Unit IV: User Authentication

Contents:

4.1 User Authentication Principles

In most computer security contexts, user authentication is the fundamental building block and the primary line of defense. User authentication is the basis for most types of access control and for user accountability.
User authentication encompasses two functions. First, the user identifies herself to the system by presenting a credential, such as user ID. Second, the system verifies the user by the exchange of authentication information.

NIST SP 800-63-3 (Digital Authentication Guideline, October 2016) defines digital user authentication as the process of establishing confidence in user identities that are presented electronically to an information system. Systems can use the authenticated identity to determine if the authenticated individual is authorized to perform particular functions, such as database transactions or access to system resources. In many cases, the authentication and transaction, or other authorized function, take place across an open network such as the Internet. Equally authentication and subsequent authorization can take place locally, such as across a local area network.

List of security requirements for identification and authentication services:

Basic Security Requirements:

1. Identify information system users, processes acting on behalf of users, or devices.
2. Authenticate (or verify) the identities of those users, processes, or devices, as a prerequisite to allowing access to organizational information systems.

Derived Security Requirements:

3. Use multifactor authentication for local and network access to privileged accounts and for network access to non-privileged accounts.
4. Employ replay-resistant authentication mechanisms for network access to privileged and non-privileged accounts.
5. Prevent reuse of identifiers for a defined period.
6. Disable identifiers after a defined period of inactivity.
7. Enforce a minimum password complexity and change of characters when new passwords are created.
8. Prohibit password reuse for a specified number of generations.
9. Allow temporary password use for system logons with an immediate change to a permanent password.
10. Store and transmit only cryptographically-protected passwords.
11. Obscure feedback of authentication information.

Means of 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 include 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. Multifactor authentication refers to the use of more than one of the authentication means in the preceding list. The strength of authentication systems is largely determined by the number of factors incorporated by the system. Implementations that use two factors are considered to be stronger than those that use only one factor; systems that incorporate three factors are stronger than systems that only incorporate two of the factors, and so on.

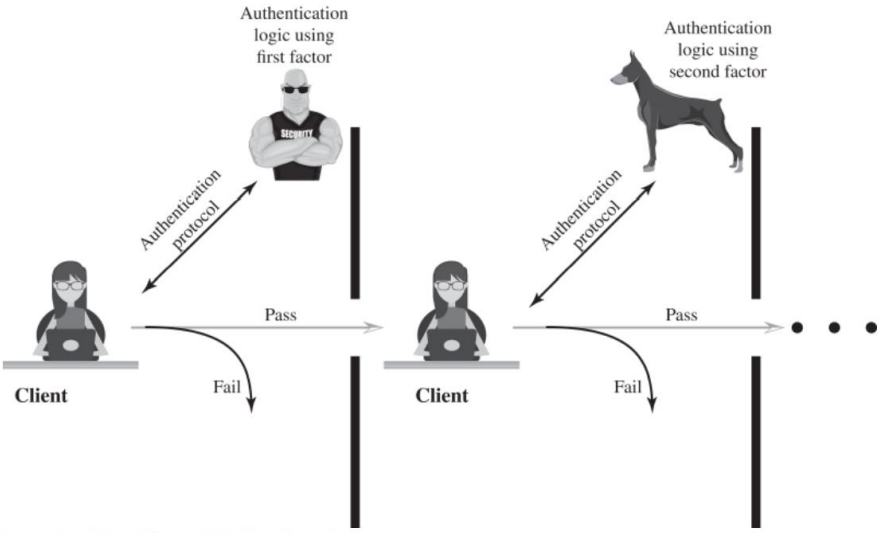


Figure 3.2 Multifactor Authentication

4.2 Password-Based Authentication

A widely used line of defense against intruders is the password system. Virtually all multiuser systems, network-based servers, Web-based e-commerce sites, and other similar services require that a user provide not only a name or identifier (ID) but also a password. The system compares the password to a previously stored password for that user ID, maintained in a system password file. The password serves to authenticate the ID of the individual logging on to the system. In turn, the ID provides security in the following ways:

- The ID determines whether the user is authorized to gain access to a system. In some systems, only those who already have an ID filed on the system are allowed to gain access.
- The ID determines the privileges accorded to the user. A few users may have administrator or "superuser" status that enables them to read files and perform functions that are especially protected by the operating system. Some systems have guest or anonymous accounts, and users of these accounts have more limited privileges than others.
- The ID is used in what is referred to as discretionary access control. For example, by listing the IDs of the other users, a user may grant permission to them to read files owned by that user.

The Vulnerability of Passwords

Typically, a system that uses password-based authentication maintains a password file indexed by user ID. One technique that is typically used is to store not the user's password but a one-way hash function of the password.
The main forms of attack against password-based authentication and its countermeasure strategy are:

- Offline dictionary attack: Typically, strong access controls are used to protect the system's password file. However, experience shows that determined hackers can frequently bypass such controls and gain access to the file. The attacker obtains the system password file and compares the password hashes against hashes of commonly used passwords. If a match is found, the attacker can gain access by that ID/password combination.
  Countermeasures include controls to prevent unauthorized access to the password file, intrusion detection measures to identify a compromise, and rapid reissuance of passwords should the password file be compromised.

- Specific account attack: The attacker targets a specific account and submits password guesses until the correct password is discovered.
  The standard countermeasure is an account lockout mechanism, which locks out access to the account after a number of failed login attempts. Typical practice is no more than five access attempts.

- Popular password attack: A variation of the preceding attack is to use a popular password and try it against a wide range of user IDs. A user's tendency is to choose a password that is easily remembered; this unfortunately makes the password easy to guess.
  Countermeasures include policies to inhibit the selection by users of common passwords and scanning the IP addresses of authentication requests and client cookies for submission patterns.

- Password guessing against single user: The attacker attempts to gain knowledge about the account holder and system password policies and uses that knowledge to guess the password.
  Countermeasures include training in and enforcement of password policies that make passwords difficult to guess. Such policies address the secrecy, minimum length of the password, character set, prohibition against using well-known user identifiers, and length of time before the password must be changed.

- Workstation hijacking: The attacker waits until a logged-in workstation is unattended.
  The standard countermeasure is automatically logging the workstation out after a period of inactivity. Intrusion detection schemes can be used to detect changes in user behavior.

- Exploiting user mistakes: If the system assigns a password, then the user is more likely to write it down because it is difficult to remember. This situation creates the potential for an adversary to read the written password. A user may intentionally share a password, to enable a colleague to share files, for example. Also, attackers are frequently successful in obtaining passwords by using social engineering tactics that trick the user or an account manager into revealing a password. Many computer systems are shipped with preconfigured passwords for system administrators. Unless these preconfigured passwords are changed, they are easily guessed.
  Countermeasures include user training, intrusion detection, and simpler passwords combined with another authentication mechanism.

- Exploiting multiple password use: Attacks can also become much more effective or damaging if different network devices share the same or a similar password for a given user.
  Countermeasures include a policy that forbids the same or similar password on particular network devices.

- Electronic monitoring: If a password is communicated across a network to log on to a remote system, it is vulnerable to eavesdropping. Simple encryption will not fix this problem, because the encrypted password is, in effect, the password and can be observed and reused by an adversary.

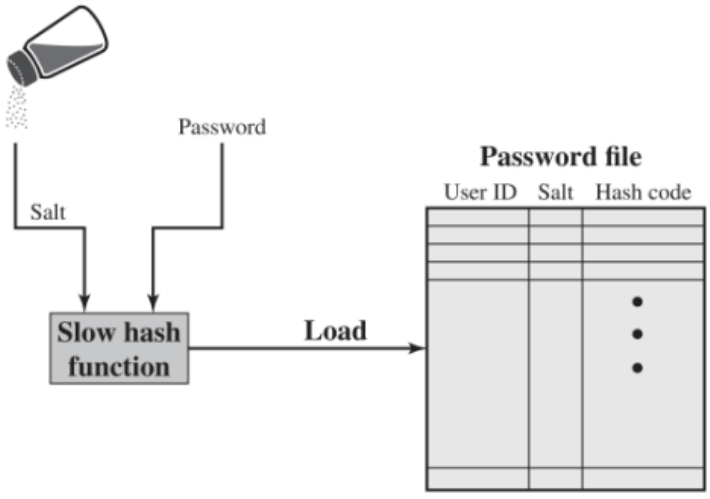
Despite the many security vulnerabilities of passwords, they remain the most commonly used user authentication technique, and this is unlikely to change in the foreseeable future. Among the reasons for the persistent popularity of passwords are the following:

1. Techniques that utilize client-side hardware, such as fingerprint scanners and smart card readers, require the implementation of the appropriate user authentication software to exploit this hardware on both the client and server systems. Until there is widespread acceptance on one side, there is reluctance to implement on the other side, so we end up with a who-goes-first stalemate.
2. Physical tokens, such as smart cards, are expensive and/or inconvenient to carry around, especially if multiple tokens are needed.
3. Schemes that rely on a single sign-on to multiple services, using one of the non-password techniques described in this chapter, create a single point of security risk.
4. Automated password managers that relieve users of the burden of knowing and entering passwords have poor support for roaming and synchronization across multiple client platforms, and their usability had not be adequately researched.

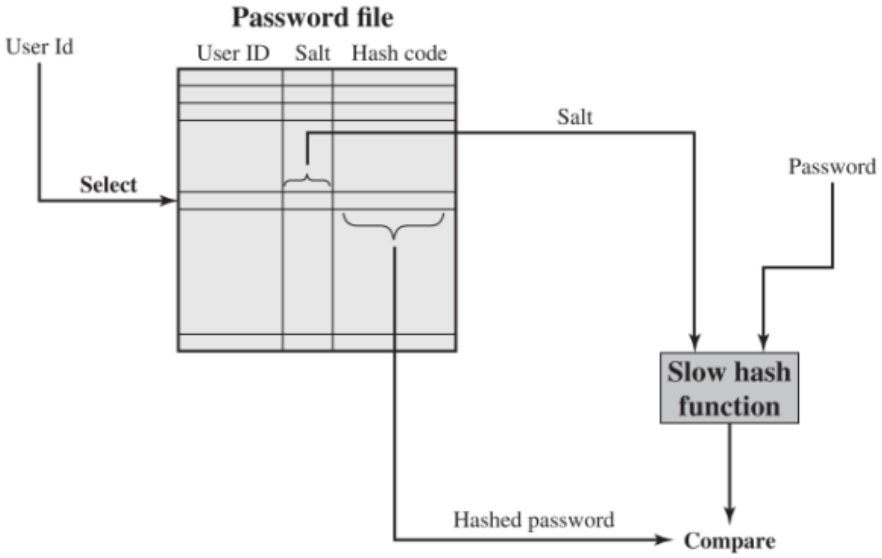
The Use of Hashed Passwords

A widely used password security technique is the use of hashed passwords and a salt value. This scheme is found on virtually all UNIX variants as well as on a number of other operating systems.
To load a new password into the system, the user selects or is assigned a password. This password is combined with a fixed-length salt value. In older implementations, this value is related to the time at which the password is assigned to the user. Newer implementations use a pseudorandom or random number. The password and salt serve as inputs to a hashing algorithm to produce a fixed-length hash code. The hash algorithm is designed to be slow to execute in order to thwart attacks. The hashed password is then stored, together with a plaintext copy of the salt, in the password file for the corresponding user ID. The hashed password method has been shown to be secure against a variety of cryptanalytic attacks.
When a user attempts to log on to a UNIX system, the user provides an ID and a password. The operating system uses the ID to index into the password file and retrieve the plaintext salt and the encrypted

password. The salt and user-supplied password are used as input to the encryption routine. If the result matches the stored value, the password is accepted.



(a) Loading a new password



(b) Verifying a password

The salt serves three purposes:

- It prevents duplicate passwords from being visible in the password file. Even if two users choose the same password, those passwords will be assigned different salt values. Hence, the hashed passwords of the two users will differ.
- It greatly increases the difficulty of offline dictionary attacks. For a salt of length b bits, the number of possible passwords is increased by a factor of increasing the difficulty of guessing a password in a dictionary attack.
- It becomes nearly impossible to find out whether a person with passwords on two or more systems has used the same password on all of them.

Password Cracking of User-Chosen Passwords

Traditional Approaches

The traditional approach to password guessing, or password cracking as it is called, is to develop a large dictionary of possible passwords and to try each of these against the password file. This means that each password must be hashed using each available salt value then compared with stored hash values. If no match is found, the cracking program tries variations on all the words in its dictionary of likely passwords. Such variations include backward spelling of words, additional numbers or special characters, or sequence of characters.

An alternative is to trade off space for time by precomputing potential hash values. In this approach the attacker generates a large dictionary of possible passwords. For each password, the attacker generates the hash values associated with each possible salt value. The result is a mammoth table of hash values known as a rainbow table. Results showed that using 1.4 GB of data, could be cracked 99.9% of all alphanumeric Windows password hashes in 13.8 seconds. This approach can be countered using a sufficiently large salt value and a sufficiently large hash length.

To counter the use of large salt values and hash lengths, password crackers exploit the fact that some people choose easily guessable passwords. A particular problem is that users, when permitted to choose their own password, tend to choose short ones.

Password length is only part of the problem. Many people, when permitted to choose their own password, pick a password that is guessable, such as their own name, their street name, a common dictionary word, and so forth. This makes the job of password cracking straightforward. The cracker simply has to test the password file against lists of likely passwords. Because many people use guessable passwords, such a strategy should succeed on virtually all systems.

Attacks that use a combination of brute-force and dictionary techniques have become common. A notable example of this dual approach is John the Ripper, an open-source password cracker first developed in 1996, and still in use.

Modern Approaches

At present users are doing a better job of selecting passwords, and organizations are doing a better job of forcing users to pick stronger passwords, a concept known as a complex password policy.
However, password-cracking techniques have improved to keep pace. The improvements are of two kinds.
First, the processing capacity available for password cracking has increased dramatically. Now used increasingly for computing, graphics processors allow password-cracking programs to work thousands of times faster than they did just a decade ago on similarly priced PCs that used traditional CPUs alone. Only a decade ago, such speeds were possible only when using pricey supercomputers.
The second area of improvement in password cracking is in the use of sophisticated algorithms to generate potential passwords.
To develop techniques that are more efficient and effective than simple dictionary and brute-force attacks, researchers and hackers have studied the structure of passwords. To do this, analysts need a large pool of real-word passwords to study, which they now have. The first big breakthrough came in late 2009, when an SQL injection attack against online games service RockYou.com exposed 32 million plaintext passwords used by its members to log in to their accounts.

Password File Access Control

One way to prevent a password attack is to deny the opponent access to the password file. If the hashed password portion of the file is accessible only by a privileged user, then the opponent cannot read it without already knowing the password of a privileged user. Often, the hashed passwords are kept in a separate file from the user IDs, referred to as a shadow password file. Special attention is paid to making the shadow password file protected from unauthorized access. Although password file protection is certainly worthwhile, there remain vulnerabilities:

- Many systems, including most UNIX systems, are susceptible to unanticipated break-ins. A hacker may be able to exploit a software vulnerability in the operating system to bypass the access control system long enough to extract the password file. Alternatively, the hacker may find a weakness in the file system or database management system that allows access to the file.
- An accident of protection might render the password file readable, thus compromising all the accounts.
- Some of the users have accounts on other machines in other protection domains, and they use the same password. Thus, if the passwords could be read by anyone on one machine, a machine in another location might be compromised.
- A lack of, or weakness in, physical security may provide opportunities for a hacker. Sometimes, there is a backup to the password file on an emergency repair disk or archival disk. Access to this backup enables the attacker to read the password file. Alternatively, a user may boot from a disk running another operating system such as Linux and access the file from this OS.
- Instead of capturing the system password file, another approach to collecting user IDs and passwords is through sniffing network traffic.

Thus, a password protection policy must complement access control measures with techniques to force users to select passwords that are difficult to guess.

Password Selection Strategies

When not constrained, many users choose a password that is too short or too easy to guess. At the other extreme, if users are assigned passwords consisting of eight randomly selected printable characters, password cracking is effectively impossible. But it would be almost as impossible for most users to remember their passwords. Fortunately, even if we limit the password universe to strings of characters that are reasonably memorable, the size of the universe is still too large to permit practical cracking. Our goal, then, is to eliminate guessable passwords while allowing the user to select a password that is memorable. Four basic techniques are in use:

- User education
- Computer-generated passwords
- Reactive password checking
- Complex password policy

User education: Users can be told the importance of using hard-to-guess passwords and can be provided with guidelines for selecting strong passwords. This user education strategy is unlikely to succeed at most installations, particularly where there is a large user population or a lot of turnover. Many users will simply ignore the guidelines. Others may not be good judges of what is a strong password. For example, many users (mistakenly) believe that reversing a word or capitalizing the last letter makes a password unguessable.

Nonetheless, it makes sense to provide users with guidelines on the selection of passwords. Perhaps the best approach is the following advice: A good technique for choosing a password is to use the first letter of each word of a phrase. However, do not pick a well-known phrase like “An apple a day keeps the doctor

away" (Aaadktda). Instead, pick something like "My dog's first name is Rex" (MdfniR) or "My sister Peg is 24 years old" (MsPi24yo). Studies have shown users can generally remember such passwords, but they are not susceptible to password guessing attacks based on commonly used passwords.

Computer-generated passwords also have problems. If the passwords are quite random in nature, users will not be able to remember them. Even if the password is pronounceable, the user may have difficulty remembering it and so be tempted to write it down. In general, computer generated password schemes have a history of poor acceptance by users. A random number generator produces a random stream of characters used to construct the password.

A reactive password checking strategy is one in which the system periodically runs its own password cracker to find guessable passwords. The system cancels any passwords that are guessed and notifies the user. This tactic has a number of drawbacks. First, it is resource intensive if the job is done right. Because a determined opponent who is able to steal a password file can devote full CPU time to the task for hours or even days, an effective reactive password checker is at a distinct disadvantage. Furthermore, any existing passwords remain vulnerable until the reactive password checker finds them.

Complex password policy: A promising approach to improved password security is a complex password policy, or proactive password checker. In this scheme, a user is allowed to select his or her own password. However, at the time of selection, the system checks to see if the password is allowable and, if not, rejects it. Such checkers are based on the philosophy that, with sufficient guidance from the system, users can select memorable passwords from a fairly large password space that are not likely to be guessed in a dictionary attack. The trick with a proactive password checker is to strike a balance between user acceptability and strength. If the system rejects too many passwords, users will complain that it is too hard to select a password. If the system uses some simple algorithm to define what is acceptable, this provides guidance to password crackers to refine their guessing technique.

Rule Enforcement: The first approach is a simple system for rule enforcement. For example, NIST SP 800-63-2 suggests the following alternative rules:

- Password must have at least sixteen characters (basic16).
- Password must have at least eight characters including an uppercase and lowercase letter, a symbol, and a digit. It may not contain a dictionary word (comprehensive8).

Although this approach is superior to simply educating users, it may not be sufficient to prevent password crackers. This scheme alerts crackers as to which passwords not to try, but may still make it possible to do password cracking. The process of rule enforcement can be automated by using a proactive password checker, such as the openware pam_passwdqc (openwall.com/passwdqc/), which enforces a variety of rules on passwords and is configurable by the system administrator.

Password Checker: Another possible procedure is simply to compile a large dictionary of possible "bad" passwords. When a user selects a password, the system checks to make sure that it is not on the disapproved list. There are two problems with this approach:

- Space: The dictionary must be very large to be effective.
- Time: The time required to search a large dictionary may itself be large. In addition, to check for likely permutations of dictionary words, either those words must be included in the dictionary, making it truly huge, or each search must also involve considerable processing.

4.3 Token-Based Authentication

Objects that a user possesses for the purpose of user authentication are called tokens. Usually cards that have the appearance and size of bank cards are used as tokens.
Some of the cards used as tokens are:

Card Type	Defining Feature	Example
Embossed	Raised characters only, on front	Old credit card
Magnetic stripe	Magnetic bar on back, characters on front	Bank card
Memory	Electronic memory inside	Prepaid phone card
Smart Contact Contactless	Electronic memory and processor inside Electronic contacts exposed on surface Radio antenna embedded inside	Biometric ID card

Memory card
Memory cards can store but not process data. The most common such card is the bank card with a magnetic stripe on the back. A magnetic stripe can store only a simple security code, which can be read (and unfortunately reprogrammed) by an inexpensive card reader. There are also memory cards that include an internal electronic memory. Memory cards can be used alone for physical access, such as a hotel room. For authentication, a user provides both the memory card and some form of password or personal identification number (PIN). A typical application is an automatic teller machine (ATM). The memory card, when combined with a PIN or password, provides significantly greater security than a password alone. An adversary must gain physical possession of the card (or be able to duplicate it) plus must gain knowledge of the PIN.
The potential drawbacks of this card are:

1. Requires special reader – increases the cost of using the token.
2. Token loss - A lost token temporarily prevents its owner from gaining system access.
3. User dissatisfaction - Although users may have no difficulty in accepting the use of a memory card for ATM access, its use for computer access may be deemed inconvenient.

Smart Cards
A wide variety of devices qualify as smart tokens. These can be categorized along four dimensions that are not mutually exclusive:

1. Physical characteristics: Smart tokens include an embedded microprocessor. A smart token that looks like a bank card is called a smart card. Other smart tokens can look like calculators, keys, or other small portable objects.
2. User interface: Manual interfaces include a keypad and display for human/ token interaction.
3. Electronic interface: A smart card or other token requires an electronic interface to communicate with a compatible reader/writer. A card may have one or both of the following types of interface:
    - Contact: A contact smart card must be inserted into a smart card reader with a direct connection to a conductive contact plate on the surface of the card (typically gold plated). Transmission of commands, data, and card status takes place over these physical contact points.
    - Contactless: A contactless card requires only close proximity to a reader. Both the reader and the card have an antenna, and the two communicate using radio frequencies. Most contactless cards also derive power for the internal chip from this electromagnetic signal. The range is typically one-half to three inches for non-battery-powered cards, ideal for applications such as building entry and payment that require a very fast card interface.

4. Authentication protocol: The purpose of a smart token is to provide a means for user authentication. We can classify the authentication protocols used with smart tokens into three categories:
    - Static: With a static protocol, the user authenticates himself or herself to the token then the token authenticates the user to the computer. The latter half of this protocol is similar to the operation of a memory token.
    - Dynamic password generator: In this case, the token generates a unique password periodically (e.g., every minute). This password is then entered into the computer system for authentication, either manually by the user or electronically via the token. The token and the computer system must be initialized and kept synchronized so the computer knows the password that is current for this token.
    - Challenge-response: In this case, the computer system generates a challenge, such as a random string of numbers. The smart token generates a response based on the challenge. For example, public-key cryptography could be used and the token could encrypt the challenge string with the token's private key.

For user authentication, the most important category of smart token is the smart card, which has the appearance of a credit card, has an electronic interface, and may use any of the type of protocols just described.

A smart card contains within it an entire microprocessor, including processor, memory, and I/O ports. Some versions incorporate a special co-processing circuit for cryptographic operation to speed the task of encoding and decoding messages or generating digital signatures to validate the information transferred. In some cards, the I/O ports are directly accessible by a compatible reader by means of exposed electrical contacts. Other cards rely instead on an embedded antenna for wireless communication with the reader.

A typical smart card includes three types of memory. Read-only memory (ROM) stores data that does not change during the card's life, such as the card number and the cardholder's name. Electrically erasable programmable ROM (EEPROM) holds application data and programs, such as the protocols that the card can execute. It also holds data that may vary with time. For example, in a telephone card, the EEPROM holds the remaining talk time. Random access memory (RAM) holds temporary data generated when applications are executed.

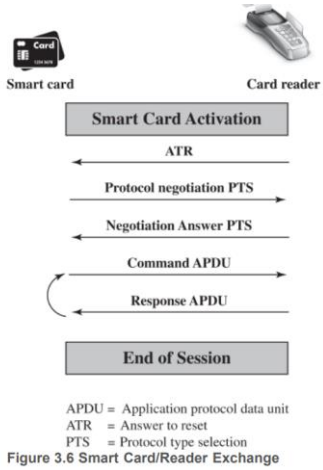


Figure 3.6 Smart Card/Reader Exchange

Electronic Identity Cards

An application of increasing importance is the use of a smart card as a national identity card for citizens. A national electronic identity (eID) card can serve the same purposes as other national ID cards, and similar cards such as a driver's license, for access to government and commercial services. In addition, an eID

card can provide stronger proof of identity and be used in a wider variety of applications. In effect, an eID card is a smart card that has been verified by the national government as valid and authentic.
One of the most recent and most advanced eID deployments is the German eID card neuer Personalausweis. The card has human-readable data printed on its surface, including the following:

- Personal data: Such as name, date of birth, and address; this is the type of printed information found on passports and drivers' licenses.
- Document number: An alphanumerical nine-character unique identifier of each card.
- Card access number (CAN): A six-digit decimal random number printed on the face of the card. This is used as a password, as explained subsequently.
- Machine readable zone (MRZ): Three lines of human- and machine-readable text on the back of the card. This may also be used as a password.

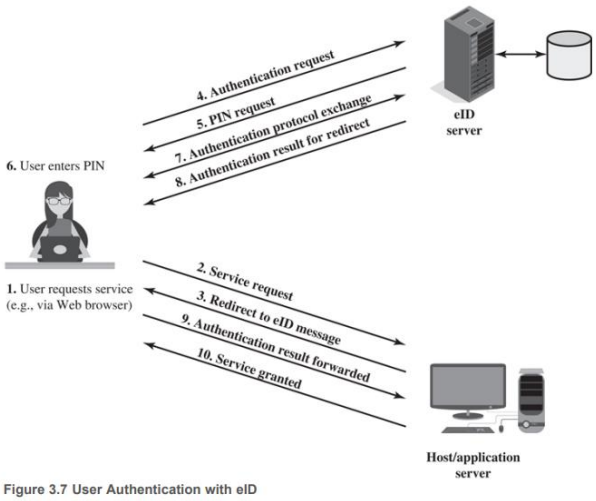


Figure 3.7 User Authentication with eID

4.4 Biometric Authentication

A biometric authentication system attempts to authenticate an individual based on his or her unique physical characteristics. These include static characteristics, such as fingerprints, hand geometry, facial characteristics, and retinal and iris patterns; and dynamic characteristics, such as voiceprint and signature.

In essence, biometrics is based on pattern recognition. Compared to passwords and tokens, biometric authentication is both technically more complex and expensive. While it is used in a number of specific applications, biometrics has yet to mature as a standard tool for user authentication to computer systems.

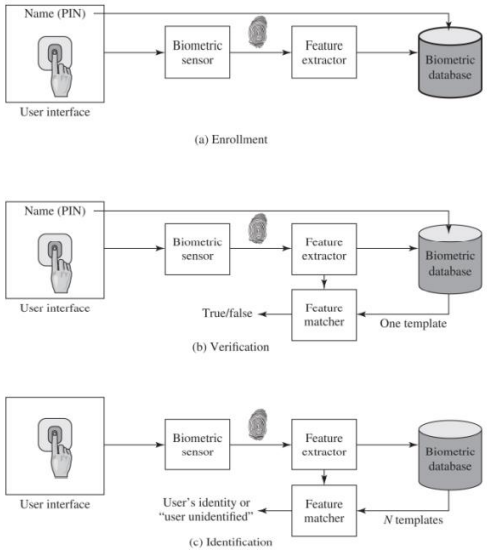
Physical Characteristics Used in Biometric Applications
A number of different types of physical characteristics are either in use or under study for user authentication. The most common are the following:

1. Facial characteristics: Facial characteristics are the most common means of human-to-human identification; thus it is natural to consider them for identification by computer. The most common approach is to define characteristics based on relative location and shape of key facial features, such as eyes, eyebrows, nose, lips, and chin shape. An alternative approach is to use an infrared camera to produce a face thermogram that correlates with the underlying vascular system in the human face.
2. Fingerprints: Fingerprints have been used as a means of identification for centuries, and the process has been systematized and automated particularly for law enforcement purposes. A fingerprint is the pattern of ridges and furrows on the surface of the fingertip. Fingerprints are believed to be unique across the entire human population. In practice, automated fingerprint

recognition and matching system extract a number of features from the fingerprint for storage as a numerical surrogate for the full fingerprint pattern.

3. Hand geometry: Hand geometry systems identify features of the hand, including shape, and lengths and widths of fingers.
4. Retinal pattern: The pattern formed by veins beneath the retinal surface is unique and therefore suitable for identification. A retinal biometric system obtains a digital image of the retinal pattern by projecting a low-intensity beam of visual or infrared light into the eye.
5. Iris: Another unique physical characteristic is the detailed structure of the iris.
6. Signature: Each individual has a unique style of handwriting and this is reflected especially in the signature, which is typically a frequently written sequence. However, multiple signature samples from a single individual will not be identical. This complicates the task of developing a computer representation of the signature that can be matched to future samples.
7. Voice: Whereas the signature style of an individual reflects not only the unique physical attributes of the writer but also the writing habit that has developed, voice patterns are more closely tied to the physical and anatomical characteristics of the speaker. Nevertheless, there is still a variation from sample to sample over time from the same speaker, complicating the biometric recognition task

Operations in a biometric authentication system



Each individual who is to be included in the database of authorized users must first be enrolled in the system. This is analogous to assigning a password to a user. For a biometric system, the user presents a name and, typically, some type of password or PIN to the system. At the same time, the system senses some biometric characteristic of this user (e.g., fingerprint of right index finger). The system digitizes the input then extracts a set of features that can be stored as a number or set of numbers representing this unique biometric characteristic; this set of numbers is referred to as the user's template. The user is now enrolled in the system, which maintains for the user a name (ID), perhaps a PIN or password, and the biometric value.

Depending on application, user authentication on a biometric system involves either verification or identification. Verification is analogous to a user logging on to a system by using a memory card or smart card coupled with a password or PIN. For biometric verification, the user enters a PIN and also uses a biometric sensor. The system extracts the corresponding feature and compares that to the template stored for this user. If there is a match, then the system authenticates this user.

For an identification system, the individual uses the biometric sensor but presents no additional information. The system then compares the presented template with the set of stored templates. If there is a match, then this user is identified. Otherwise, the user is rejected.

Biometric Accuracy

In any biometric scheme, some physical characteristic of the individual is mapped into a digital representation. For each individual, a single digital representation, or template, is stored in the computer. When the user is to be authenticated, the system compares the stored template to the presented template. Given the complexities of physical characteristics, we cannot expect that there will be an exact match between the two templates. Rather, the system uses an algorithm to generate a matching score (typically a single number) that quantifies the similarity between the input and the stored template.

A high-security application may require a very low false match rate. For a forensic application, in which the system is looking for possible candidates, to be checked further, the requirement may be for a low false nonmatch rate.

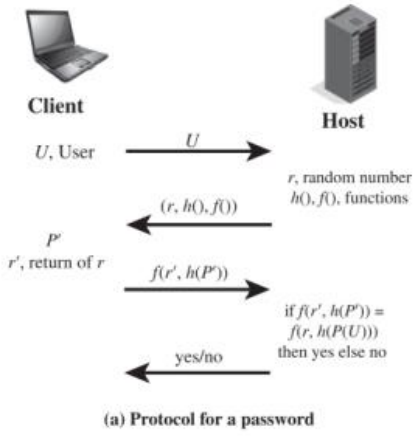
4.5 Remote User Authentication

The simplest form of user authentication is local authentication, in which a user attempts to access a system that is locally present, such as a stand-alone office PC or an ATM machine. The more complex case is that of remote user authentication, which takes place over the Internet, a network, or a communications link. Remote user authentication raises additional security threats, such as an eavesdropper being able to capture a password, or an adversary replaying an authentication sequence that has been observed.

To counter threats to remote user authentication, systems generally rely on some form of challenge-response protocol.

Password Protocol

The figure provides a simple example of a challenge-response protocol for authentication via password. In this example,



(a) Protocol for a password

1) a user first transmits his or her identity to the remote host.
2) The host generates a random number r, often called a nonce, and returns this nonce to the user. In addition, the host specifies two functions, h() and f(), to be used in the response. This transmission from host to user is the challenge.
3) The user's response is the quantity f(r', h(P')), where r'=r and P' is the user's password. The function h is a hash function, so the response consists of the hash function of the user's password combined with the random number using the function f.
4) The host stores the hash function of each registered user's password, depicted as h(P(U)) for user U. When the response arrives, the host compares the incoming to the calculated f(r, h(P(U))). If the quantities match, the user is authenticated.

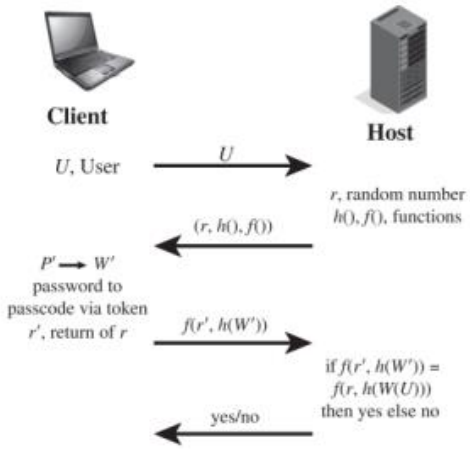
This scheme defends against several forms of attack. The host stores not the password but a hash code of the password. This secures the password from intruders into the host system. In addition, not even the hash of the password is transmitted directly, but rather a function in which the password hash is one of the arguments. Thus, for a suitable function f, the password hash cannot be captured during transmission.

Finally, the use of a random number as one of the arguments of f defends against a replay attack, in which an adversary captures the user's transmission and attempts to log on to a system by retransmitting the user's messages.

Token Protocol

Figure provides a simple example of a token protocol for authentication.

1. As before, a user first transmits his or her identity to the remote host.
2. The host returns a random number and the identifiers of functions f() and h() to be used in the response.
3. At the user end, the token provides a passcode W'. The token either stores a static passcode or generates a one-time random passcode. For a one-time random passcode, the token must be synchronized in some fashion with the host. In either case, the user activates the passcode by entering a password P'. This password is shared only between the user and the token and does not involve the remote host. The token responds to the host with the quantity f(r', h(W')).
4. For a static passcode, the host stores the hashed value h(W(U)); for a dynamic passcode, the host generates a one-time passcode (synchronized to that generated by the token) and takes its hash. Authentication then proceeds in the same fashion as for the password protocol.

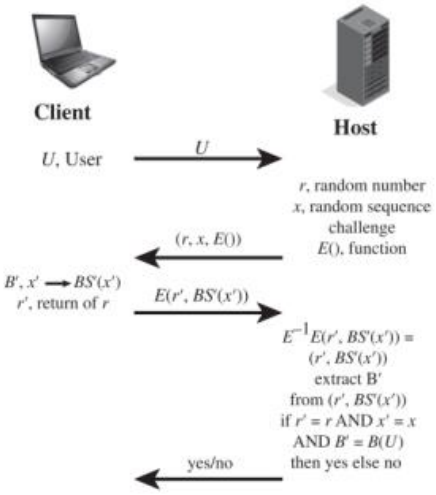


(b) Protocol for a token

Static Biometric Protocol

Figure is an example of a user authentication protocol using a static biometric.

1. As before, the user transmits an ID to the host.
2. Host responds with a random number r and, in this case, the identifier for an encryption E().
3. On the user side is a client system that controls a biometric device. The system generates a biometric template BT' from the user's biometric B' and returns the ciphertext E(r', D', BT') where D' identifies this particular biometric device.
4. The host decrypts the incoming message to recover the three transmitted parameters and compares these to locally stored values. For a match, the host must find r'=r. Also, the matching score between BT' and the stored template must exceed a predefined threshold. Finally, the host provides a simple authentication of the biometric capture device by comparing the incoming device ID to a list of registered devices at the host database.

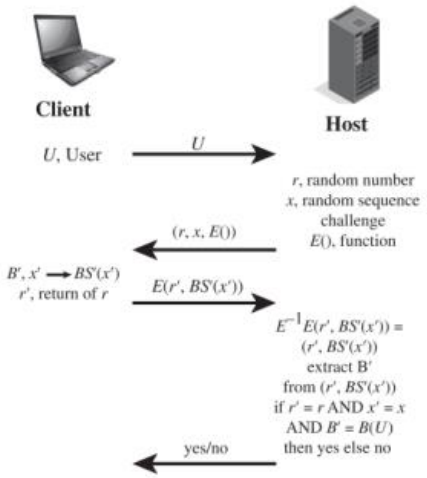


(d) Protocol for dynamic biometric

Dynamic Biometric Protocol

Figure is an example of a user authentication protocol using a dynamic biometric. The principal difference from the case of a stable biometric is that the host provides a random sequence as well as a random number as a challenge. The sequence challenge is a sequence of numbers, characters, or words.

1. The human user at the client end must then vocalize (speaker verification), type (keyboard dynamics verification), or write (handwriting verification) the sequence to generate a biometric signal BS'(x').
2. The client side encrypts the biometric signal and the random number.
3. At the host side, the incoming message is decrypted. The incoming random number r' must be an exact match to the random number that was originally used as a challenge (r). In addition, the host generates a comparison based on the incoming biometric signal BS'(x'). The stored template BT(U) for this user and the original signal x. If the comparison value exceeds a predefined threshold, the user is authenticated.



(d) Protocol for dynamic biometric

4.6 Two Factor Authentication

Two-factor authentication (2FA), sometimes referred to as two-step verification or dual-factor authentication, is a security process in which users provide two different authentication factors to verify themselves.

2FA is implemented to better protect both a user's credentials and the resources the user can access. Two-factor authentication provides a higher level of security than authentication methods that depend on single-factor authentication (SFA), in which the user provides only one factor -- typically, a password or passcode. Two-factor authentication methods rely on a user providing a password as the first factor and a second, different factor -- usually either a security token or a biometric factor, such as a fingerprint or facial scan.

Two-factor authentication adds an additional layer of security to the authentication process by making it harder for attackers to gain access to a person's devices or online accounts because, even if the victim's password is hacked, a password alone is not enough to pass the authentication check.

Two-factor authentication has long been used to control access to sensitive systems and data. Online service providers are increasingly using 2FA to protect their users' credentials from being used by hackers who stole a password database or used phishing campaigns to obtain user passwords.

Authentication Factors
There are several ways in which someone can be authenticated using more than one authentication method. Currently, most authentication methods rely on knowledge factors, such as a traditional password, while two-factor authentication methods add either a possession factor or an inherence factor.

Authentication factors, listed in approximate order of adoption for computing, include the following:

- A **knowledge factor** is something the user knows, such as a password, a personal identification number (PIN) or some other type of shared secret.
- A **possession factor** is something the user has, such as an ID card, a security token, a cellphone, a mobile device or a smartphone app, to approve authentication requests.

- A biometric factor, also known as an **inherence factor**, is something inherent in the user's physical self. These may be personal attributes mapped from physical characteristics, such as fingerprints authenticated through a fingerprint reader. Other commonly used inherence factors include facial and voice recognition or behavioral biometrics, such as keystroke dynamics, gait or speech patterns.
- A **location factor** is usually denoted by the location from which an authentication attempt is being made. This can be enforced by limiting authentication attempts to specific devices in a particular location or by tracking the geographic source of an authentication attempt based on the source Internet Protocol address or some other geolocation information, such as Global Positioning System (GPS) data, derived from the user's mobile phone or other device.
- A **time factor** restricts user authentication to a specific time window in which logging on is permitted and restricts access to the system outside of that window.

The vast majority of two-factor authentication methods rely on the first three authentication factors, though systems requiring greater security may use them to implement multifactor authentication (MFA), which can rely on two or more independent credentials for more secure authentication.

Steps in two-factor authentication

Enabling two-factor authentication varies depending on the specific application or vendor. However, two-factor authentication processes involve the same general, multistep process:

1. The user is prompted to log in by the application or the website.
2. The user enters what they know -- usually, username and password. Then, the site's server finds a match and recognizes the user.
3. For processes that don't require passwords, the website generates a unique security key for the user. The authentication tool processes the key, and the site's server validates it.
4. The site then prompts the user to initiate the second login step. Although this step can take a number of forms, the user has to prove that they have something only they would have, such as biometrics, a security token, an ID card, a smartphone or other mobile device. This is the inherence or possession factor.
5. Then, the user may have to enter a one-time code that was generated during step four.
6. After providing both factors, the user is authenticated and granted access to the application or website.

Practice Questions

1. Explain the different means of authenticating a user’s identity?
2. List and briefly describe the principal threats to the secrecy of passwords.
3. List and briefly describe the common techniques for selecting or assigning passwords.
4. Explain the differences between a simple memory card and a smart card.
5. List and briefly describe the principal physical characteristics used for biometric identification.
6. In the context of biometric user authentication, explain the terms enrollment, verification and identification.
7. Explain two factor authentication in detail.