concurrent handshake message transfer between the two end-point hosts without implementing additional reliable message delivery primitives for the protocol. Although this affects the individual results of the simulation, it does not introduce errors to the comparison of protocol efficiency.

Timing loop synchronization for the protocol runs. The execution times for each protocol run at separate hosts were calculated by obtaining the number of system clock ticks consumed by each process loop (the number of clock ticks per millisecond is given by the system configuration value CLK_TCK). The timing loops were synchronized by first starting the hosts that are initially on receive mode (Trusted Server and Bob) and then running the protocol run initiator (Alice). Multiple simulation values were obtained by repeating the process for all hosts with a fixed startup delay for host running Alice.

Simulation results. The average time taken by each host's main processing loop was calculated in milliseconds and time for completion of

Table 17.3 Simulated performance comparison of round efficient authenticated key distribution protocols (values in milliseconds)

Protocol	*Total time with communication workload only*	*Total time with cryptographic and communication workloads*
Gong - using nonces	87	267
Kao-Chow - using nonces	114	312
Boyd - using nonces	51	202
Gong - using timestamps	56	195

protocol runs are shown in table 17.3. As the experimental setup was a switched LAN, for each protocol the time for communication related workload is smaller than for the cryptographic workload. However, in an actual high speed WAN, the longer distances and the underlying MAC layer protocol overheads (for example, ATM switching times for virtual connection setup [6]) will make the time for cryptographic processing much less than the time for data transmission.

6. CONCLUSION

The simulation experiments as well as the analytical evaluations we have done to measure the relative efficiency of selected protocols have allowed the us to conclude that timestamp-based protocols are favorable candidates to implement network authentication schemes suitable for high speed network environments.

Additionally, we were able to make several interesting observations. Although we expected Gong's timestamp-based protocol to achieve the best efficiency value for the average time to complete a protocol run, Boyd's nonce-based protocol achieved similar results. However, the total workload (measured by the total CPU time consumed by all the threads) at end-points for Boyd's scheme was significantly higher (by almost a factor of two) than for Gong's scheme.

Another interesting simulation result was the efficiency characteristics of Kao-Chow scheme. The actual communication overhead (measured by the average time to complete a protocol run) obtained experimentally was much higher than implied by the scheme's analytical evaluation in comparison to other protocol schemes. We attribute this particular result to the serialized nature of the Kao-Chow protocol scheme which does not permit communication tasks (in a multi-threaded environment) to overlap each other and reduce the total time for completion. In comparison, Boyd's scheme is a good example of a protocol that successfully hide its total delay by concurrent running of protocol steps. This observation highlights the point that comparison of analytical values on a protocols lower bound for a selected metric (in this instance, number of rounds) by themselves are not very accurate indicators of performance in an actual implementation. In [21], Reiter and Stubblebine present a different perspective on selection of metrics that are not performance related but quantifies the level of assurance. Their work seek to harmonize the various models for authenticity (such as PGP [24] and Maurer [15]) and then evaluate the level of assurance (as a numeric value) that can be achieved on the authenticity of entities by different protocols.

Finally, it should be noted that a trusted server's knowledge about the session key shared by end-point clients is also a factor in the protocol efficiency. Techniques to improve the efficiency lower bounds of a protocol may result in the trusted server not knowing the final session key. This may have implications in some operating environments where the trusted server must store the actual key (for example, corporate key escrow) and extra messages would be required to deliver the session key to the trusted server, as in the case of Boyd's round efficient protocol.

References

[1] R. J. Anderson and R. M. Needham. Programming Satan's computer. In J. van Leeuwen, editor, *Computer Science Today. Recent Trends and Developments*, volume 1000 of *Lecture Notes in Computer Science*, pages 426–440. Springer-Verlag, New York, NY, 1995.

[2] R. J. Anderson and R. M. Needham. Robustness principles for public key protocols. In D. Coppersmith, editor, *Advances in Cryptology - CRYPTO'95*, volume 963 of *Lecture Notes in Computer Science*, pages 236–247, Santa Barbara, CA, August 1995. Springer-Verlag.

[3] K. P. Birman. *Building Secure and Reliable Network Applications.* Manning Publications Co, Greenwich, CT, 1996.

[4] C. A. Boyd. A class of flexible and efficient key management protocols. In *Proceedings of the 9th IEEE Computer Security Foundations Workshop*, pages 2–8, Ireland, June 1996. IEEE Computer Society Press.

[5] M. Burrows, M. Abadi, and R. M. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, February 1990.

[6] S. Chatterjee. Requirements for success in gigabit networking. *Communications of the ACM*, 40(7):64–73, July 1997.

[7] D. E. Denning and G. M. Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, 24(8):533–536, August 1981.

[8] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[9] L. Gong. A security risk of depending on synchronized clocks. *Operating Systems Review*, 26(1):49–53, January 1992.

[10] L. Gong. Lower bounds on messages and rounds for network authentication protocols. In V. Ashby, editor, *Proceedings of the 1st ACM Conference on Computers and Communications Security (CCS'93)*, pages 26–37, Fairfax, Virginia, November 1993. ACM Press.

[11] L. Gong. Efficient network authentication protocols: Lower bounds and optimal implementations. *Distributed Computing*, 9(3):131–145, December 1995.

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

[13] A. Kehne, J. Schonwalder, and H. Langendorfer. A nonce-based protocol for multiple authentication. *Operating Systems Review*, 26(4):84–89, October 1992.

[14] A. Mathuria. Comparing lower bounds on messages and rounds for two classes of key establishment protocols. *SIGCOMM Computer and Communication Review*, 28(5):91–98, October 1998.

[15] U. Maurer. Modeling a public-key infrastructure. In E. Bertino, H. Kurth, G. Martella, and E. Montolivo, editors, *Proceedings of the 4th European Symposium on Research in Computer Security (ESORICS'96)*, volume 1146 of *Lecture Notes in Computer Science*, pages 325–350, Rome, Italy, September 1996. Springer-Verlag.

[16] R. C. Merkle. Secure communication over insecure channels. *Communications of the ACM*, 21(4):294–299, April 1978.

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

[18] B. C. Neuman and S. G. Stubblebine. A note on the use of timestamps as nonces. *Operating Systems Review*, 27(2):10–14, April 1993.

[19] B. C. Neuman and T. Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications Magazine*, 32(9):33–38, September 1994.

[20] B. Preneel and P. C. van Oorschot. MDx-MAC and building fast MACs from hash functions. In D. Coppersmith, editor, *Advances in Cryptology - CRYPTO'95*, volume 963 of *Lecture Notes in Computer Science*, pages 1–14, Santa Barbara, CA, August 1995. Springer-Verlag.

[21] M. K. Reiter and S. G. Stubblebine. Authentication metric analysis and design. *ACM Transactions on Information and System Security*, 2(2):138–158, May 1999.

[22] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[23] W. R. Stevens. *UNIX Network Programming: Networking APIs: Sockets and XTI*, volume 1. Prentice-Hall PTR, Upper Saddle River, NJ, second edition, 1998.

[24] P. R. Zimmermann. *The Official PGP User's Guide*. MIT Press, Cambridge, MA, 1995.