Chapter 15 – USER AUTHENTICATION

This chapter examines some of the authentication functions that have been developed to support network-based use authentication. In most computer security contexts, user authentication is the fundamental building block and the primary line of defense. RFC 2828 defines user authentication as the process of verifying an identity claimed by or for a system entity. An authentication process consists of two steps:

- **Identification step:** Presenting an identifier to the security system. (Identifiers should be assigned carefully, because authenticated identities are the basis for other security services, such as access control service.)

- **Verification step:** Presenting or generating authentication information that corroborates the binding between the entity and the identifier.”

In essence, identification is the means by which a user provides a claimed identity to the system; user authentication is the means of establishing the validity of the claim. Note that user authentication is distinct from message authentication.

There are four general means of authenticating a user's identity, which can be used alone or in combination:

- **Something the individual knows:** Examples includes a password, a personal identification number (PIN), or answers to a prearranged set of questions.

- **Something the individual possesses:** Examples include electronic keycards, smart cards, and physical keys. This type of authenticator is referred to as a *token*.

- **Something the individual is (static biometrics):** Examples include recognition by fingerprint, retina, and face.

- **Something the individual does (dynamic biometrics):** Examples include recognition by voice pattern, handwriting characteristics, and typing rhythm.
All of these methods, properly implemented and used, can provide secure user authentication. However, each method has problems. An adversary may be able to guess or steal a password. Similarly, an adversary may be able to forge or steal a token. A user may forget a password or lose a token. Further, there is a significant administrative overhead for managing password and token information on systems and securing such information on systems. With respect to biometric authenticators, there are a variety of problems, including dealing with false positives and false negatives, user acceptance, cost, and convenience.

### Authentication Protocols

An important application area is that of **mutual authentication** protocols. Such protocols enable communicating parties to satisfy themselves mutually about each other's identity and to exchange session keys. This topic was examined in Chapter 14. There, the focus was key distribution. Central to the problem of authenticated key exchange are two issues: **confidentiality and timeliness**. To prevent masquerade and to prevent compromise of session keys, essential identification and session key information must be communicated in encrypted form. The second issue, timeliness, is important because of the threat of message replays.

Replay Attacks are where a valid signed message is copied and later resent. Such replays, at worst, could allow an opponent to compromise a session key or successfully impersonate another party. At minimum, a successful replay can disrupt operations by presenting parties with messages that appear genuine but are not.

**Examples of replay attacks:**

**Simple replay:** The opponent simply copies a message and replays it later.

**Repetition that can be logged:** An opponent can replay a timestamped message within the valid time window.
**Repetition that cannot be detected:** This situation could arise because the original message could have been suppressed and thus did not arrive at its destination; only the replay message arrives.

**Backward replay without modification:** This is a replay back to the message sender. This attack is possible if symmetric encryption is used and the sender cannot easily recognize the difference between messages sent and messages received on the basis of content.

Possible countermeasures include the use of:

- **Sequence numbers** (generally impractical since must remember last number used with every communicating party)
- **Timestamps** (needs synchronized clocks amongst all parties involved, which can be problematic)
- **Challenge/response** (using unique, random, unpredictable nonce, but not suitable for connectionless applications because of handshake overhead)

**One-Way Authentication**

One application for which encryption is growing in popularity is electronic mail (e-mail). The very nature of electronic mail, and its chief benefit, is that it is not necessary for the sender and receiver to be online at the same time. Instead, the e-mail message is forwarded to the receiver’s electronic mailbox, where it is buffered until the receiver is available to read it. Accordingly, the e-mail message should be encrypted such that the mail-handling system is not in possession of the decryption key. A second requirement is that of authentication. Typically, the recipient wants some assurance that the message is from the alleged sender.
15.2 REMOTE USER AUTHENTICATION USING SYMMETRIC ENCRYPTION

Mutual Authentication

As discussed earlier, a two-level hierarchy of symmetric encryption keys can be used to provide confidentiality for communication in a distributed environment. Usually involves the use of a trusted key distribution center (KDC). Each party in the network shares a secret master key with the KDC.

The KDC is responsible for generating session keys, and for distributing those keys to the parties involved, using the master keys to protect these session keys.

Needham-Schroeder Protocol

The Needham-Schroeder Protocol is the original, basic key exchange protocol. Used by 2 parties who both trusted a common key server, it gives one party the info needed to establish a session key with the other.

Note that all communications is between A&KDC and A&B, B&KDC don't talk directly (though indirectly a message passes from KDC via A to B, encrypted in B's key so that A is unable to read or alter it). Other variations of key distribution protocols can involve direct communications between B&KDC.

1. A → KDC: $ID_A \ || \ ID_B \ || \ N_1$

2. KDC → A: $E(K_a, [K_s \ || \ ID_B \ || \ N_1 \ || \ E(K_b, [K_s \ || \ ID_A]))$)

3. A → B: $E(K_b, [K_s \ || \ ID_A])$

4. B → A: $E(K_s, [N_2])$

5. A → B: $E(K_s, [f(N_2)])$
Secret keys $K_a$ and $K_b$ are shared between A and the KDC and B and the KDC, respectively. The purpose of the protocol is to distribute securely a session key $K_s$ to A and B.

There is a critical flaw in the protocol, as shown. The message in step 3 can be decrypted, and hence understood only by B. But if an opponent, $X$, has been able to compromise an old session key, then $X$ can impersonate A and trick B into using the old key by simply replaying step 3. Admittedly, this is a much more unlikely occurrence than that an opponent has simply observed and recorded step 3.

Denning proposes to overcome this weakness by a modification to the Needham/Schroeder protocol that includes the addition of a timestamp to steps 2 and 3. Her proposal assumes that the master keys, $K_a$ and $K_b$ are secure, and it consists of the following steps.

1. $A \rightarrow KDC$: $ID_A || ID_B$

2. $KDC \rightarrow A$: $E(K_a, [K_s || ID_B || T || E(K_b, [K_s || ID_A || ID_B || T])])$

3. $A \rightarrow B$: $E(K_b, [K_s || ID_A || T])$

4. $B \rightarrow A$: $E(K_s, N1)$

5. $A \rightarrow B$: $E(K_s, f(N1))$

T is a timestamp that assures A and B that the session key has only just been generated. Thus, both A and B know that the key distribution is a fresh exchange.

The Denning protocol seems to provide an increased degree of security compared to the Needham/Schroeder protocol. However, a new concern is raised: namely, that this new scheme requires reliance on clocks that are synchronized throughout the network. It points out a risk involved. The risk is based on the fact that the distributed clocks can
become unsynchronized as a result of faults in the clocks or the synchronization mechanism.

The problem occurs when a sender’s clock is ahead of the intended recipient’s clock. In this case, an opponent can intercept a message from the sender and replay it later when the timestamp in the message becomes current at the recipient’s site. This replay could cause unexpected results. Gong refers to such attacks as suppress-replay attacks.

15.4 REMOTE USER AUTHENTICATION USING ASYMMETRIC ENCRYPTION

Mutual Authentication

In Chapter 14, we presented one approach to the use of public-key encryption for the purpose of session key distribution (Figure 14.8). This protocol assumes that each of the two parties is in possession of the current public key of the other. It may not be practical to require this assumption.

1. A → AS: $ID_A || ID_B$
2. AS → A: $E(PR_{as}, [ID_A || PU_a || T]) || E(PR_{as}, [ID_B || PU_b || T])$
3. A → B: $E(PR_{as}, [ID_A || PU_a || T]) || E(PR_{as}, [ID_B || PU_b || T]) || E(NU_b, E(PR_a, [K_s || T]))$

A protocol using timestamps is provided in that uses a central system, referred to as an authentication server (AS), because it is not actually responsible for secret key distribution. Rather, the AS provides public-key certificates. The session key is chosen and encrypted by A; hence, there is no risk of exposure by the AS. The timestamps protect against replays of compromised keys. See text for details. This protocol is compact but, as before, requires synchronization of clocks. Another approach, proposed by Woo and Lam, makes use of nonces.
Chapter 15 User Authentication protocols

Note the authors themselves spotted a flaw in it and submitted a revised version of the algorithm in. In both this example and the protocols described earlier, protocols that appeared secure were revised after additional analysis. These examples highlight the difficulty of getting things right in the area of authentication.

One-Way Authentication

We have already presented public-key encryption approaches that are suited to electronic mail, including the straightforward encryption of the entire message for confidentiality (Figure 12.1b), authentication (Figure 12.1c), or both (Figure 12.1d). These approaches require that either the sender know the recipient's public key (confidentiality) or the recipient know the sender's public key (authentication) or both (confidentiality plus authentication).

If confidentiality is the primary concern, then better to encrypt the message with a one-time secret key, and also encrypt this one-time key with B's public key. If authentication is the primary concern, then a digital signature may suffice (but could be replaced by an opponent). To counter such a scheme, both the message and signature can be encrypted with the recipient's public key. The latter two schemes require that B know A's public key and be convinced that it is timely. An effective way to provide this assurance is the digital certificate.
15.3 KERBEROS

Kerberos is an authentication service developed as part of Project Athena at MIT, and is one of the best known and most widely implemented trusted third party key distribution systems.

Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users. Unlike most other authentication schemes, Kerberos relies exclusively on symmetric encryption, making no use of public-key encryption. Two versions of Kerberos are in common use: v4 & v5.

In a more open environment, in which network connections to other machines are supported, an approach that requires the user to prove his or her identity for each service invoked, and also require that servers prove their identity to clients, is needed to protect user information and resources housed at the server. Kerberos supports this approach, and assumes a distributed client/server architecture that employs one or more Kerberos servers to provide an authentication service. The first published report on Kerberos [STEI88] listed the following requirements:

• **Secure**: A network eavesdropper should not be able to obtain the necessary information to impersonate a user.

• **Reliable**: For all services that rely on Kerberos for access control, lack of availability of the Kerberos service means lack of availability of the supported services. Hence, Kerberos should be highly reliable and should employ a distributed server architecture, with one system able to back up another.

• **Transparent**: Ideally, the user should not be aware that authentication is taking place, beyond the requirement to enter a password.

• **Scalable**: The system should be capable of supporting large numbers of clients and servers. This suggests a modular, distributed architecture.
To support these requirements, Kerberos is a trusted third-party authentication service that uses a protocol based on that proposed by Needham and Schroeder which was discussed earlier in this chapter.

**The Version 4 Authentication Dialogue**

- Kerberos V4 is a basic third-party authentication scheme.

- The core of Kerberos is the **Authentication server (AS)** and **Ticket Granting Servers (TGS)** – these are trusted by all users and servers and must be securely administered.

- The protocol includes a sequence of interactions between the client, AS, TGT and desired server. Version 4 of Kerberos makes use of DES, in a rather elaborate protocol, to provide the authentication service.

The heart of the first problem is the lifetime associated with the ticket-granting ticket. If this lifetime is very short (e.g., minutes), then the user will be repeatedly asked for a password. If the lifetime is long (e.g., hours), then an opponent has a greater opportunity for replay. Similarly, if an opponent captures a service-granting ticket and uses it before it expires, the opponent has access to the corresponding service.

The second problem is that there may be a requirement for servers to authenticate themselves to users.

First, consider the problem of captured ticket-granting tickets and the need to determine that the ticket presenter is the same as the client for whom the ticket was issued. An efficient way of doing this is to use a session encryption key to secure information. Table 15.1a shows the technique for distributing the session key.
Table 15.1 Summary of Kerberos Version 4 Message Exchanges

(a) Authentication Service Exchange to obtain ticket-granting ticket

(1) C → AS  \[ ID_c \| \; ID_{tgs} \| \; TS_1 \]
(2) AS → C  \[ E(K_c, [K_{c,tgs} \| \; ID_{tgs} \| \; TS_2 \| \; LifeTime_2 \| \; Ticket_{tgs}]) \]
\[ Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| \; ID_c \| \; AD_C \| \; ID_{tgs} \| \; TS_2 \| \; LifeTime_2]) \]

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

(3) C → TGS  \[ ID_v \| \; Ticket_{tgs} \| \; Authenticator_c \]
(4) TGS → C  \[ E(K_{c,tgs}, [K_{c,v} \| \; ID_v \| \; TS_4 \| \; Ticket_v]) \]
\[ Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \| \; ID_c \| \; AD_C \| \; ID_{tgs} \| \; TS_2 \| \; LifeTime_2]) \]
\[ Ticket_v = E(K_v, [K_{c,v} \| \; ID_c \| \; AD_C \| \; ID_v \| \; TS_4 \| \; LifeTime_4]) \]
\[ Authenticator_c = E(K_{c,tgs}, [ID_c \| \; AD_C \| \; TS_3]) \]

(c) Client/Server Authentication Exchange to obtain service

(5) C → V  \[ Ticket_v \| \; Authenticator_c \]
(6) V → C  \[ E(K_{c,v}, [TS_5 + 1]) \] (for mutual authentication)
\[ Ticket_v = E(K_v, [K_{c,v} \| \; ID_c \| \; AD_C \| \; ID_v \| \; TS_4 \| \; LifeTime_4]) \]
\[ Authenticator_c = E(K_{c,v}, [ID_c \| \; AD_C \| \; TS_5]) \]

Table 15.1a shows the technique for distributing the session key. As before, the client sends a message to the AS requesting access to the TGS. The AS responds with a message, encrypted with a key derived from the user’s password (Kc) that contains the ticket. The encrypted message also contains a copy of the session key, \( K_{c,tgs} \), where the subscripts indicate that this is a session key for C and TGS. Because this session key is inside the message encrypted with (Kc), only the user’s client can read it. The same session key is included in the ticket, which can be read only by the TGS. Thus, the session key has been securely delivered to both C and the TGS.
Chapter 15 User Authentication protocols

Note that several additional pieces of information have been added to this first phase of the dialogue. Message (1) includes a timestamp, so that the AS knows that the message is timely. Message (2) includes several elements of the ticket in a form accessible to C. This enables C to confirm that this ticket is for the TGS and to learn its expiration time. Note that the ticket does not prove anyone's identity but is a way to distribute keys securely. It is the authenticator that proves the client's identity.

C sends the TGS a message that includes the ticket plus the ID of the requested service (message (3) in Table 15.1b). In addition, C transmits an authenticator, which includes the ID and address of C’s user and a timestamp. Unlike the ticket, which is reusable, the authenticator is intended for use only once and has a very short lifetime. The TGS can decrypt the ticket with the key that it shares with the AS. This ticket indicates that user C has been provided with the session key Kc,tgs. In effect, the ticket says, “Anyone who uses Kc,tgs must be C.” The TGS uses the session key to decrypt the authenticator.

The reply from the TGS, in message (4), follows the form of message (2). C now has a reusable service-granting ticket for V. When C presents this ticket, as shown in message (5), it also sends an authenticator. The server can decrypt the ticket, recover the session key, and decrypt the authenticator. If mutual authentication is required, the server can reply as shown in message (6).

Finally, at the conclusion of this process, the client and server share a secret key. This key can be used to encrypt future messages between the two or to exchange a new random session key for that purpose.
### Chapter 15 User Authentication protocols

<table>
<thead>
<tr>
<th>Message (1)</th>
<th>Client requests ticket-granting ticket.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ID_C$</td>
<td>Tells AS identity of user from this client.</td>
</tr>
<tr>
<td>$ID_{tg}$</td>
<td>Tells AS that user requests access to TGS.</td>
</tr>
<tr>
<td>$TS_1$</td>
<td>Allows AS to verify that client’s clock is synchronized with that of AS.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Message (2)</th>
<th>AS returns ticket-granting ticket.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_c$</td>
<td>Encryption is based on user’s password, enabling AS and client to verify password, and protecting contents of message (2).</td>
</tr>
<tr>
<td>$K_{c,tg}$</td>
<td>Copy of session key accessible to client created by AS to permit secure exchange between client and TGS without requiring them to share a permanent key.</td>
</tr>
<tr>
<td>$ID_{tg}$</td>
<td>Confirms that this ticket is for the TGS.</td>
</tr>
<tr>
<td>$TS_2$</td>
<td>Informs client of time this ticket was issued.</td>
</tr>
<tr>
<td>$Lifetime_{tg}$</td>
<td>Informs client of the lifetime of this ticket.</td>
</tr>
<tr>
<td>$Ticket_{tg}$</td>
<td>Ticket to be used by client to access TGS.</td>
</tr>
</tbody>
</table>

(a) **Authentication Service Exchange**

<table>
<thead>
<tr>
<th>Message (3)</th>
<th>Client requests service-granting ticket.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ID_V$</td>
<td>Tells TGS that user requests access to server V.</td>
</tr>
<tr>
<td>$Ticket_{tg}$</td>
<td>Assures TGS that this user has been authenticated by AS.</td>
</tr>
<tr>
<td>$Authenticator_c$</td>
<td>Generated by client to validate ticket.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Message (4)</th>
<th>TGS returns service-granting ticket.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{c,tg}$</td>
<td>Key shared only by C and TGS protects contents of message (4).</td>
</tr>
<tr>
<td>$K_{c,v}$</td>
<td>Copy of session key accessible to client created by TGS to permit secure exchange between client and server without requiring them to share a permanent key.</td>
</tr>
<tr>
<td>$ID_V$</td>
<td>Confirms that this ticket is for server V.</td>
</tr>
<tr>
<td>$TS_4$</td>
<td>Informs client of time this ticket was issued.</td>
</tr>
<tr>
<td>$Ticket_V$</td>
<td>Ticket to be used by client to access server V.</td>
</tr>
<tr>
<td>$Ticket_{tg}$</td>
<td>Reusable so that user does not have to reenter password.</td>
</tr>
<tr>
<td>$K_{tg}$</td>
<td>Ticket is encrypted with key known only to AS and TGS, to prevent tampering.</td>
</tr>
</tbody>
</table>
Chapter 15 User Authentication protocols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{c, tgs}$</td>
<td>Copy of session key accessible to TGS used to decrypt authenticator, thereby authenticating ticket.</td>
</tr>
<tr>
<td>$ID_C$</td>
<td>Indicates the rightful owner of this ticket.</td>
</tr>
<tr>
<td>$AD_C$</td>
<td>Prevents use of ticket from workstation other than one that initially requested the ticket.</td>
</tr>
<tr>
<td>$ID_{tgs}$</td>
<td>Assures server that it has decrypted ticket properly.</td>
</tr>
<tr>
<td>$TS_2$</td>
<td>Informs TGS of time this ticket was issued.</td>
</tr>
<tr>
<td>$Lifetime_2$</td>
<td>Prevents replay after ticket has expired.</td>
</tr>
<tr>
<td>$Authenticator_c$</td>
<td>Assures TGS that the ticket presenter is the same as the client for whom the ticket was issued has very short lifetime to prevent replay.</td>
</tr>
</tbody>
</table>

$K_{c, tgs}$ | Authenticator is encrypted with key known only to client and TGS, to prevent tampering. |
$ID_C$ | Must match ID in ticket to authenticate ticket. |
$AD_C$ | Must match address in ticket to authenticate ticket. |
$TS_3$ | Informs TGS of time this authenticator was generated. |

(b) Ticket-Granting Service Exchange

<table>
<thead>
<tr>
<th>Message (5)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Ticket_V$</td>
<td>Assures server that this user has been authenticated by AS.</td>
</tr>
<tr>
<td>$Authenticator_c$</td>
<td>Generated by client to validate ticket.</td>
</tr>
</tbody>
</table>

Message (6) | Optional authentication of server to client. |
$K_{e, v}$ | Assures C that this message is from V. |
$TS_8 + 1$ | Assures C that this is not a replay of an old reply. |
$Ticket_v$ | Reusable so that client does not need to request a new ticket from TGS for each access to the same server. |
$K_v$ | Ticket is encrypted with key known only to TGS and server, to prevent tampering. |
$K_{e, v}$ | Copy of session key accessible to client; used to decrypt authenticator, thereby authenticating ticket. |
$ID_C$ | Indicates the rightful owner of this ticket. |
$AD_C$ | Prevents use of ticket from workstation other than one that initially requested the ticket. |
$ID_V$ | Assures server that it has decrypted ticket properly. |
A full-service Kerberos environment consisting of a Kerberos server, a number of clients, and a number of application servers requires the following:

1. The Kerberos server must have the user ID and hashed passwords of all participating users in its database. All users are registered with the Kerberos server.

2. The Kerberos server must share a secret key with each server. All servers are registered with the Kerberos server.

A full-service Kerberos environment consisting of a Kerberos server, a number of clients, and a number of application servers is referred to as a Kerberos realm. A Kerberos realm is a set of managed nodes that share the same Kerberos database, and are part of the same administrative domain. If have multiple realms, their Kerberos servers must share keys and trust each other.
DIFFERENCES BETWEEN VERSIONS 4 AND 5

Version 5 is intended to address the limitations of version 4 in two areas: environmental shortcomings and technical deficiencies.

Environmental shortcomings.

1. Encryption system dependence: Version 4 requires the use of DES. Export restriction on DES as well as doubts about the strength of DES were thus of concern. In version 5, ciphertext is tagged with an encryption-type identifier so that any encryption technique may be used.
2. Internet protocol dependence: Version 4 requires the use of Internet Protocol (IP) addresses. Other address types, such as the ISO network address, are not accommodated. Version 5 network addresses are tagged with type and length, allowing any network address type to be used.

3. Message byte ordering: In version 4, the sender of a message employs a byte ordering of its own choosing and tags the message to indicate least significant byte in lowest address or most significant byte in lowest address. This technique works but does not follow established conventions. In version 5, all message structures are defined using Abstract Syntax Notation One (ASN.1) and Basic Encoding Rules (BER), which provide an unambiguous byte ordering.

4. Ticket lifetime: Lifetime values in version 4 are encoded in an 8-bit quantity in units of five minutes. Thus, the maximum lifetime that can be expressed is $2^8 \times 5 = 1280$ minutes (a little over 21 hours). This may be inadequate for some applications. In version 5, tickets include an explicit start time and end time, allowing tickets with arbitrary lifetimes.

5. Authentication forwarding: Version 4 does not allow credentials issued to one client to be forwarded to some other host and used by some other client. This capability would enable a client to access a server and have that server access another server on behalf of the client. For example, a client issues a request to a print server that then accesses the client’s file from a file server, using the client’s credentials for access. Version 5 provides this capability.

6. Interrealm authentication: In version 4, interoperability among N realms requires on the order of $N^2$ Kerberos-to-Kerberos relationships, as described earlier. Version 5 supports a method that requires fewer relationships, as described shortly.
Technical deficiencies:

1. **Double encryption:** Note in Table 15.1 [messages (2) and (4)] that tickets provided to clients are encrypted twice - once with the secret key of the target server and then again with a secret key known to the client. The second encryption is not necessary and is computationally wasteful.

2. **PCBC encryption:** Encryption in version 4 makes use of a nonstandard mode of DES known as propagating cipher block chaining (PCBC). It has been demonstrated that this mode is vulnerable to an attack involving the interchange of ciphertext blocks. Version 5 provides explicit integrity mechanisms, a checksum or hash code is attached to the message prior to encryption using CBC.

3. **Session keys:** Each ticket includes a session key that is used by the client to encrypt the authenticator sent to the service associated with that ticket. In addition, the session key may subsequently be used by the client and the server to protect messages passed during that session. However, because the same ticket may be used repeatedly to gain service from a particular server, there is the risk that an opponent will replay messages from an old session to the client or the server. In version 5, it is possible for a client and server to negotiate a subsession key, which is to be used only for that one connection.

4. **Password attacks:** Both versions are vulnerable to a password attack. The message from the AS to the client includes material encrypted with a key based on the client’s password. An opponent can capture this message and attempt to decrypt it by trying various passwords. If the result of a test decryption is of the proper form, then the opponent has discovered the client’s password and may subsequently use it to gain authentication credentials from Kerberos. Version 5 does provide a mechanism known as preauthentication, which should make password attacks more difficult, but it does not prevent them.

*****