5. Authentication

Contents

• Introduction
• Password-based Authentication
• Address-based Authentication
• Cryptographic Authentication Protocols
• Eavesdropping and Server Database Reading
• Trusted Intermediaries
• Session Key Establishment
• Authorization
• Delegation
• Authentication of People

Introduction

Authentication = Process of reliably verifying the identity of
• a user,
• a computer,
• both computer and user.

Forms of authentication:
• password-based,
• address-based,
• cryptographic.

\{ + combinations \}
Main problems:

- Eavesdropping.
- Users cannot easily remember complex passwords. They need to change their password at regular intervals ⇒ remembering passwords becomes more difficult.
- Problem with some mobile-telephone-systems: access authorization by ⟨telephone number, password⟩ ⇒ insecure

How can a malefactor gain access to passwords?

- **On-Line Attack:**
  Simply try different passwords
  ⇒ countermeasure: limit the number of attempts allowed (similar to the mechanisms used by cash dispensers)
5. Authentication

'Off-Line' vs. 'On-Line' Password Guessing

- **Off-Line Attack:**
  Intruder captures a quantity $X$ that was calculated from a password
  
  “Advantage”:
  - massive computing power can be employed to find the password
  - no limitation in the number of attempts
  - password structures can be guessed

![Diagram](attachment:image.png)

- **Storing User Passwords**

  Where does a server store the information about passwords?
  From where does it get them?

  - **Individually:**
    each server maintains a password-database
  - **Central Authorization Node:**
    server retrieves information from a central node
  - **Authorization Facilitator Node:**
    server sends the information received from user to this node, which performs the authentication and informs the server about the outcome.
5. Authentication

Storing User Passwords

No matter where authentication information is stored, the database should *never store unencrypted passwords*.

**Alternatives:**
- Store hashes of passwords
- Encrypt the stored passwords
- Combination: Encrypt the database of hashed passwords

---

5. Authentication

Address-based Authentication

Most network addresses are hierarchical.

For example: IP (Internet Protocol)

<table>
<thead>
<tr>
<th>Country</th>
<th>Network</th>
<th>Subnetwork</th>
<th>Station</th>
</tr>
</thead>
</table>

Possible forms of access authentication:

- Maintain list of network addresses of “equivalent” machines, i.e., provide users who have access to machine $X$ with the same access rights for machine $Y$
  
  **Problem:** user must have identical account names on all systems.

- Extension (in order to avoid problem of identical account names):
  
  Store entry:
  
  $\langle$remote address, remote account name, local account name$\rangle$
Implementation e.g. in UNIX:

- `/etc/hosts.equiv` file contains list of computers that have identical user account assignments.
- `.rhosts` file in a user’s home directory contains a list of tuples `<computer, account>` that are granted access to this user’s account.
  
  (To prevent users from granting access to others a centrally managed proxy database may be installed instead.)

Address-based authentication protects against eavesdropping, but is subject to two other threats:

- If someone gains privileged access to a node he can access all users’ resources of this node. He can also get access to other machines if users from these machines have access to “his” node.

- If someone on the network impersonates network addresses he can access all network resources of all users who have accounts on any of these nodes.
  
  Impersonating a network address is very easy on broadcast LANs. In point-to-point networks this is more difficult as the response would be sent to the impersonated node.
5. Authentication

Network Address Impersonation

Sending messages:
- Impersonating a network address is easy.
- The effect is topology-dependent.

**Example:** **Token ring.** If *Adr. X* and *Adr. 1* are on the same ring and *Adr. X* transmits a packet using *Adr. 1*, then *Adr. 1* removes *Adr. X*’s packet and sends a 'duplicate address' error message.

**Example:** **Attentive router.** A router may observe that a packet was sent by the wrong station. (requires a trusted router).

**But:** On broadcast networks, as e.g. Ethernet, impersonation is easy.

---

Receiving messages:
- **On the same LAN:**
  - Eavesdropping messages is trivial on broadcast networks.
  - One only needs to watch for packets with the desired destination address.

- **Coupled LANs:**
  - Eavesdropping messages is possible if the intruder is connected to a LAN on the path between the source and destination of the messages.
  - He needs to check the data link address of the router.
  - **But:** a router receives many packets, all of which need to be checked but most of which will have to be discarded.
5. Authentication

Network Address Impersonation

- Coupled LANs and intruder not on the path:
  - More difficult: intruder could generate routing messages in order to re-route traffic.

- Source Routing:
  - Re-routing messages is easy in case of source routing.
  - Each packet contains a source route with intermediate destinations.

**Example:** Assume intruder 'Trudy' plans to send messages to destination 'Bob' impersonating 'Alice'.

Trudy uses the source route \((\text{Alice, Trudy, D})\).

Bob will reply using source route \((\text{D, Trudy, Alice})\).

\(\Rightarrow\) The return traffic will go to Trudy.

5. Authentication

Cryptographic Authentication Protocols

Cryptographic authentication is much more secure than password-based or address-based authentication.

**Idea:**

- Alice proves her identity to Bob by performing a cryptographic operation on a quantity Bob supplies.
- The cryptographic operation is based on Alice’s secret.

There is a difference between authenticating a human and authenticating a computer:

- A computer can store a high-quality secret and it can perform cryptographic operations in contrast to (most) humans.
5. Authentication

Cryptographic Authentication Protocols

The computer can do cryptographic operations on behalf of its user. The user only has to remember a password.

System has to obtain a cryptographic key based on the password by:

- doing a hash of the password,
- using the password to decrypt a higher-quality key (e.g. RSA private key).

Passwords as cryptographic keys:

- very large numbers as cryptographic key ⇒ difficult to remember
- computer can convert a simple text string into a cryptographic key (for instance: DES, RSA)

Eavesdropping and Server Database Reading

Public key technology makes authentication secure from

- eavesdropping,
- an intruder reading the server database.

Idea: Using her private key Alice performs a cryptographic operation on a value supplied by Bob.
5. Authentication
Eavesdropping and Server Database Reading

Without public key cryptography:
- Protection against both eavesdropping and server database reading is difficult with one single protocol.
- It is easy to do one or the other.

Protocol 1:

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows hash of Alice's password</td>
<td></td>
</tr>
</tbody>
</table>

Bob computes hash of fiddlesticks and compares it with stored value.

Secure from server database reading but not from eavesdropping.

Protocol 2:

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows Alice's secret</td>
<td></td>
</tr>
<tr>
<td>Picks random R</td>
<td></td>
</tr>
</tbody>
</table>

Bob computes same function and compares it to X.

Secure from eavesdropping, but not from server database reading.
Assume that network security is based on secret key technology.

Consider a large network with \( n \) nodes.

Each computer may need to authenticate each other computer,
\( \Rightarrow \) each computer needs to know \( n-1 \) keys.

Adding a new node would cause generation of \( n \) keys, as that new node needs to have a shared secret with each other node.

The keys would have to be securely distributed to all the other nodes.
\( \Rightarrow \) clearly unworkable.

Alternatives:
- Key Distribution Center (KDC)
- Certification Authorities (CAs)
- Multiple Trusted Intermediaries
Disadvantages of KDCs:

- KDC has enough information to impersonate all nodes and users. If it is compromised, all network resources will be vulnerable.
- KDC is a single point of failure. If it goes down, nobody can use anything on the network.
- KDC might be a performance bottleneck, since everyone will frequently have to communicate with it.

Certification Authorities (CAs)

- Key distribution is easier with public key cryptography.
- Each node knows its own private key, and the public keys can be obtained from a central entity.

**Problem:** How to be sure that the public key information is correct?

**Solution:** Establish a trusted node, a *Certification Authority* (CA), to generate *certificates*.
- Certificates consist of a public key, a name (Alice) and a signature of a CA.
- CAs are the public key equivalent of KDCs.
The advantages of CAs over KDCs include:

- The CA does not need to be on-line. Key exchange may be done through e.g. floppy disks or smart cards.

- Since the CA does not necessarily have to be on-line or process communication protocols, a comparably simple device can be employed ⇒ improved security.

- If the CA crashes the network will still be usable. Only the installation of new user will be impossible.

- Certificates are not security-sensitive. One cannot write bogus certificates as only the CA generate signatures.

- A compromised CA cannot decrypt conversations.

Disadvantage of CAs: once a certificate has been issued it is difficult to revoke it if the CA is not online.

Typically, a certificate is valid for one year.

**Problem:** what happens if a certificate has to be revoked?

**Solution:** (similar to credit cards)

- Publish a list of all revoked certificates ⇒ *Certificate Revocation List* (CRL).

- The CRLs will be distributed periodically or after a new version to all computers.

- Certificate is valid if it has a valid CA signature and is not listed on the CRL.
Problem with both KDCs and CAs:
- there is only one single entity trusted by all stakeholders of the system,
- it might be difficult to establish a single trusted entity at an international scale.
- **Solution**: subdivide the world into *domains* with a trusted entity (KDC or CA) for each such domain.
- Authentication across different domains is possible but more complicated

### Example: Alice is in the CIA, Boris is in the KGB. CIA and KGB each manage a KDC and share a key: $K_{KGB-CIA}$.

- Let me talk to Boris
  - $K_{new}$
  - $K_{Alice-Boris}$
  - Boris: talk to Alice from CIA; use $K_{Alice-Boris}$

- Let me talk to KGB’s KDC
  - $K_{new}$
  - $K_{CIA-KGB}$ (Alice from my domain wants to talk to you; use $K_{new}$)
  - CIA’s KDC generates $K_{new}$
  - $K_{CIA-KGB}$ (Alice from my domain wants to talk to you; use $K_{new}$)

- Let me talk to KGB’s KDC
  - $K_{new}$
  - $K_{Alice-Boris}$
  - Boris: talk to Alice from CIA; use $K_{Alice-Boris}$
5. Authentication
Multiple KDC Domains

- A global network comprises of thousands of domains.
- Most simple authentication method: each pair of KDCs share a key ⇒ very complicated.
- Other methods: tree of KDCs or less structured logical interconnection of KDCs

\[ \text{KDC}_1 \rightarrow \text{KDC}_2 \rightarrow \text{KDC}_3 \]
\[ \text{KDC}_4 \rightarrow \text{KDC}_5 \]
\[ \text{KDC}_6 \]

5. Authentication
Multiple CA Domains

Situation is similar with CAs: multiple CA domains.
Alice obtains a certificate, signed by her own CA, stating the public key of Boris’ CA.
Using that certificate plus Boris’ certificate she can verify Boris’ public key.
- Alice obtains Boris’ CA certificate stating that its public key is \( P_1 \), signed by her own CA.
- Alice obtains Boris’ certificate stating that his public key is \( P_2 \), signed with key \( P_1 \).
Because she has both certificates she now can verify Boris’s public key.

Similar to KDCs: there will be a chain of CAs.
Session Key Establishment

- With cryptographic authentication computer networks would be more secure.
- But there are security vulnerabilities that occur after authentication.
- Protection against eavesdropping, session hijackers, and message manglers can be done by using cryptography throughout the whole conversation.
- Using public keys to encrypt all data is computationally expensive.
- It is more practical to have principals agree on a secret key, and to protect the remainder of the conversation with this key.
- The authentication protocol should help the principals agree on a secret key.

Suppose Alice and Bob are using a shared secret key for authentication. They could continue to use that key to protect their conversation. However, it is a good idea to generate a separate session key:

- Keys “wear out” if used a lot. The more encrypted data an intruder has the better his chances of finding the key.
  \[ \Rightarrow \text{use a secret per-session key generated at the time of authentication} \]
- It might be possible for an intruder to record messages from a previous conversation and inject those packets into a current conversation.

Using sequence numbers per session might not help to distinguish packets transmitted a month ago from today's packets.
5. Authentication
Session Key Establishment

- If the long-term shared secret key were compromised, it would be desirable to prevent an old recorded conversation from being decrypted.
- You may want to establish a session and give a relatively untrusted piece of software the session key, which is good only for that conversation.
  
  Giving it your long-term secret key would be insecure, since the untrusted software could store it for future misuse.

Therefore, authentication protocols usually establish a session key in addition to providing authentication.

5. Authentication
Authorization

**Authentication** establishes who you are.

**Authorization** establishes what you are allowed to do.

Method for authorization:

- An **ACL (access control list)** associated with each resource, which lists every individual allowed to use that resource.
  
  Maintaining ACLs on every file can quickly become prohibitively expensive if the user group is very large.

- A common solution is to introduce the concept of **groups**. Instead of specifying all the individuals on every file, establish an entry in the ACL for separate group-indication.
  
  Problem: efforts may duplicated if different groups exist with identical membership.
5. Authentication
Authorization

There are two possible methods of eliminating this duplication effort:

- Store the group-list at a central server \( \Rightarrow \) bottleneck.
- Include information about to which groups an individual belongs into that individual’s certificate (in case of public key based schemes) or into his ticket (in the case of a secret key KDC based scheme). \( \Rightarrow \) an individual’s certificate may become very large.

Hierarchical groups:

- It is more flexible, convenient, and scalable if recursive group membership is possible.
- Management becomes easier.
- Problem: groups will be administrated on different servers. Authorization depends on server availability.

5. Authentication
Delegation

Sometimes it may be necessary for a user to grant permission to act on his behalf to a computer, printer, software, or another user. This permission is known as **delegation** or **authentication forwarding**.

Delegation-method: generate a special message, signed by the authorized user, specifying:

- to whom he is delegating rights,
- which rights are being delegated,
- and for how long.

After the duration has expired, the message no longer grants any permission.
Using Public key cryptography: Sign the delegation message with your private key.

Using KDCs: Ask the KDC to give user Bob certain permissions.

The KDC:
- constructs a message listing the permissions you would like to grant Bob,
- encrypts the message with a key known only to the KDC,
- and gives Bob the encrypted message.

Bob cannot decrypt the message, but he can present it to the KDC when asking for access.

---

Password-related issues:
- how to bring users to choosing unguessable passwords,
- how to store password information securely at a system,
- how to avoid divulging information to eavesdroppers.

Authentication depends on the capability
- to store a high-quality cryptographic key,
- to perform cryptographic operations.

A computer has both capabilities; a person has neither.
User authentication consists of a computer verifying that you are who you claim to be.

There are three main techniques:

- what you know,
- what you have,
- what you are.

Passwords fit into the *what you know* category.
Physical keys fit into the *what you have* category.
Biometric devices, such as voice recognition systems or fingerprint analyzers, fit into the *what you are* category.

**Passwords** are a classical technique (originating from the military domain).

People log into a computer by typing their user name and passwords.

Problems with using passwords for authentication:

- Eavesdroppers might see the password when users log in.
- An intruder might read the file with the password information.
- The password might be easy to guess (on-line).
- The password may be cracked by an off-line attack.
- Attempts to force users to choose unguessable passwords, might render the system so inconvenient that it becomes unusable, or that users resort to writing passwords down.
5. Authentication

On-Line Password Guessing

- An intruder can impersonate a user if he can guess his password.
- Many people choose e.g. a names or a birthday date as their password. This makes passwords easy to remember, but it also makes them easy to guess.
- Even more complex may be guessable if the intruder is allowed many guesses.
- A solution is to limit the number of permitted guesses (cash dispenser)
  Problem: A computer vandal can, possibly with the aid of a computer, guess five bad passwords for all accounts, thus locking them all.
- Another approach is to allow a limited number of password guesses per connect attempt.

Off-Line Password Guessing

- The passwords stored on a system must be protected. One approach is to store a hash of the password (as e.g. under UNIX).
- An intruder can obtain a cryptographic hash of the password through either eavesdropping or reading a database.
- He can guess a password performing the same hash and comparing it with the stolen quantity (e.g., the ‘Dictionary’ attack).
- A approach to slow down an attacker:
  - When setting a new password the system chooses a random number (salt).
  - The system stores the salt and a hash of the combination of the stored salt and the supplied password.

<table>
<thead>
<tr>
<th>userID</th>
<th>salt value</th>
<th>password hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>2758</td>
<td>hash(2758</td>
</tr>
</tbody>
</table>
5. Authentication

Passwords and Careless Users

- One big problem with password authentication is users handling them very carelessly.
- People write down passwords, store them in unprotected files or send them by e-mail.
- Users often choose the same password for several systems.

The solution is to educate users on the importance of security.
- A technique that offers some security, but which is probably overused, is to require frequent mandatory password changes.
- In general, it is impossible to make systems secure without the cooperation of the legitimate users.

A Login Trojan Horse to Capture Passwords

- An old approach to capture passwords is to leave a Trojan horse on a public terminal.
- This program displays a login prompt. Unsuspecting users enter their user name and password.
- The Trojan horse logs the names and passwords to a file.
- Approaches toward minimizing this threat are based on making it difficult for a program to look like a normal login prompt:
  - Control the original prompt.
  - Check the bit-mapped screen (log-in screens take up the entire screen and have no border).
  - On most systems there is some way to interrupt running programs. Users could be trained to use the interrupt key.
Authentication Tokens

An authentication token is an physical device that a person carries and uses to authenticate.
Unless they are physically attached to the user they must be coupled with one of the other two mechanisms (what you know, what you are) to be secure, since they are subject to theft.

Authentication tokens include:
- keys (e.g., for homes and cars),
- credit cards.

Credit cards have a magnetic strip that contains information about a large secret.
One big advantage is that people tend to be less willing to ‘loan’ a token to someone than to share a password.

Disadvantages of authentication tokens:
- Use of these tokens requires custom hardware (a key slot or card reader) on every access device. This may be expensive and it requires standardisation.
- Tokens can be lost or stolen. For reasonable security, tokens must be supplemented with a PIN or password.
- These devices offer little or no protection against eavesdropping.
A better form of authentication token is the **smart card**. This is a device about the size of a credit card but with an embedded CPU and memory. When inserted into a **smart card reader**, the card can communicate with the device.

Forms of smart cards:

- **PIN protected memory card.** This card gives access to information upon entering a PIN. After a number of invalid PINs the card locks itself.
- **Cryptographic challenge/response cards.** This card has a cryptographic key in memory. The card can encrypt or decrypt data using this key, but will not reveal the key even if the correct PIN has been entered.

- **Cryptographic calculator.** Like a cryptographic card but it requires no electrical connection to the terminal. It has a display and a keyboard, and all interconnection is through the user. Similar to smart cards this device can perform cryptographic operations using a key. It can emulate a smart card.

The biggest advantage of these readerless smart cards is that they can be used from normal terminals with no need for special hardware.
Biometric authentication devices measure the physical characteristics of a user and match them against a profile.

**Advantage:** You can’t ‘loan’ something that might help someone fool a biometric authentication device; nor would it make sense to steal the device.

Technology available today include:

- **Retinal scanner.** Device that examines the tiny blood vessels in the back of your eye.
- **Fingerprint readers.**
- **Handprint readers.** Device that measures the dimensions of the hand.
- **Voiceprints.** Frequency spectrum analysis of the voice.
- **Keystroke timing.** The exact way in which people type is quite distinctive.
- **Signatures.** This is a classic human form of authentication.