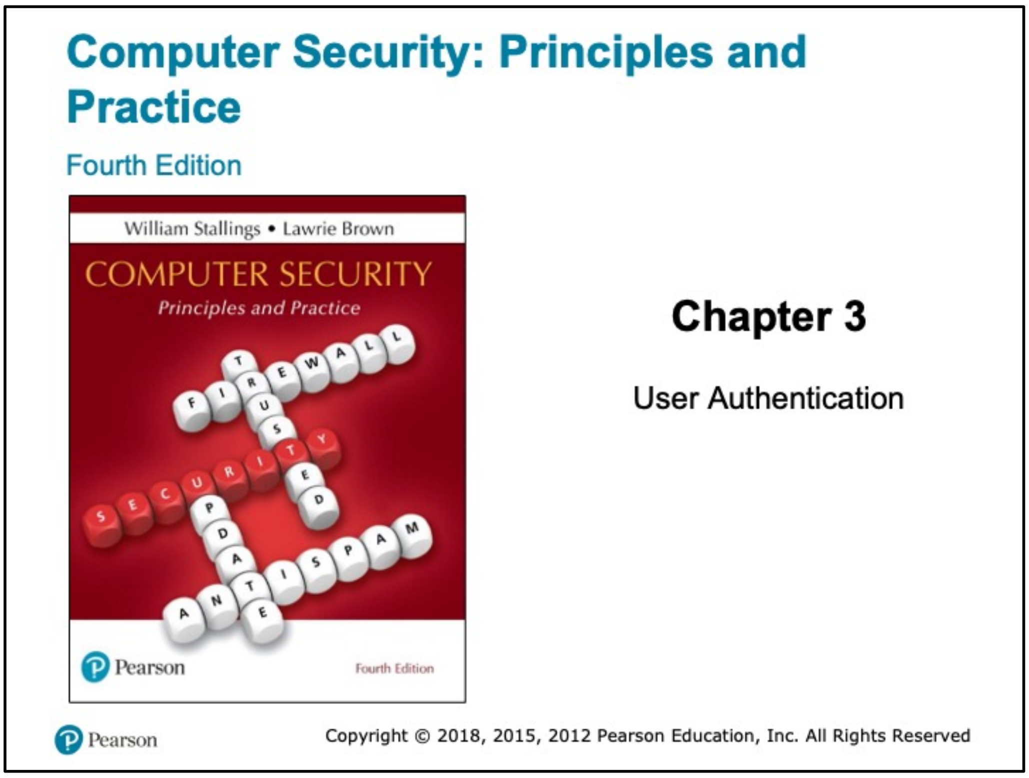
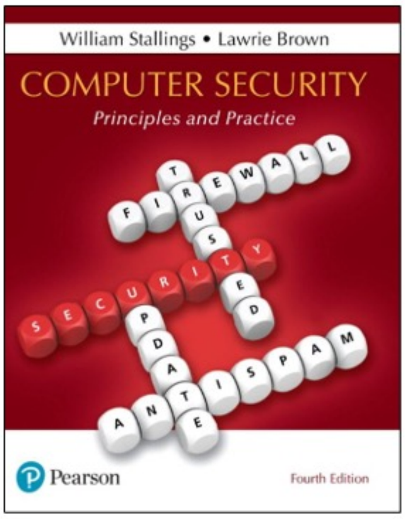


Computer Security: Principles and Practice

Fourth Edition



William Stallings • Lawrie Brown

COMPUTER SECURITY

Principles and Practice

Pearson

Fourth Edition

Chapter 3

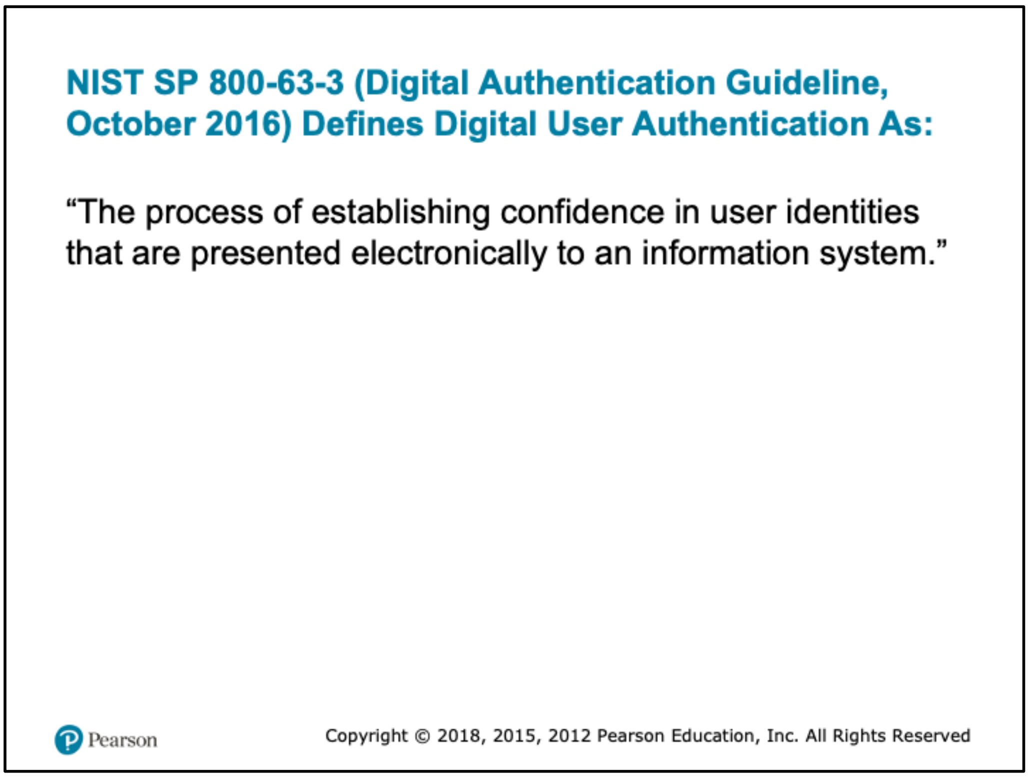
User Authentication

If this PowerPoint presentation contains mathematical equations, you may need to check that your computer has the following installed:
1) MathType Plugin
2) Math Player (free versions available)
3) NVDA Reader (free versions available)

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.

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 user authentication is distinct from message authentication. As defined in Chapter 2, message authentication is a procedure that allows communicating parties to verify that the contents of a received message have not been altered, and that the source is authentic. This chapter is concerned solely with user authentication.

This chapter first provides an overview of different means of user authentication, then examines each in some detail.



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."

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.

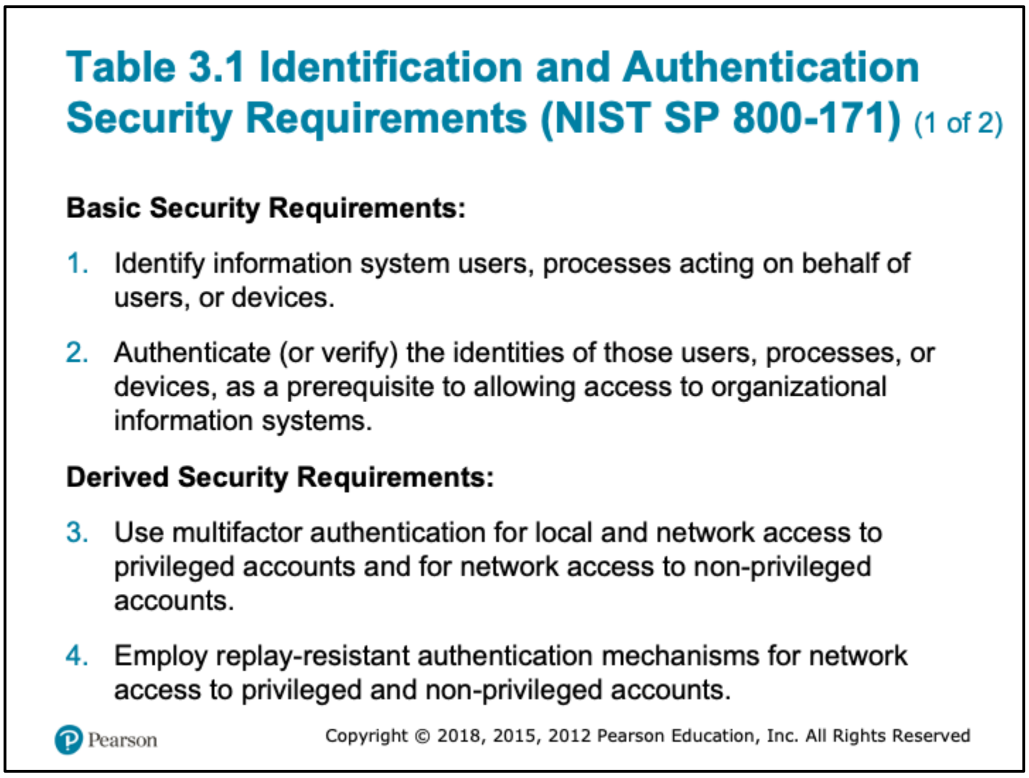


Table 3.1 Identification and Authentication Security Requirements (NIST SP 800-171) (1 of 2)

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.

Pearson

Table 3.1, from NIST SP 800-171 (Protecting Controlled Unclassified Information in Nonfederal Information Systems and Organizations , December 2016), provides a useful list of security requirements for identification and authentication services.

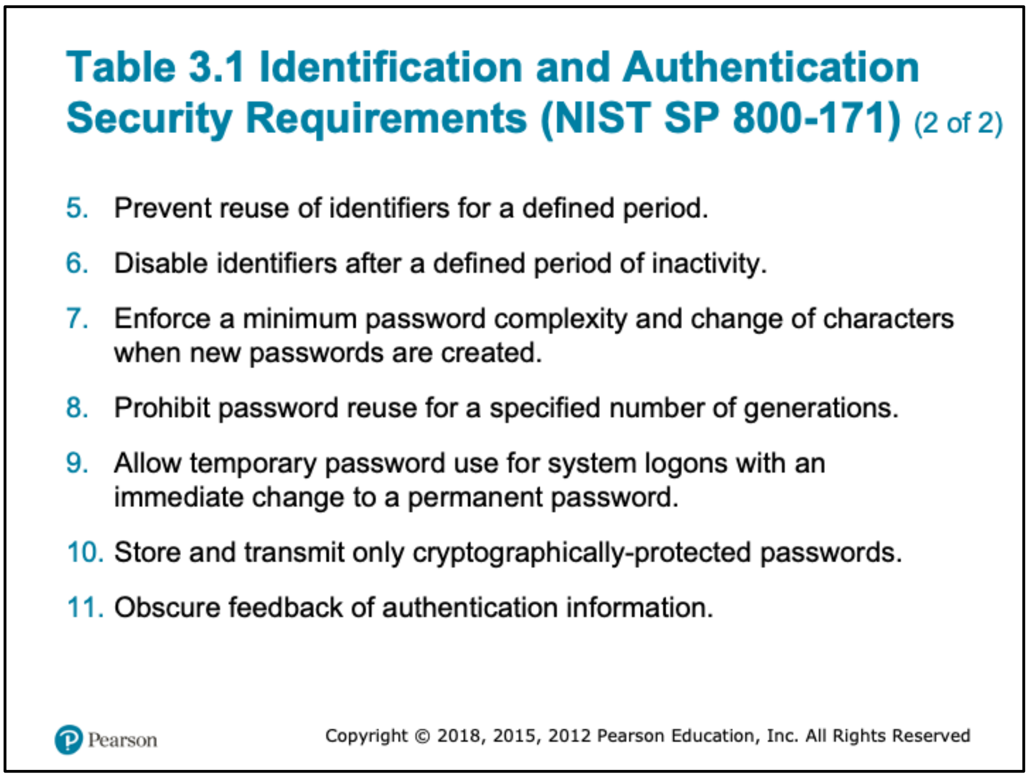
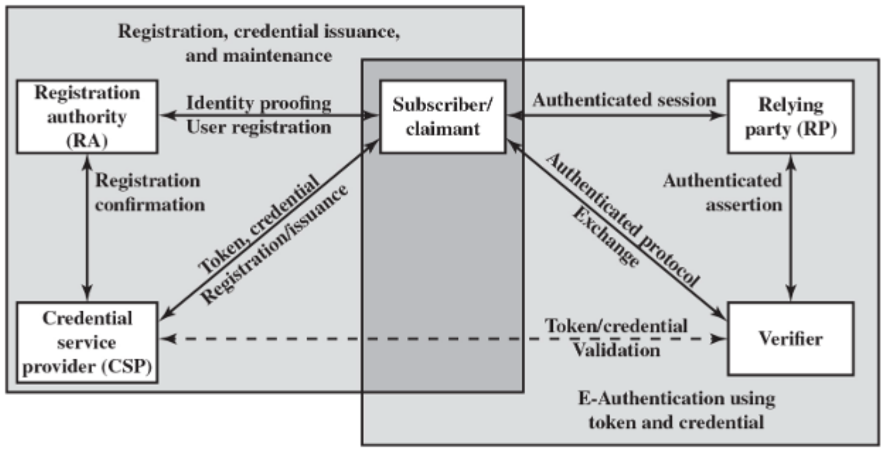


Table 3.1 Identification and Authentication Security Requirements (NIST SP 800-171) (2 of 2)

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.

Figure 3.1 The NIST SP 800-63-3 E-Authentication Architectural Model


Registration, credential issuance, and maintenance
Registration authority (RA)
Identity proofing
User registration
Subscriber/ claimant
Authenticated session
Relying party (RP)
Registration confirmation
Token, credential Registration/issuance
Authenticated protocol Exchange
Authenticated assertion
Credential service provider (CSP)
Token/credential Validation
Verifier
E-Authentication using token and credential

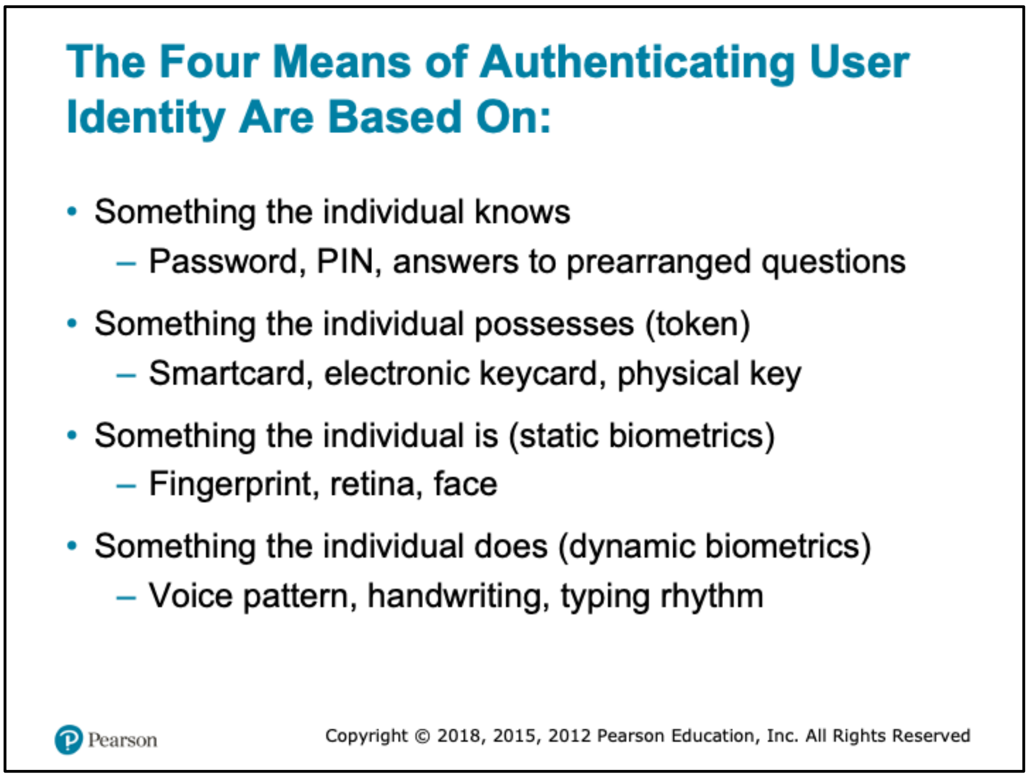
NIST SP 800-63-3 defines a general model for user authentication that involves a number of entities and procedures. We discuss this model with reference to Figure 3.1.

The initial requirement for performing user authentication is that the user must be registered with the system. The following is a typical sequence for registration. An applicant applies to a registration authority (RA) to become a subscriber of a credential service provider (CSP) . In this model, the RA is a trusted entity that establishes and vouches for the identity of an applicant to a CSP. The CSP then engages in an exchange with the subscriber. Depending on the details of the overall authentication system, the CSP issues some sort of electronic credential to the subscriber. The credential is a data structure that authoritatively binds an identity and additional attributes to a token possessed by a subscriber, and can be verified when presented to the verifier in an authentication transaction. The token could be an encryption key or an encrypted password that identifies the subscriber. The token may be issued by the CSP, generated directly by the subscriber, or provided by a third party. The token and credential may be used in subsequent authentication events.

Once a user is registered as a subscriber, the actual authentication process can take place between the subscriber and one or more systems that perform authentication and, subsequently, authorization. The party to be authenticated is called a claimant and the party verifying that identity is called a verifier . When a claimant successfully demonstrates possession and control of a token to a verifier through an authentication protocol, the verifier can verify that the claimant is the subscriber named in the corresponding credential. The verifier passes on an assertion about the identity of the subscriber to the relying party (RP) . That assertion includes identity information about a subscriber, such as the subscriber name, an identifier assigned at registration, or other subscriber attributes that were verified in the registration process. The RP can use the authenticated information provided by the verifier to make access control or authorization decisions.

An implemented system for authentication will differ from or be more complex than this simplified model, but the model illustrates the key roles and functions needed for a secure authentication system.

The first block is labeled, registration, credential issuance, and maintenance has three sub blocks, namely Subscriber slash claimant, Registration authority, R A, Credential service provider, C S P. Double headed arrow extends between Subscriber and R A, and is labeled, Identity proofing, user registration. Double headed arrow extends between C S P and R A, and is labeled, registration confirmation. Double headed arrow extends between C S P and Subscriber, and is labeled, token, credential, registration slash issuance. The second block is labeled, E authentication using token and credential has three sub blocks, namely Subscriber slash claimant, Relying party, R P, Verifier. Double headed arrow extends between Subscriber and R P, and is labeled, Authenticated session. Double headed arrow extends between R P and Verifier, and is labeled, authenticated assertion. Double headed arrow extends between Verifier and Subscriber, and is labeled, authenticated protocol exchange. Double headed arrow extends between C S P and verifier and is labeled, token slash credential validation.



The Four Means of Authenticating User Identity Are Based On:

- Something the individual knows
  - Password, PIN, answers to prearranged questions
- Something the individual possesses (token)
  - Smartcard, electronic keycard, physical key
- Something the individual is (static biometrics)
  - Fingerprint, retina, face
- Something the individual does (dynamic biometrics)
  - Voice pattern, handwriting, typing rhythm

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.

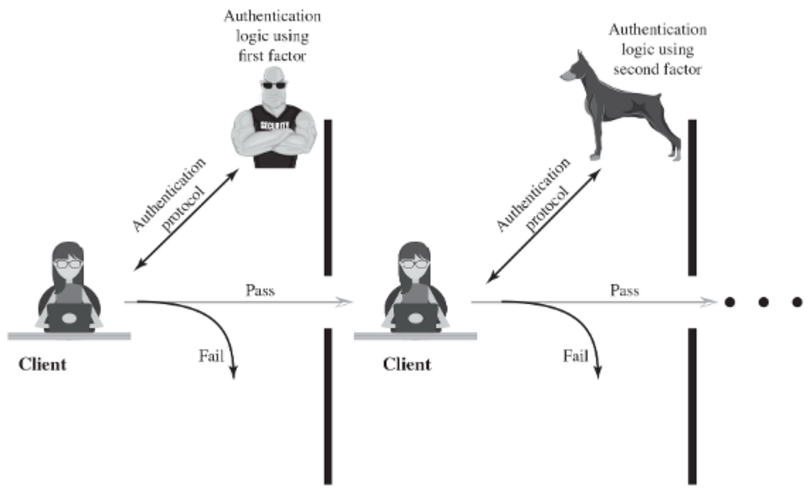
• **Something the individual possesses:** Examples include electronic keycards, smart cards, and physical keys. This type of authenticator is referred to as a *token*.

• **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.

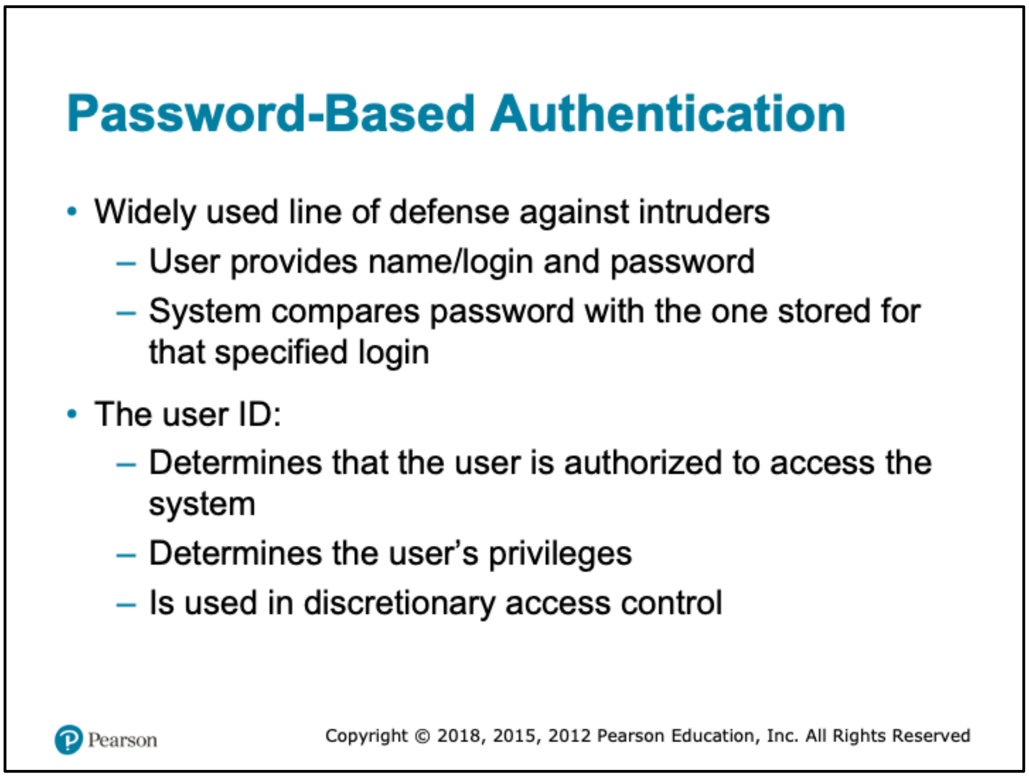
All of these methods, properly implemented and used, can provide secure user authentication. However, each method has problems. An adversary may be able to guess or steal a password. Similarly, an adversary may be able to forge or steal a token. A user may forget a password or lose a token. Further, 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.

Figure 3.2 Multifactor Authentication


Authentication logic using first factor
Authentication logic using second factor
Authentication protocol
Authentication protocol
Pass
Pass
Fail
Fail
Client
Client

Multifactor authentication refers to the use of more than one of the authentication means in the preceding list (see Figure 3.2). 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.

The first illustration depicts an authentication protocol exchanged between the client and the first factor, security personnel. If the authentication protocol passes, the client enters the next stage, else the authentication fails. The second illustration depicts an authentication protocol exchanged between the client and the second factor, using logic. If the authentication protocol passes, the client enters the next stage, else the authentication fails.



Password-Based Authentication

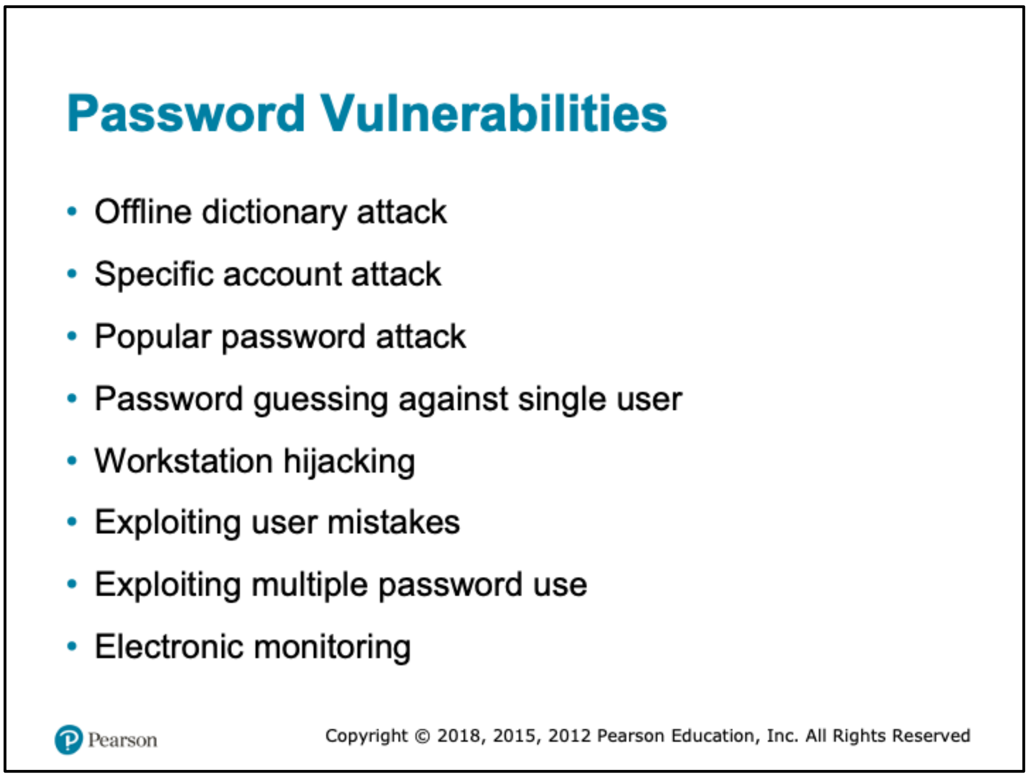
- Widely used line of defense against intruders
  - User provides name/login and password
  - System compares password with the one stored for that specified login
- The user ID:
  - Determines that the user is authorized to access the system
  - Determines the user's privileges
  - Is used in discretionary access control

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 supervisory 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.



Password Vulnerabilities

- Offline dictionary attack
- Specific account attack
- Popular password attack
- Password guessing against single user
- Workstation hijacking
- Exploiting user mistakes
- Exploiting multiple password use
- Electronic monitoring

In this subsection, we outline the main forms of attack against password-based authentication and briefly outline a countermeasure strategy. The remainder of Section 3.2 goes into more detail on the key countermeasures.

We can identify the following attack strategies:

• **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.

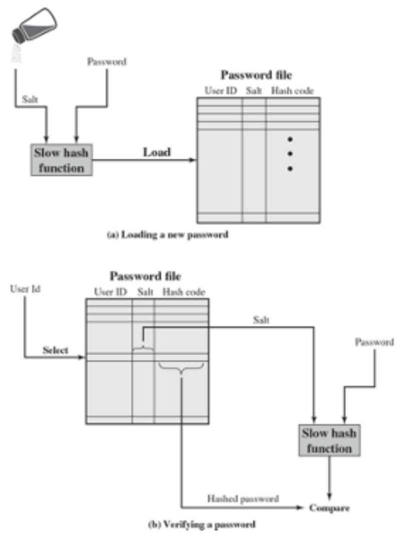
• **Workstation hijacking:** The attacker waits until a logged-in workstation is unattended. The standard countermeasure is automatically logging the workstation out after a period of inactivity. Intrusion detection schemes can be used to detect changes in user behavior.

• **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.

Figure 3.3 UNIX Password Scheme



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. The following procedure is employed (Figure 3.3a). 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 [MORR79]. 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 [WAGN00].

When a user attempts to log on to a UNIX system, the user provides an ID and a password (Figure 3.3b). 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.

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.

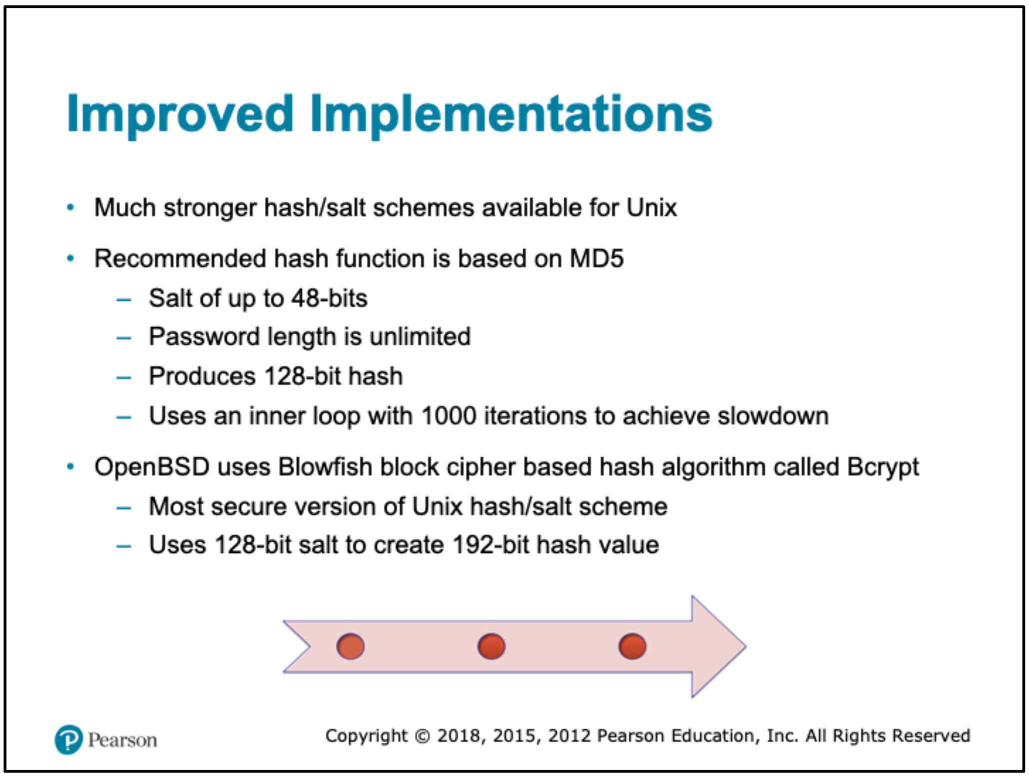
- It greatly increases the difficulty of offline dictionary attacks. For a salt of length b bits, the number of possible passwords is increased by a factor of 2b, increasing the difficulty of guessing a password in a dictionary attack.

- 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.

To see the second point, consider the way that an offline dictionary attack would work. The attacker obtains a copy of the password file. Suppose first that the salt is not used. The attacker's goal is to guess a single password. To that end, the attacker submits a large number of likely passwords to the hashing function. If any of the guesses matches one of the hashes in the file, then the attacker has found a password that is in the file. But faced with the UNIX scheme, the attacker must take each guess and submit it to the hash function once for each salt value in the dictionary file, multiplying the number of guesses that must be checked.

There are two threats to the UNIX password scheme. First, a user can gain access on a machine using a guest account or by some other means and then run a password guessing program, called a password cracker, on that machine. The attacker should be able to check many thousands of possible passwords with little resource consumption. In addition, if an opponent is able to obtain a copy of the password file, then a cracker program can be run on another machine at leisure. This enables the opponent to run through millions of possible passwords in a reasonable period.

Illustration a, depicts a password file. The password file table has column headers, User I D, Salt, Hash Code. The slow hash function has inputs salt, and password. The slow hash function has inputs salt, and password. The slow hash function is given as input to the password file. Illustration b depicts a password file. The password file table has column headers, User I D, Salt, Hash Code. The password file has select input, user id given as input to the user id column. The slow hash function has inputs salt, and password. The slow hash function has inputs salt, and password. The hashed password and the slow hash function outputs are compared.

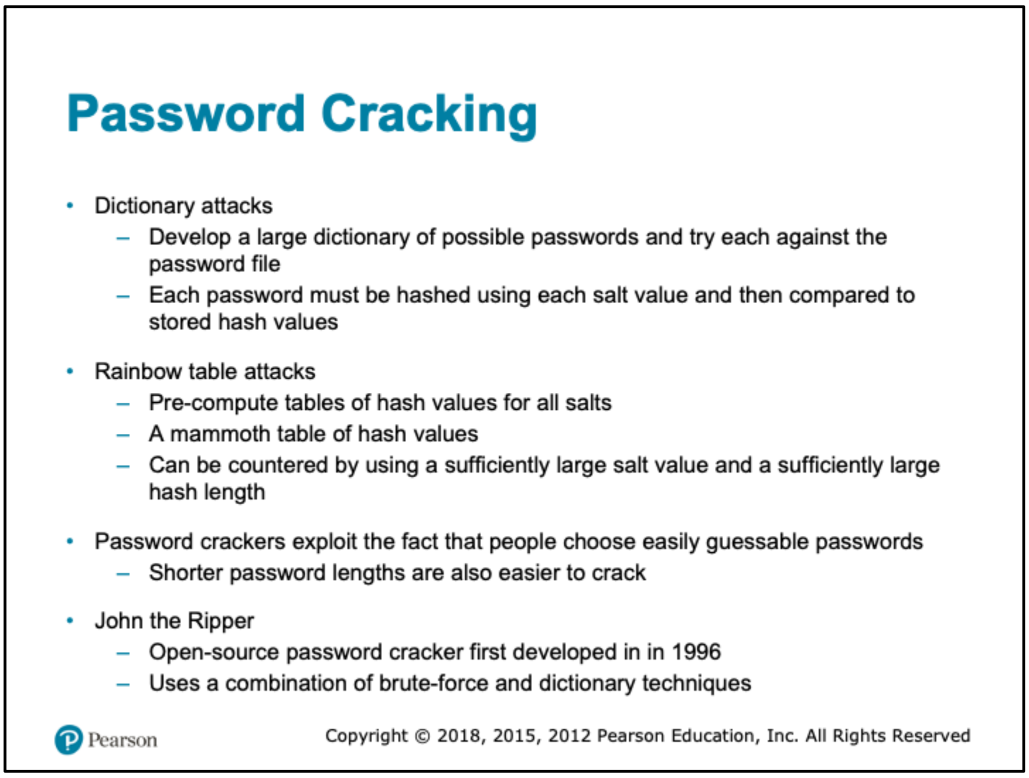


Improved Implementations

- Much stronger hash/salt schemes available for Unix
- Recommended hash function is based on MD5
  - Salt of up to 48-bits
  - Password length is unlimited
  - Produces 128-bit hash
  - Uses an inner loop with 1000 iterations to achieve slowdown
- OpenBSD uses Blowfish block cipher based hash algorithm called Bcrypt
  - Most secure version of Unix hash/salt scheme
  - Uses 128-bit salt to create 192-bit hash value

There are other, much stronger, hash/salt schemes available for UNIX. The recommended hash function for many UNIX systems, including Linux, Solaris, and FreeBSD (a widely used open source UNIX), is based on the MD5 secure hash algorithm (which is similar to, but not as secure as SHA-1). The MD5 crypt routine uses a salt of up to 48 bits and effectively has no limitations on password length. It produces a 128-bit hash value. It is also far slower than crypt(3). To achieve the slowdown, MD5 crypt uses an inner loop with 1000 iterations.

Probably the most secure version of the UNIX hash/salt scheme was developed for OpenBSD, another widely used open source UNIX. This scheme, reported in [PROV99], uses a hash function based on the Blowfish symmetric block cipher. The hash function, called Bcrypt, is quite slow to execute. Bcrypt allows passwords of up to 55 characters in length and requires a random salt value of 128 bits, to produce a 192-bit hash value. Bcrypt also includes a cost variable; an increase in the cost variable causes a corresponding increase in the time required to perform a Bcyrpt hash. The cost assigned to a new password is configurable, so that administrators can assign a higher cost to privileged users.



Password Cracking

- Dictionary attacks
  - Develop a large dictionary of possible passwords and try each against the password file
  - Each password must be hashed using each salt value and then compared to stored hash values
- Rainbow table attacks
  - Pre-compute tables of hash values for all salts
  - A mammoth table of hash values
  - Can be countered by using a sufficiently large salt value and a sufficiently large hash length
- Password crackers exploit the fact that people choose easily guessable passwords
  - Shorter password lengths are also easier to crack
- John the Ripper
  - Open-source password cracker first developed in in 1996
  - Uses a combination of brute-force and dictionary techniques

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 salt value in the password file and then compared to stored hash values. If no match is found, then 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,

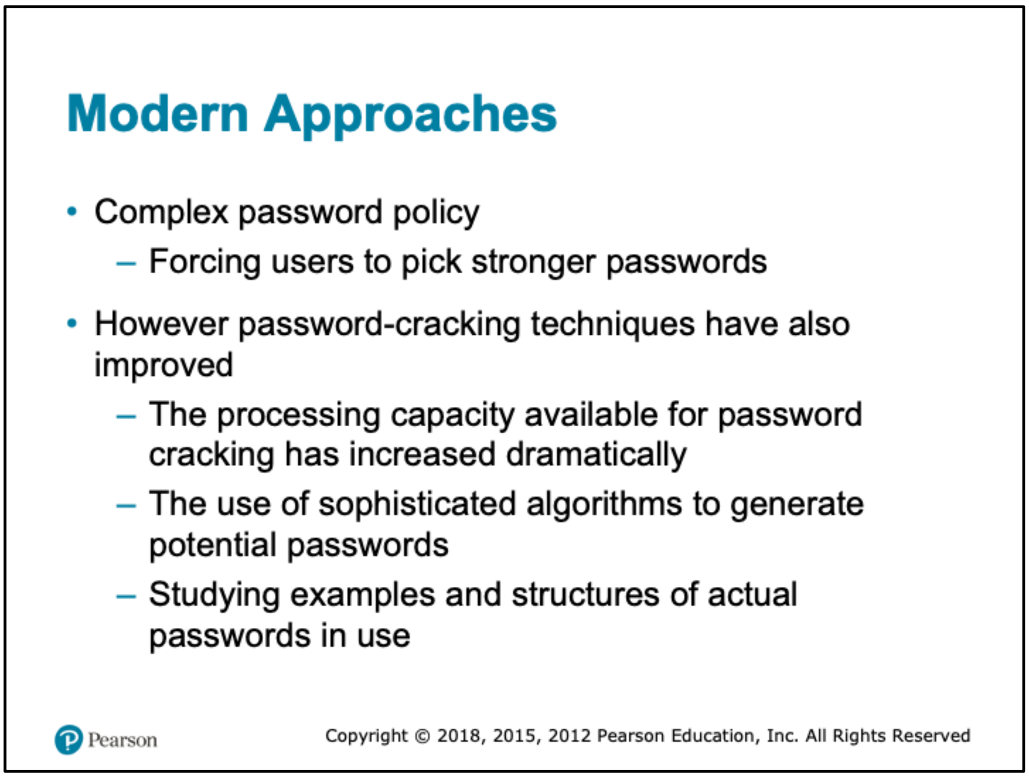
An alternative is to trade off space for time by precomputing potential hash values. In this approach the attacker generates a large dictionary of possible passwords. For each password, the attacker generates the hash values associated with each possible salt value. The result is a mammoth table of hash values known as a **rainbow table**. For example, [OECH03] showed that using 1.4 GB of data, he could crack 99.9% of all alphanumeric Windows password hashes in 13.8 seconds. This approach can be countered by using a sufficiently large salt value and a sufficiently large hash length. Both the FreeBSD and OpenBSD approaches should be secure from this attack for the foreseeable future.

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. [BONN12] summarizes the results of a number of studies over the past few years involving over 40 million hacked passwords, as well as their own analysis of almost 70 million anonymized passwords of Yahoo! users, and found a tendency toward six to eight characters of length and a strong dislike of non-alphanumeric characters in passwords.

The analysis of the 70 million passwords in [BONN12] estimates that passwords provide fewer than 10 bits of security against an online, trawling attack, and only about 20 bits of security against an optimal offline dictionary attack. In other words, an attacker who can manage 10 guesses per account, typically within the realm of rate-limiting mechanisms, will compromise around 1% of accounts, just as they would against random 10-bit strings. Against an optimal attacker performing unrestricted brute force and wanting to break half of all available accounts, passwords appear to be roughly equivalent to 20-bit random strings. It can be seen then that using offline search enables an adversary to break a large number of accounts, even if a significant amount of iterated hashing is Used.

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 [OPEN13].



Modern Approaches

- Complex password policy
  - Forcing users to pick stronger passwords
- However password-cracking techniques have also improved
  - The processing capacity available for password cracking has increased dramatically
  - The use of sophisticated algorithms to generate potential passwords
  - Studying examples and structures of actual passwords in use

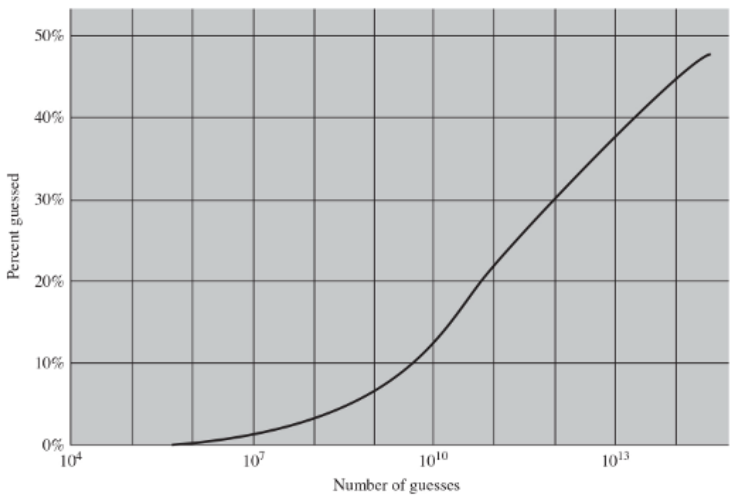
Sadly, this type of vulnerability has not lessened in the past 25 years or so. Users are doing a better job of selecting passwords, and organizations are doing a better job of forcing users to pick stronger passwords, a concept known as a complex password policy, as discussed subsequently. However, password-cracking 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. A PC running a single AMD Radeon HD7970 GPU, for instance, can try on average an $8.2 * 10^9$ password combinations each second, depending on the algorithm used to scramble them [GOOD12a]. 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. For example, [NARA05] developed a model for password generation using the probabilities of letters in natural language. The researchers used standard Markov modeling techniques from natural language processing to dramatically reduce the size of the password space to be searched.

But the best results have been achieved by studying examples of actual passwords in use. 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 [TIMM10]. Since then, numerous sets of leaked password files have become available for analysis.

Using large datasets of leaked passwords as training data, [WEIR09] reports on the development of a probabilistic context-free grammar for password cracking. In this approach, guesses are ordered according to their likelihood, based on the frequency of their character-class structures in the training data, as well as the frequency of their digit and symbol substrings. This approach has been shown to be efficient in password cracking [KELL12, ZHAN10].

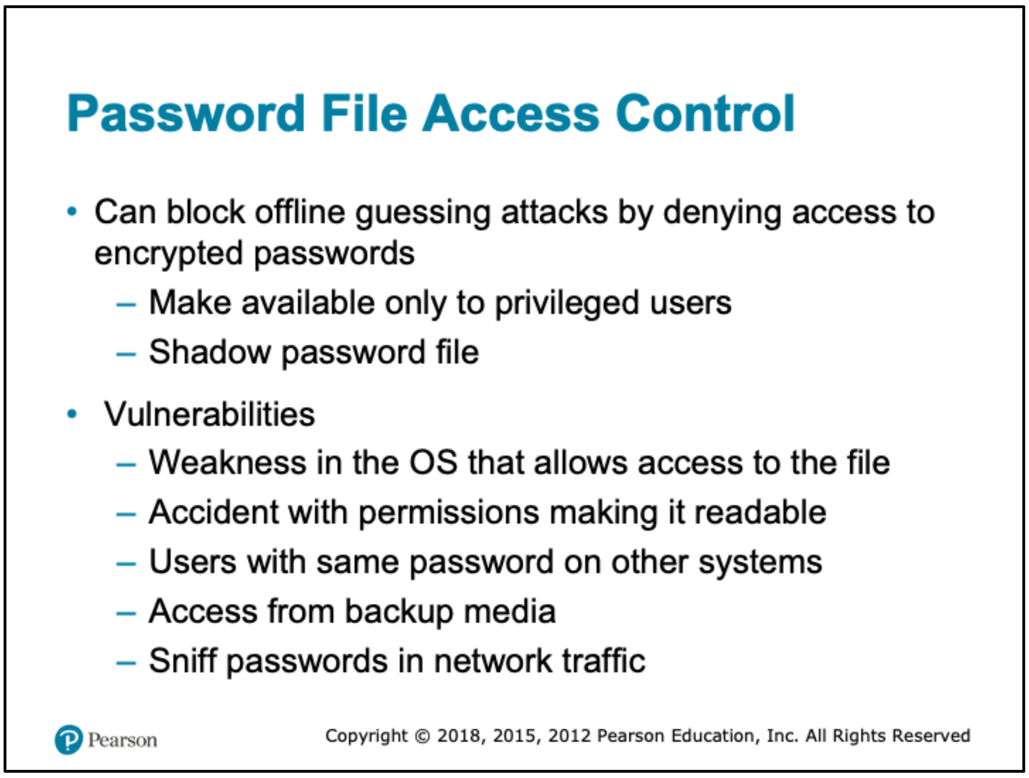
Figure 3.4 The Percentage of Passwords Guessed After a Given Number of Guesses



Pearson

[MAZU13] reports on an analysis of the passwords used by over 25,000 students at a research university with a complex password policy. The analysts used the password-cracking approach introduced in [WEIR09]. They used a database consisting of a collection of leaked password files, including the RockYou file. Figure 3.4 summarizes a key result from the paper. The graph shows the percentage of passwords that have been recovered as a function of the number of guesses. As can be seen, over 10% of the passwords are recovered after only 1010 guesses. After 1013 guesses, almost 40% of the passwords are recovered.

The horizontal axis ranges from 10 to the fourth power to10 to the fourteenth power. The vertical axis ranges from 0% to 50%, in increments of 10%. The graph plots a curve that rises from (10 to the fifth power, 0%), (10 to the 10th power, 12%), (10 to the 12th power, 30%), (10 to the 14th power, 48%). All values are estimated.


Password File Access Control

- Can block offline guessing attacks by denying access to encrypted passwords
  - Make available only to privileged users
  - Shadow password file
- Vulnerabilities
  - Weakness in the OS that allows access to the file
  - Accident with permissions making it readable
  - Users with same password on other systems
  - Access from backup media
  - Sniff passwords in network traffic

Pearson

One way to thwart 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.

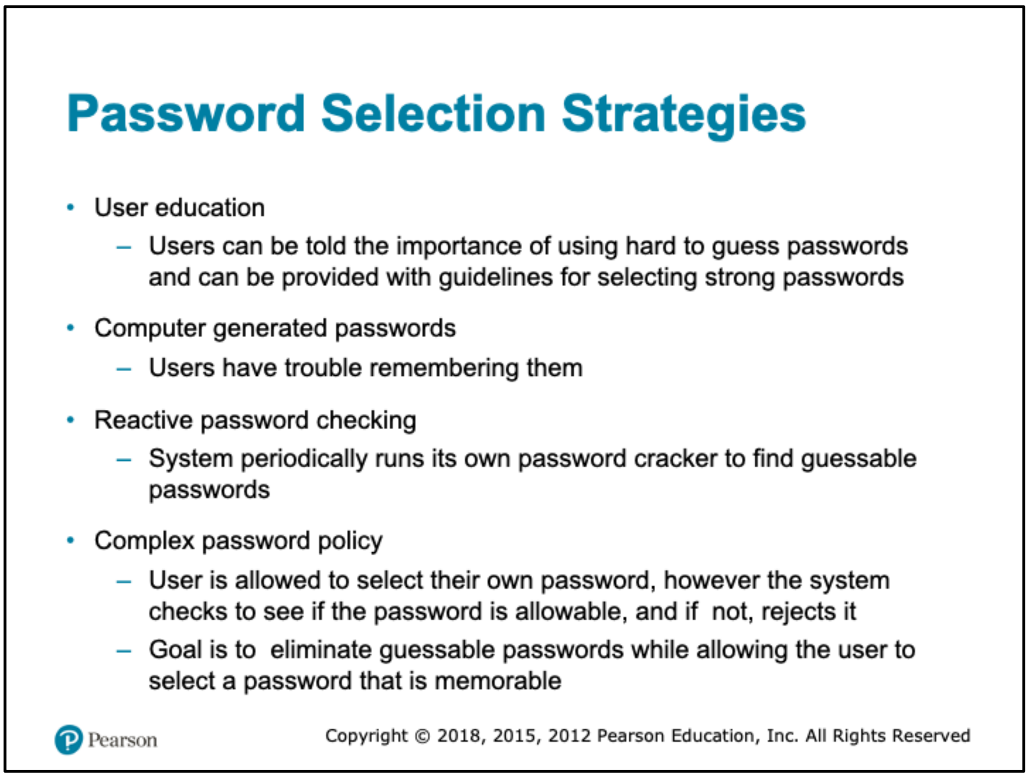
• An accident of protection might render the password file readable, thus compromising all the accounts.

• 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

- User education
  - Users can be told the importance of using hard to guess passwords and can be provided with guidelines for selecting strong passwords
- Computer generated passwords
  - Users have trouble remembering them
- Reactive password checking
  - System periodically runs its own password cracker to find guessable passwords
- Complex password policy
  - User is allowed to select their own password, however the system checks to see if the password is allowable, and if not, rejects it
  - Goal is to eliminate guessable passwords while allowing the user to select a password that is memorable

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

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.

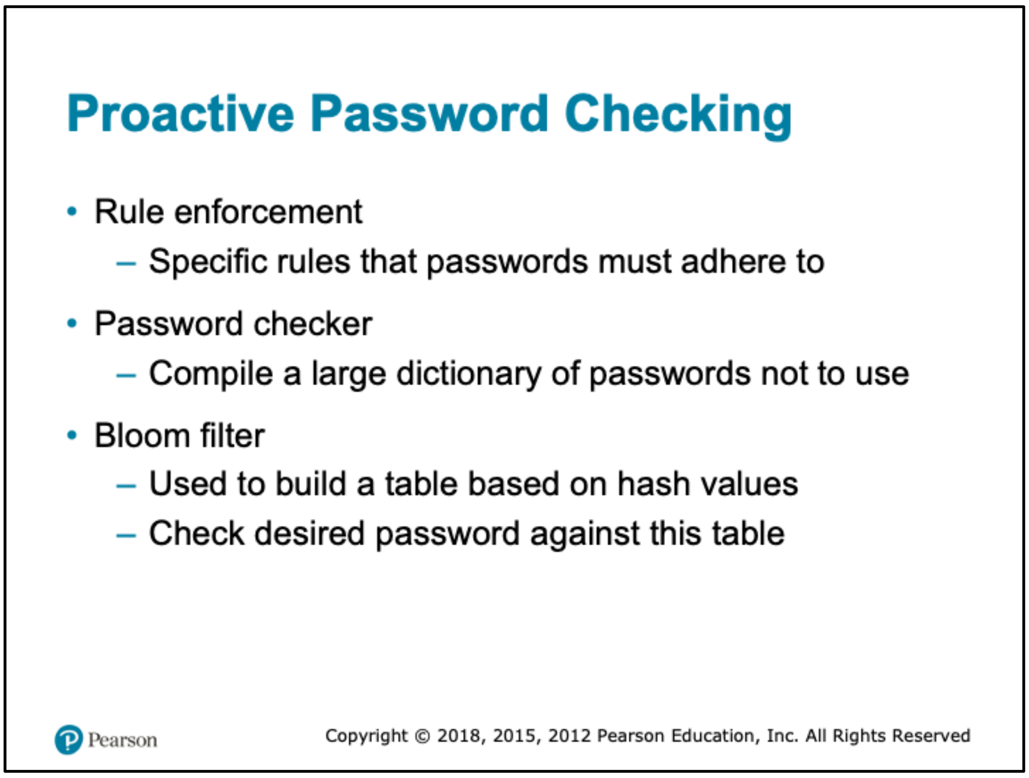
Nonetheless, it makes sense to provide users with guidelines on the selection of passwords. Perhaps the best approach is the following advice: A good technique for choosing a password is to use the first letter of each word of a phrase. However, do not pick a well-known phrase like “An apple a day keeps 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 that users can generally remember such passwords but that 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. FIPS 181 defines one of the best-designed automated password generators. The standard includes not only a description of the approach but also a complete listing of the C source code of the algorithm. The algorithm generates words by forming pronounceable syllables and concatenating them to form a word. A random number generator produces a random stream of characters used to construct the syllables and words.

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. A good example is the openware Jack the Ripper password cracker (openwall.com/john/pro/), which works on a variety of operating systems.

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. In the remainder of this subsection, we look at possible approaches to proactive password checking.



Proactive Password Checking

- Rule enforcement
  - Specific rules that passwords must adhere to
- Password checker
  - Compile a large dictionary of passwords not to use
- Bloom filter
  - Used to build a table based on hash values
  - Check desired password against this table

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 NIST considers basic16 and comprehensive8 equivalent, [KELL12] found that basic16 is superior against large numbers of guesses. Combined with a prior result that basic16 is also easier for users [KOMA11], this suggests basic16 is the better policy choice.

Although this approach is superior to simply educating users, it may not be sufficient to thwart 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.

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.

A technique [SPAF92a, SPAF92b] for developing an effective and efficient proactive password checker that is based on rejecting words on a list has been implemented on a number of systems, including Linux. It is based on the use of a Bloom filter [BLOO70].

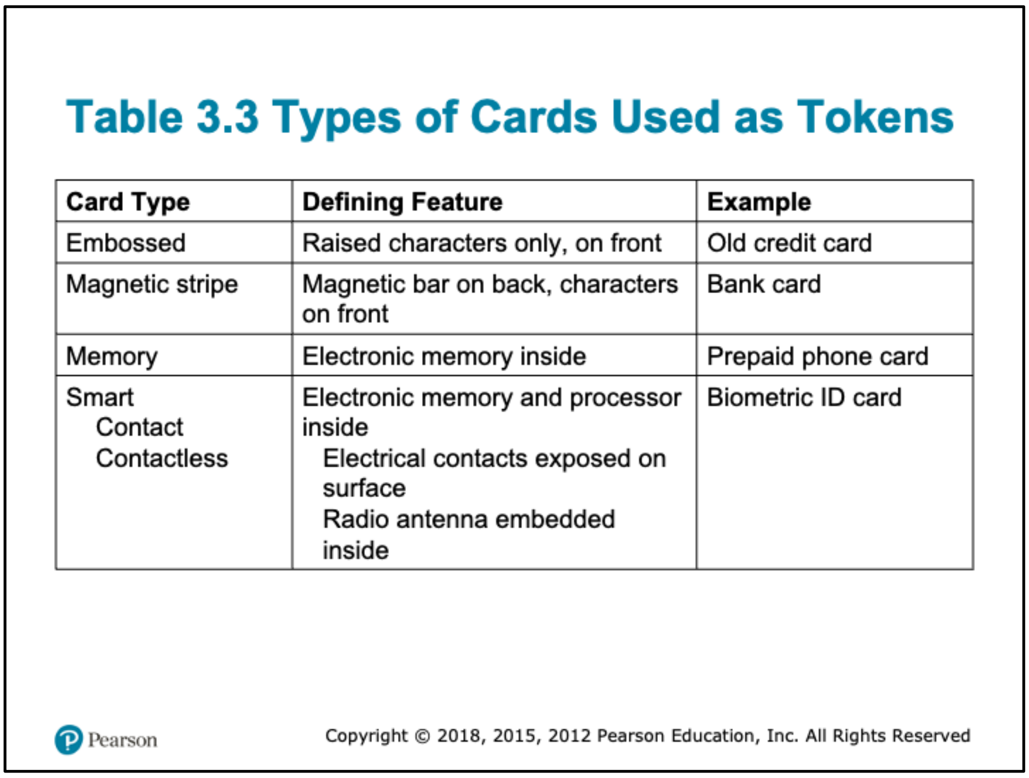
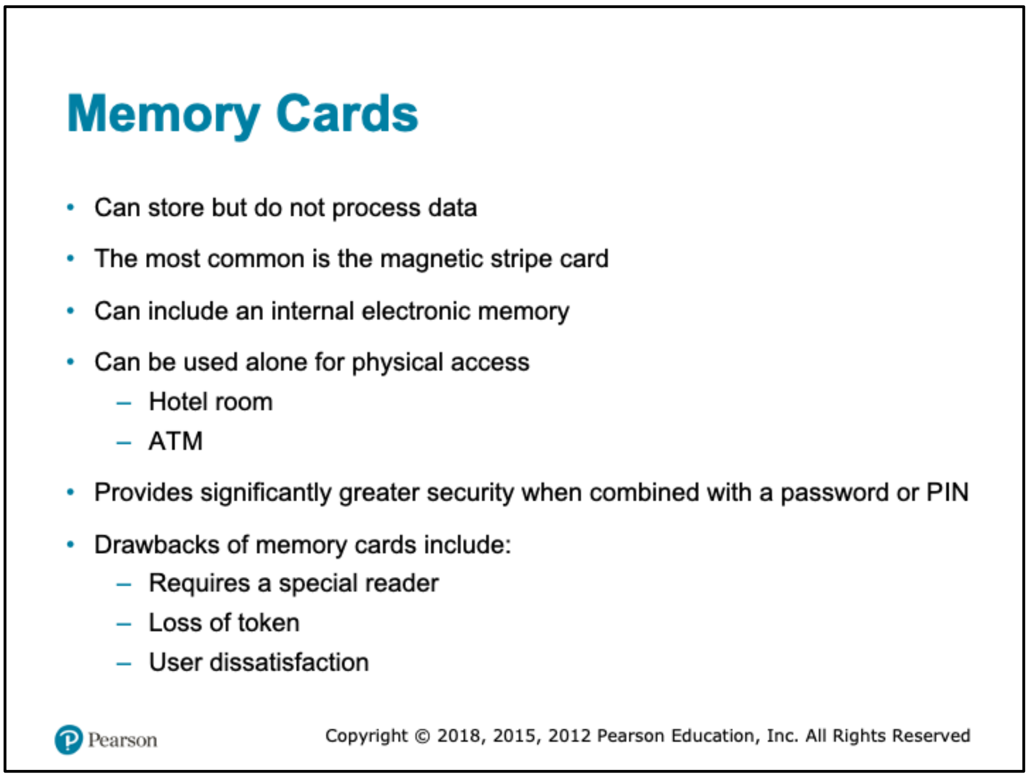


Table 3.3 Types of Cards Used as Tokens

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 Electrical contacts exposed on surface Radio antenna embedded inside	Biometric ID card

Objects that a user possesses for the purpose of user authentication are called tokens. In this section, we examine two types of tokens that are widely used; these are cards that have the appearance and size of bank cards (see Table 3.3).


Memory Cards

- Can store but do not process data
- The most common is the magnetic stripe card
- Can include an internal electronic memory
- Can be used alone for physical access
  - Hotel room
  - ATM
- Provides significantly greater security when combined with a password or PIN
- Drawbacks of memory cards include:
  - Requires a special reader
  - Loss of token
  - User dissatisfaction

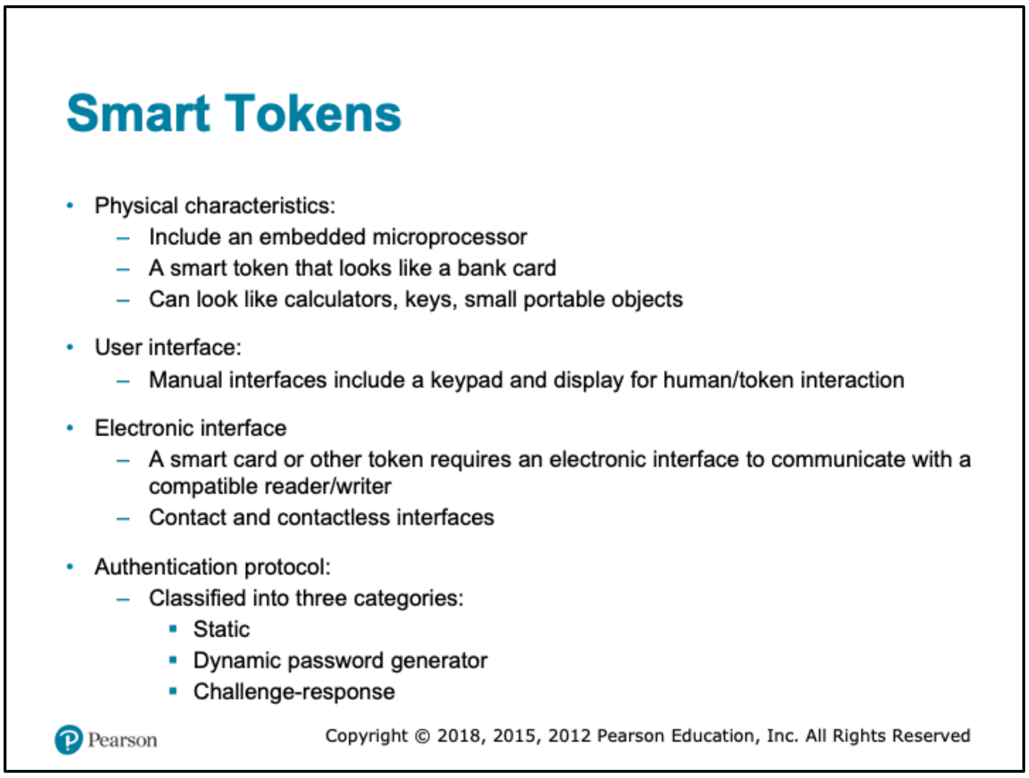
Memory cards can store but not process data. The most common such card is the bank card with a magnetic stripe on the back. A magnetic stripe can store only a simple security code, which can be read (and unfortunately reprogrammed) by an inexpensive card reader. There are also memory cards that include an internal electronic memory.

Memory cards can be used alone for physical access, such as a hotel room. For computer user authentication, such cards are typically used with 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. Among the potential drawbacks NIST SP 800-12 (*An Introduction to Computer Security: The NIST Handbook* , October 1995) notes the following:

• Requires special reader: This increases the cost of using the token and creates the requirement to maintain the security of the reader's hardware and software.

• Token loss: A lost token temporarily prevents its owner from gaining system access. Thus there is an administrative cost in replacing the lost token. In addition, if the token is found, stolen, or forged, then an adversary now need only determine the PIN to gain unauthorized access.

• 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 Tokens

- Physical characteristics:
  - Include an embedded microprocessor
  - A smart token that looks like a bank card
  - Can look like calculators, keys, small portable objects
- User interface:
  - Manual interfaces include a keypad and display for human/token interaction
- Electronic interface
  - A smart card or other token requires an electronic interface to communicate with a compatible reader/writer
  - Contact and contactless interfaces
- Authentication protocol:
  - Classified into three categories:
    - Static
    - Dynamic password generator
    - Challenge-response

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

• 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.

• User interface: Manual interfaces include a keypad and display for human/token interaction.

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.

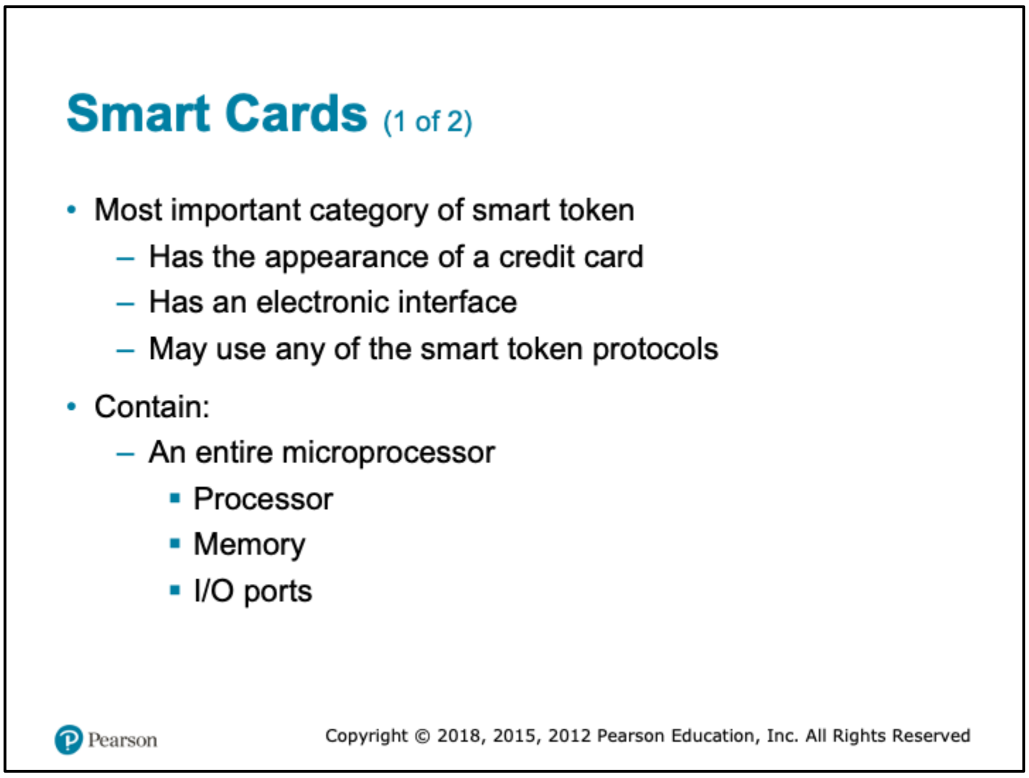
—— Contactless: A contactless card requires only close proximity to a reader. Both the reader and the card have an antenna, and the two communicate using radio frequencies. Most contactless cards also derive power for the internal chip from this electromagnetic signal. The range is typically one-half to three inches for non-battery-powered cards, ideal for applications such as building entry and payment that require a very fast card interface.

• 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 and 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 that 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.



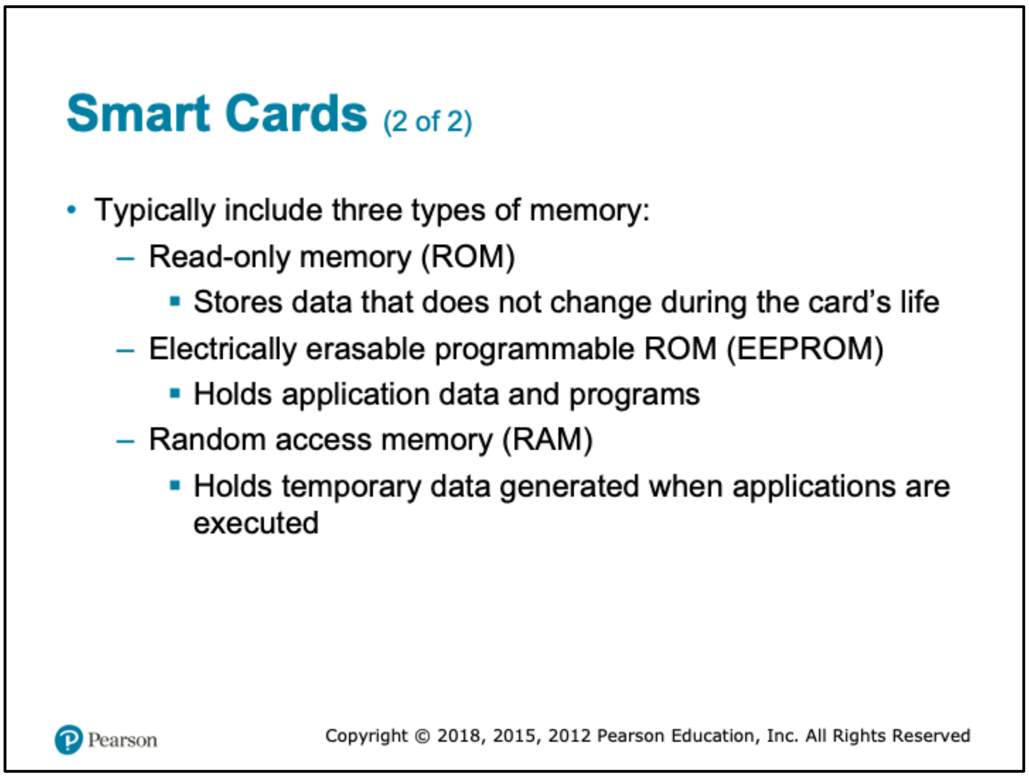
Smart Cards (1 of 2)

- Most important category of smart token
  - Has the appearance of a credit card
  - Has an electronic interface
  - May use any of the smart token protocols
- Contain:
  - An entire microprocessor
    - Processor
    - Memory
    - I/O ports

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. The remainder of this section discusses smart cards.

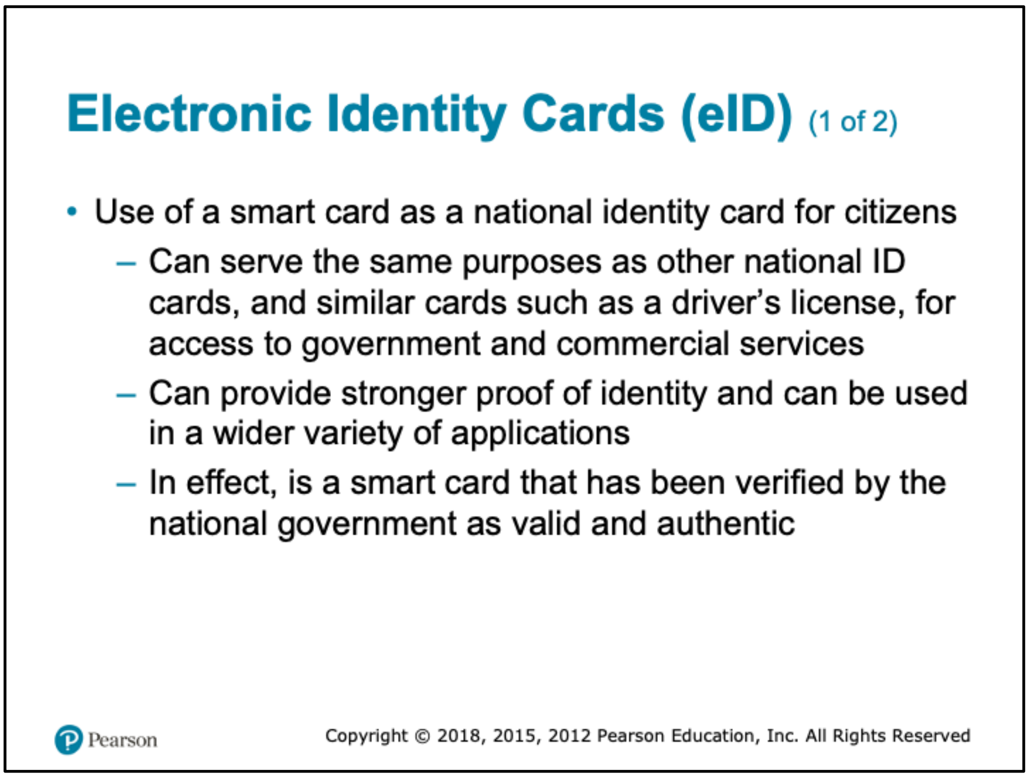
A smart card contains within it an entire microprocessor, including processor, memory, and I/O ports. Some versions incorporate a special co-processing circuit for cryptographic operation to speed the task of encoding and decoding messages or generating digital signatures to validate the information 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 talk time remaining. Random access memory (RAM) holds temporary data generated when applications are executed.



Smart Cards (2 of 2)

- Typically include three types of memory:
  - Read-only memory (ROM)
    - Stores data that does not change during the card's life
  - Electrically erasable programmable ROM (EEPROM)
    - Holds application data and programs
  - Random access memory (RAM)
    - Holds temporary data generated when applications are executed



Electronic Identity Cards (eID) (1 of 2)

- Use of a smart card as a national identity card for citizens
  - 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
  - Can provide stronger proof of identity and can be used in a wider variety of applications
  - In effect, is a smart card that has been verified by the national government as valid and authentic

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.

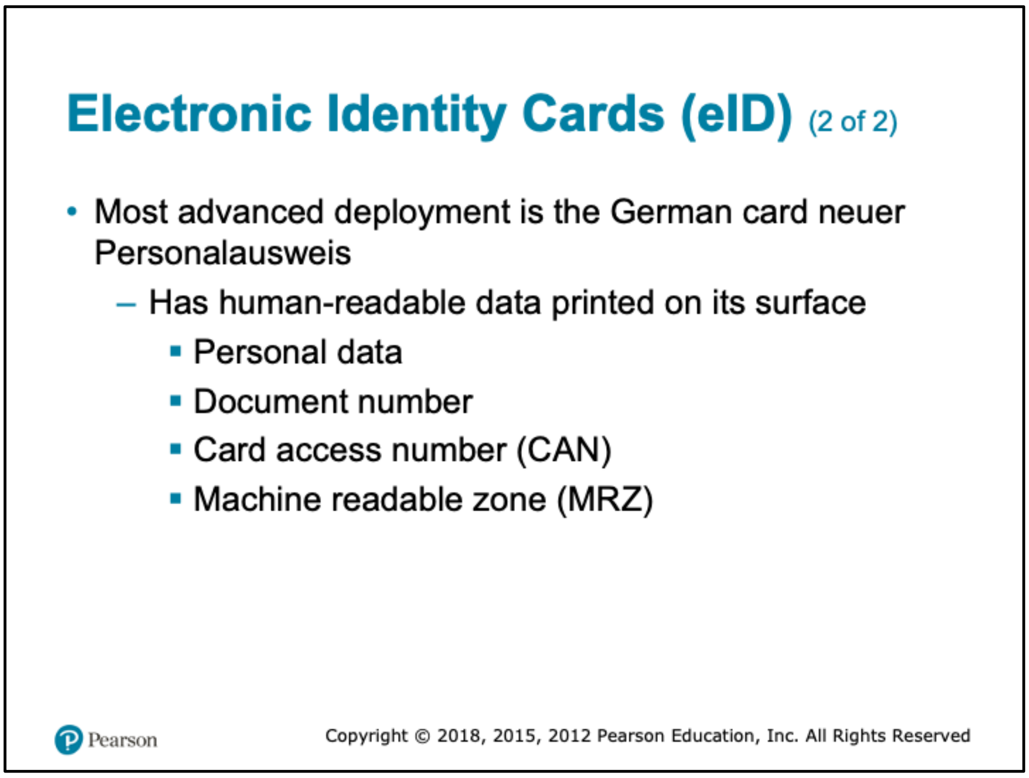
One of the most recent and most advanced eID deployments is the German eID card *neuer Personalausweis* [POLL12]. The card has human-readable data printed on its surface, including the following:

• Personal data: Such as name, date of birth, and address; this is the type of printed information found on passports and driver’s 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.

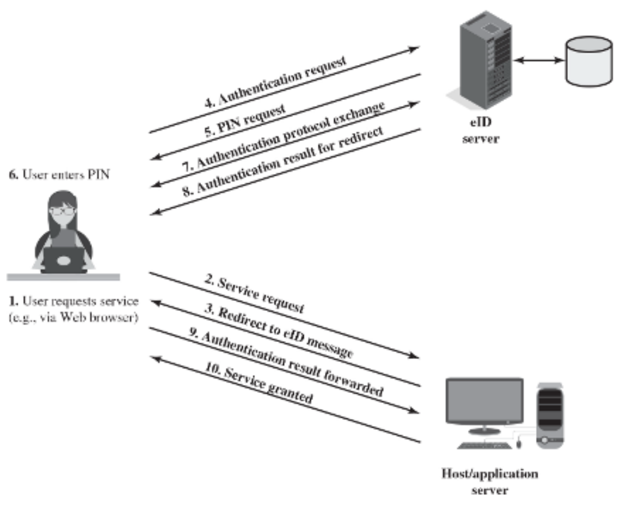
• Machine readable zone (MRZ): Three lines of human- and machine-readable text on the back of the card. This may also be used as a password.



Electronic Identity Cards (eID) (2 of 2)

- Most advanced deployment is the German card neuer Personalausweis
  - Has human-readable data printed on its surface
    - Personal data
    - Document number
    - Card access number (CAN)
    - Machine readable zone (MRZ)

Figure 3.7 User Authentication with eID



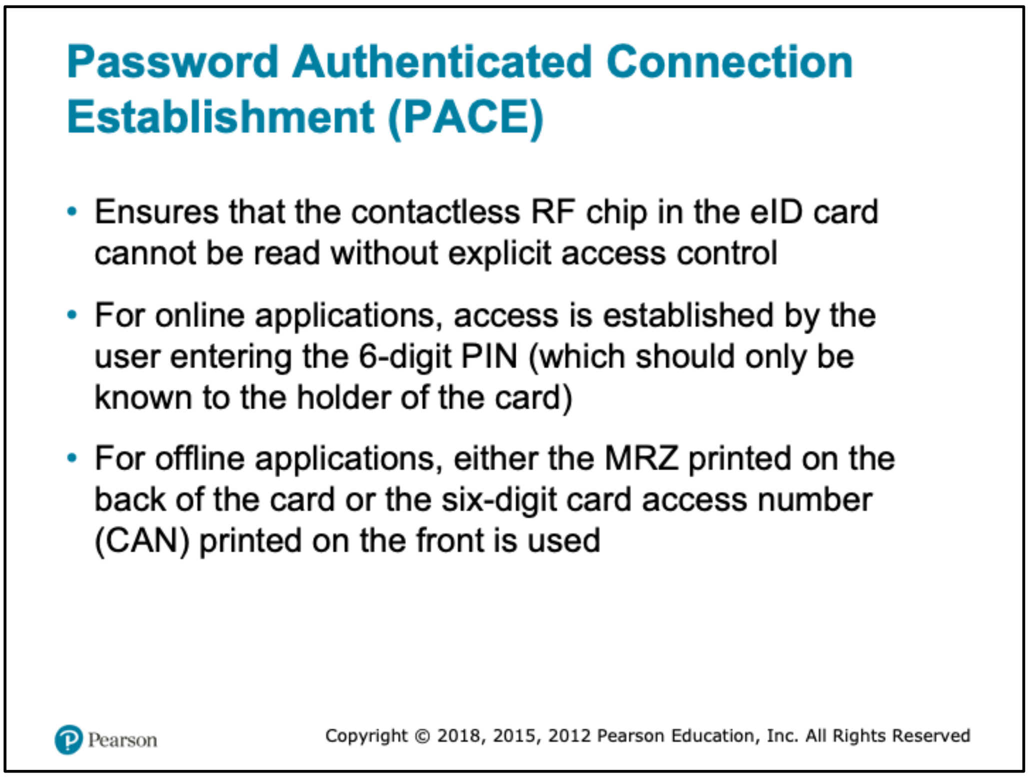
4. Authentication request
5. PIN request
7. Authentication protocol exchange
8. Authentication result for redirect
eID server
6. User enters PIN
1. User requests service (e.g., via Web browser)
2. Service request
3. Redirect to eID message
9. Authentication result forwarded
10. Service granted
Host/application server

User authentication is a good example of online use of the eID function. Figure 3.7 illustrates a Web-based scenario. To begin, an eID user visits a Web site and requests a service that requires authentication. The Web site sends back a redirect message that forwards an authentication request to an eID server. The eID server requests that the user enter the PIN number for the eID card. Once the user has correctly entered the PIN, data can be exchanged between the eID card and the terminal reader in encrypted form. The server then engages in an authentication protocol exchange with the microprocessor on the eID card. If the user is authenticated the results are sent back to the user system to be redirected to the Web server application.

For the preceding scenario, the appropriate software and hardware are required on the user system. Software on the main user system includes functionality for requesting and accepting the PIN number and for message redirection. The hardware required is an eID card reader. The card reader can be an external contact or contactless reader or a contactless reader internal to the user system.

The illustration depicts the exchanges between the user and the host server and that between the user and e I D server. The exchanges are sequenced by numbers as follows. 1. User requests service, example via web browser. 2.

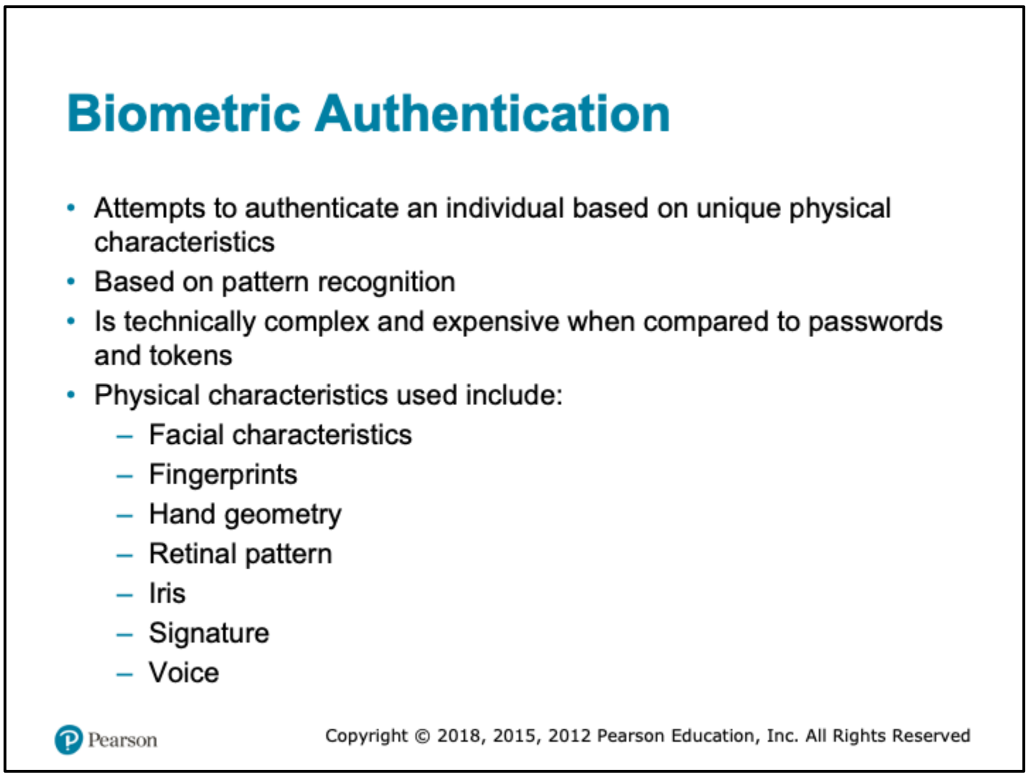
The user sends service request to the host or application server. 3. The host sends redirect to e I D message to the user. 4. The user sends an authentication request to the e I D server. 5. The e I D server sends a PIN request to the user. 6. The User then enters PIN. 7. The authentication protocol exchange takes place between the user and the e I D server. 8. The e I D server then sends an authentication result for redirect. 9. The authentication result is forwarded from the user to the host server. 10. The host server then grants service to the user.



Password Authenticated Connection Establishment (PACE)

- Ensures that the contactless RF chip in the eID card cannot be read without explicit access control
- For online applications, access is established by the user entering the 6-digit PIN (which should only be known to the holder of the card)
- For offline applications, either the MRZ printed on the back of the card or the six-digit card access number (CAN) printed on the front is used

Password Authenticated Connection Establishment (PACE) ensures that the contactless RF chip in the eID card cannot be read without explicit access control. For online applications, access to the card is established by the user entering the 6-digit PIN, which should only be known to the holder of the card. For offline applications, either the MRZ printed on the back of the card or the six-digit card access number (CAN) printed on the front is used.



Biometric Authentication

- Attempts to authenticate an individual based on unique physical characteristics
- Based on pattern recognition
- Is technically complex and expensive when compared to passwords and tokens
- Physical characteristics used include:
  - Facial characteristics
  - Fingerprints
  - Hand geometry
  - Retinal pattern
  - Iris
  - Signature
  - Voice

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 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.

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

• 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.

• 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.

• Hand geometry: Hand geometry systems identify features of the hand, including shape, and lengths and widths of fingers.

• 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.

• Iris: Another unique physical characteristic is the detailed structure of the iris.

• 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.

• 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.

Figure 3.8 Cost Versus Accuracy of Various Biometric Characteristics in User Authentication Schemes

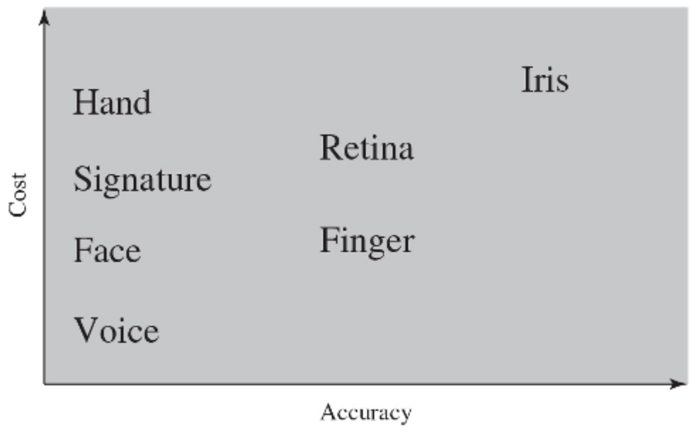


Figure 3.8 gives a rough indication of the relative cost and accuracy of these biometric measures. The concept of accuracy does not apply to user authentication schemes using smart cards or passwords. For example, if a user enters a password, it either matches exactly the password expected for that user or not. In the case of biometric parameters, the system instead must determine how closely a presented biometric characteristic matches a stored characteristic. Before elaborating on the concept of biometric accuracy, we need to have a general idea of how biometric systems work.

The accuracy for Hand, Signature, Face, and voice is low, while the cost from voice to hand gradually increases. The accuracy for Retina and Finger is comparatively high, while the cost from Retina is higher than that of the Finger. The accuracy, as well as cost for Iris, is the highest.

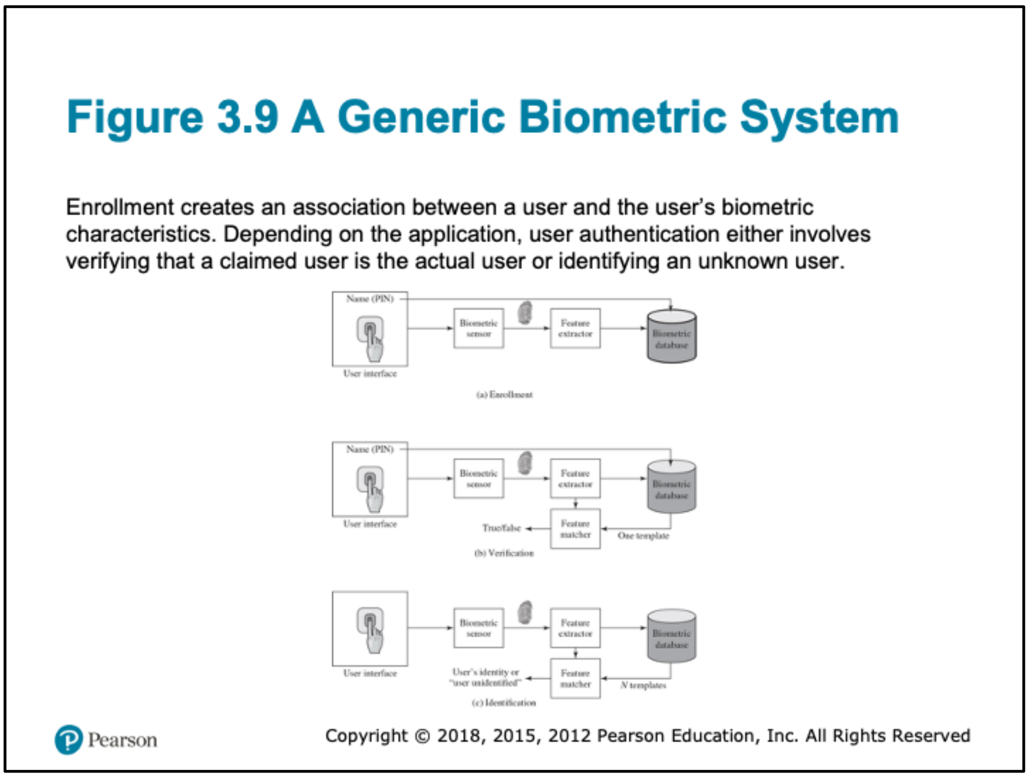


Figure 3.9 A Generic Biometric System

Enrollment creates an association between a user and the user's biometric characteristics. Depending on the application, user authentication either involves verifying that a claimed user is the actual user or identifying an unknown user.

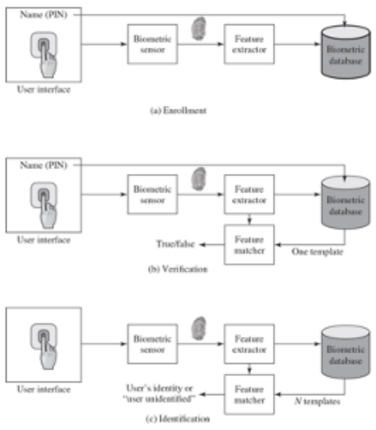


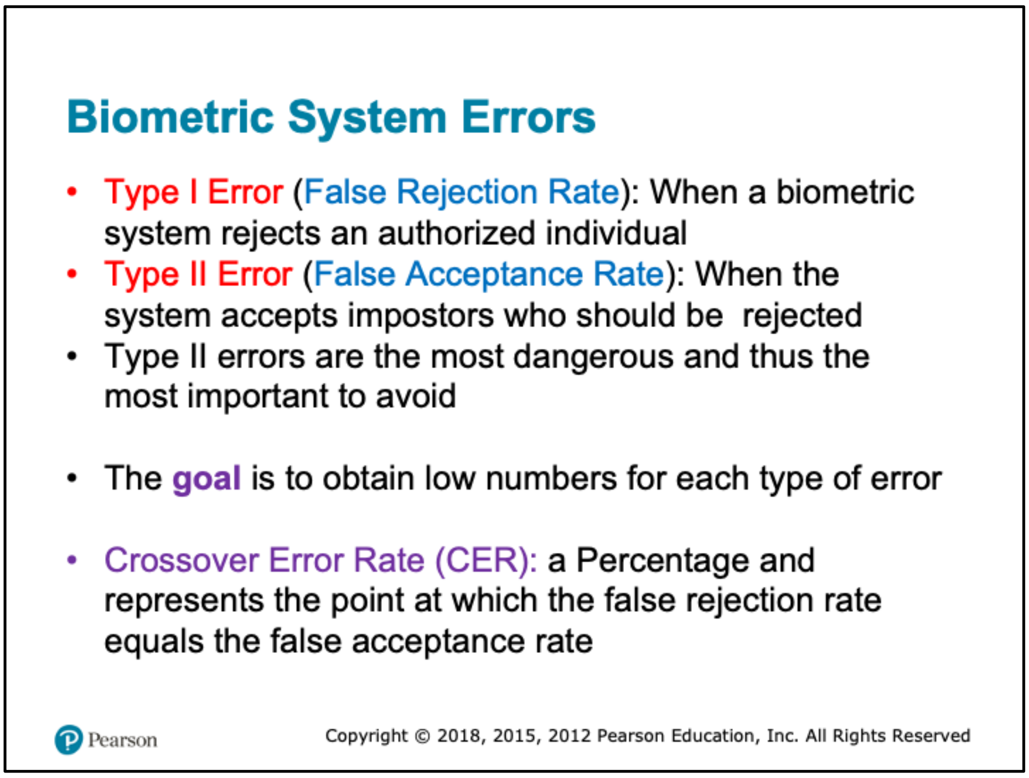
Figure 3.9 illustrates the operation of a biometric system. Each individual who is to be included in the database of authorized users must first be enrolled in the system. This is analogous to assigning a password to a user. For a biometric system, the user presents a name and, typically, some type of 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 and 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.

Illustration a, depicts the enrollment. The user interface, which includes name, PIN, is given as input directly to the biometric database, and through the biometric sensor and feature extractor. Illustration b, depicts the verification. The user interface, which includes name, PIN, is given as input directly to the biometric database, and through the biometric sensor and feature extractor. The biometric database output, one template, is fed to the feature matcher. The feature extractor output is also fed to the feature matcher. The output is either true or false. Illustration c, depicts the identification. The user interface, which includes name, PIN, is given as input to the biometric database, through the biometric sensor and feature extractor. The biometric database output, N templates, is fed to the feature matcher. The feature extractor output is also fed to the feature matcher. The output is either user’s identity or user identified.



Biometric System Errors

- Type I Error (False Rejection Rate): When a biometric system rejects an authorized individual
- Type II Error (False Acceptance Rate): When the system accepts impostors who should be rejected
- Type II errors are the most dangerous and thus the most important to avoid
- The **goal** is to obtain low numbers for each type of error
- Crossover Error Rate (CER): a Percentage and represents the point at which the false rejection rate equals the false acceptance rate

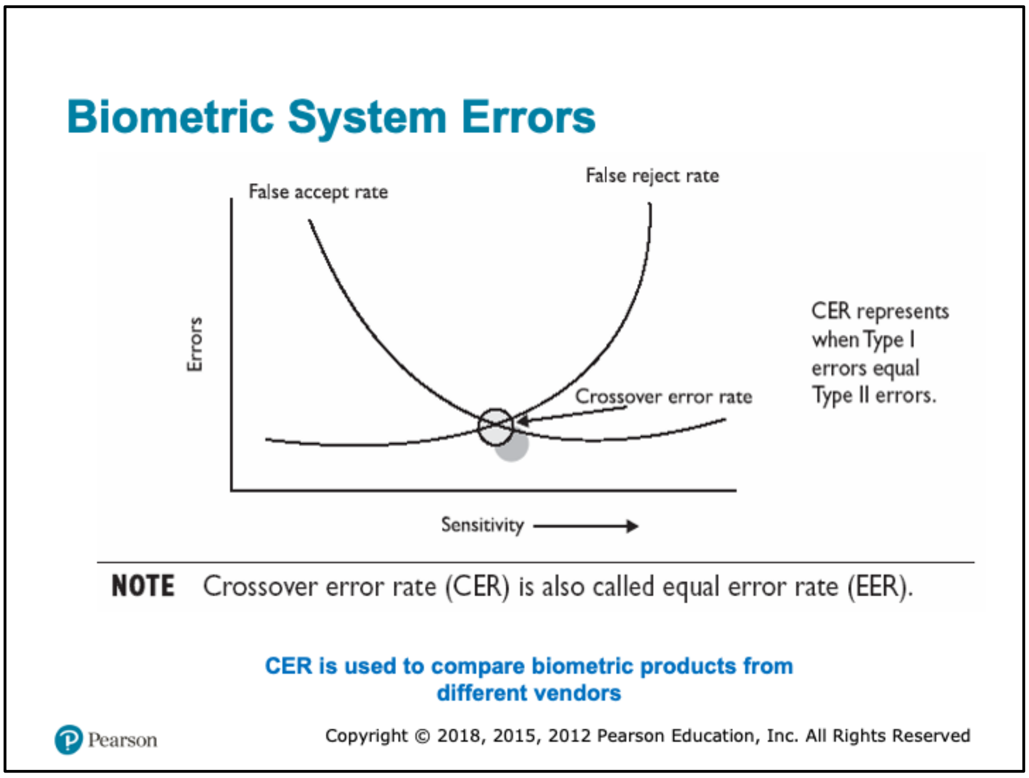
Figure 3.9 illustrates the operation of a biometric system. Each individual who is to be included in the database of authorized users must first be enrolled in the system. This is analogous to assigning a password to a user. For a biometric system, the user presents a name and, typically, some type of 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 and 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.

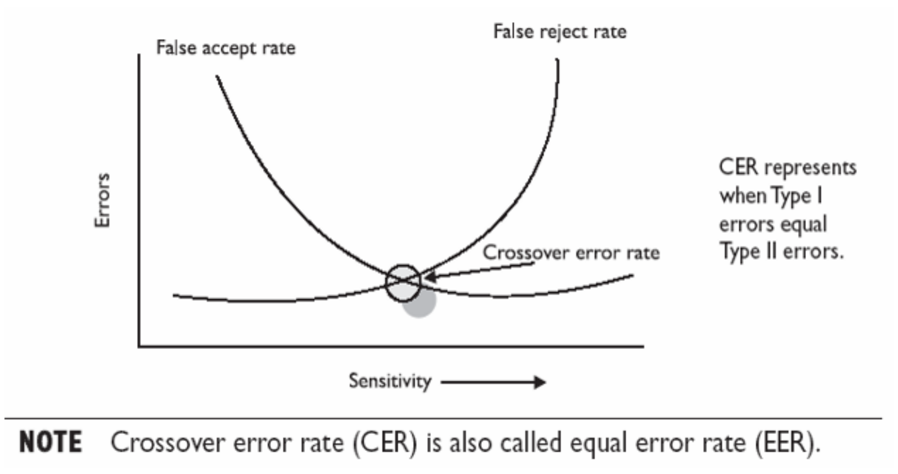
For an identification system, the individual uses the biometric sensor but

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.

Illustration a, depicts the enrollment. The user interface, which includes name, PIN, is given as input directly to the biometric database, and through the biometric sensor and feature extractor. Illustration b, depicts the verification. The user interface, which includes name, PIN, is given as input directly to the biometric database, and through the biometric sensor and feature extractor. The biometric database output, one template, is fed to the feature matcher. The feature extractor output is also fed to the feature matcher. The output is either true or false. Illustration c, depicts the identification. The user interface, which includes name, PIN, is given as input to the biometric database, through the biometric sensor and feature extractor. The biometric database output, N templates, is fed to the feature matcher. The feature extractor output is also fed to the feature matcher. The output is either user’s identity or user identified.



Biometric System Errors



NOTE Crossover error rate (CER) is also called equal error rate (EER).

CER is used to compare biometric products from different vendors

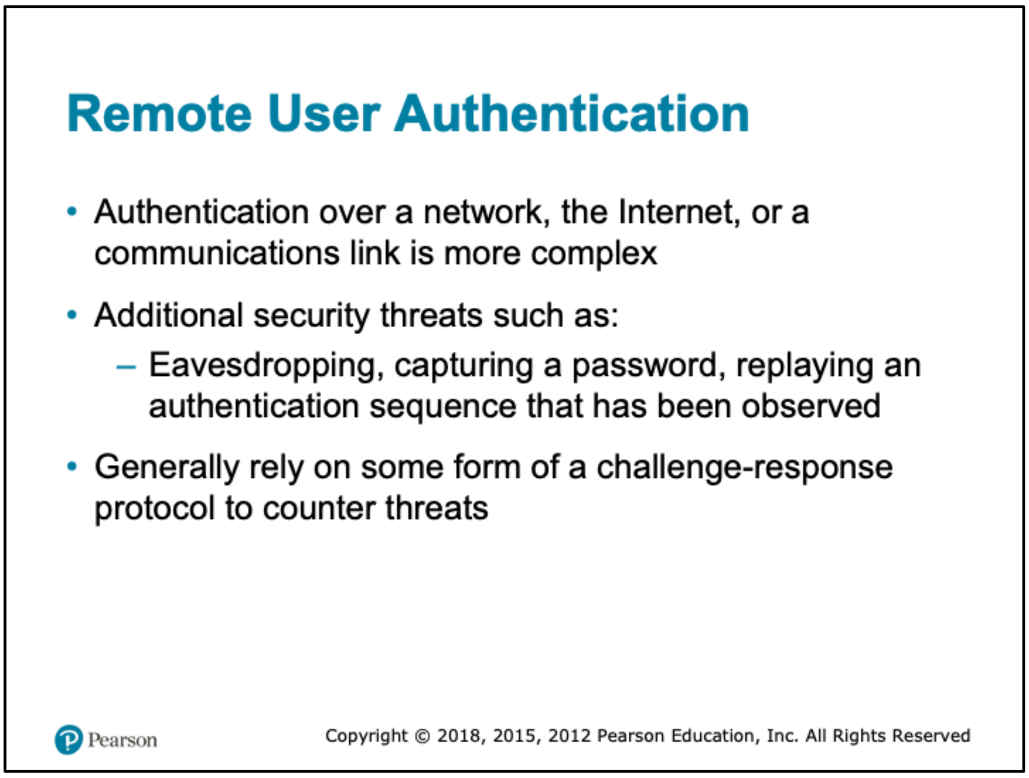
Figure 3.9 illustrates the operation of a biometric system. Each individual who is to be included in the database of authorized users must first be enrolled in the system. This is analogous to assigning a password to a user. For a biometric system, the user presents a name and, typically, some type of 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 and 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.

Illustration a, depicts the enrollment. The user interface, which includes name, PIN, is given as input directly to the biometric database, and through the biometric sensor and feature extractor. Illustration b, depicts the verification. The user interface, which includes name, PIN, is given as input directly to the biometric database, and through the biometric sensor and feature extractor. The biometric database output, one template, is fed to the feature matcher. The feature extractor output is also fed to the feature matcher. The output is either true or false. Illustration c, depicts the identification. The user interface, which includes name, PIN, is given as input to the biometric database, through the biometric sensor and feature extractor. The biometric database output, N templates, is fed to the feature matcher. The feature extractor output is also fed to the feature matcher. The output is either user's identity or user identified.



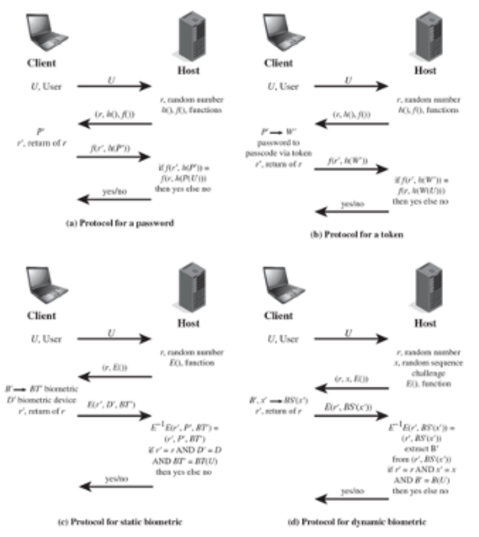
Remote User Authentication

- Authentication over a network, the Internet, or a communications link is more complex
- Additional security threats such as:
  - Eavesdropping, capturing a password, replaying an authentication sequence that has been observed
- Generally rely on some form of a challenge-response protocol to counter threats

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. In this section, we present the basic elements of such protocols for each of the types of authenticators discussed in this chapter.

Figure 3.13 Basic Challenge-Response Protocols for Remote User Authentication



Pearson

Figure 3.13a provides a simple example of a challenge-response protocol for authentication via password. Actual protocols are more complex, such as Kerberos, discussed in Chapter 23. In this example, a user first transmits his or her identity to the remote host. 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. 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 that the response consists of the hash function of the user's password combined with the random number using the function f.

The host stores the hash function of each register user's password, depicted as h(P(U)) for user U. When the response arrives, the host compares the incoming f(r', h(P')) 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. As discussed in Section 3.2, 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.

Figure 3.13b provides a simple example of a token protocol for authentication. As before, a user first transmits his or her identity to the remote host. The host returns a random number and the identifiers of functions f() and h() to be used in the response. 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')). 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.

Figure 3.13c is an example of a user authentication protocol using a static biometric. As before, the user transmits an ID to the host, which responds with a random number r and, in this case, the identifier for an encryption E(). 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. 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.

Figure 3.13d 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. 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') . The client side encrypts the biometric signal and the random number. 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.

Illustration a, depicts the protocol for password between a client and a host as follows. A right arrow from client, User, U, labeled, U extends to the host. A left arrow from host, r, random number, h of, f of, functions labeled, (r, h of, f of) extends to client, P prime, r prime, return of r. A right arrow from client, labeled, f of r prime, h of P prime extends to, if f of start expression r prime, h of P prime end expression = f of start expression r, h of P of U end expression, then yes else no. A left arrow labeled, yes or no extends from host to client. Illustration b depicts the protocol for a token as follows. A right arrow from client, User, U, labeled, U extends to the host. A left arrow from host, r, random number, h of, f of, functions labeled, (r, h of, f of) extends to client, P prime approaches W prime, password to passcode via token r prime, return of r. A right arrow from client, labeled, f of r prime, h of W prime extends to, if f of start expression r prime, h of W prime end expression = f of start expression r, h of W of U end expression, then yes else no. A left arrow labeled, yes or no extends from host to client. Illustration c depicts the protocol for static biometric as follows. A right arrow from client, User, U, labeled, U extends to the host. A left arrow from host, r, random number, E of, function labeled, (r, E of) extends to client, B prime approaches B T prime, biometric, D prime biometric device, r prime, return of r. A right arrow from client, labeled, E of start expression r prime, D prime, B T prime end expression extends to, E to the negative first power, E of start expression r prime, P prime, B T prime end expression = (r prime, P prime, B T prime), if r prime = r AND D prime = D AND B T = B T of U, then yes else no. A left arrow labeled, yes or no extends from host to client. Illustration d depicts the protocol for dynamic biometric as follows. A right arrow from client, User, U, labeled, U extends to the host. A left arrow from host, r, random number, x, random sequence challenge, E of, function labeled, (r, x, E of) extends to client, B prime, x peime approaches B S prime of x prime, r prime, return of r. A right arrow from client, labeled, E of start expression r prime, B S prime of x prime end expression extends to, E to the negative first power, E of start expression r prime, B S prime of x prime end expression = (r prime, B S prime of x prime), extract B prime from (r prime, B S prime of x prime, if r prime = r AND x prime = x AND B prime = B of U, then yes else no. A left arrow labeled, yes or no extends from host to client.

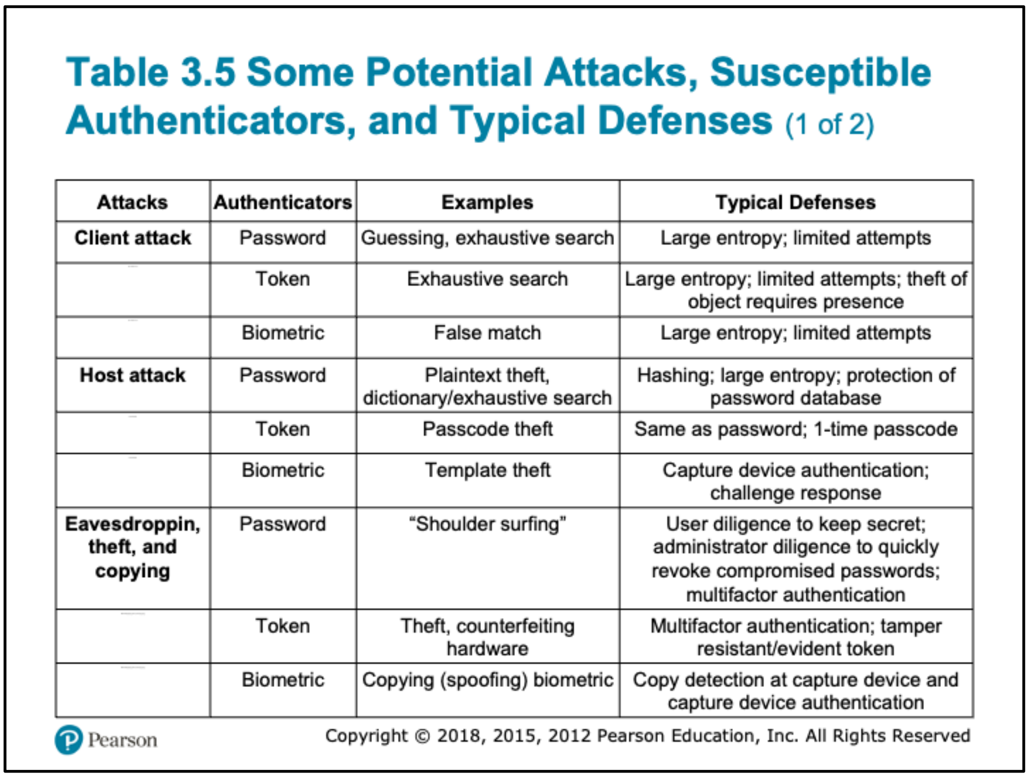


Table 3.5 Some Potential Attacks, Susceptible Authenticators, and Typical Defenses (1 of 2)

Attacks	Authenticators	Examples	Typical Defenses
Client attack	Password	Guessing, exhaustive search	Large entropy; limited attempts
	Token	Exhaustive search	Large entropy; limited attempts; theft of object requires presence
	Biometric	False match	Large entropy; limited attempts
Host attack	Password	Plaintext theft, dictionary/exhaustive search	Hashing; large entropy; protection of password database
	Token	Passcode theft	Same as password; 1-time passcode
	Biometric	Template theft	Capture device authentication; challenge response
Eavesdroppin, theft, and copying	Password	"Shoulder surfing"	User diligence to keep secret; administrator diligence to quickly revoke compromised passwords; multifactor authentication
	Token	Theft, counterfeiting hardware	Multifactor authentication; tamper resistant/evident token
	Biometric	Copying (spoofing) biometric	Copy detection at capture device and capture device authentication

Pearson

As with any security service, user authentication, particularly remote user authentication, is subject to a variety of attacks. Table 3.5, from [OGOR03], summarizes the principal attacks on user authentication, broken down by type of authenticator. Much of the table is self-explanatory. In this section, we expand on some of the table's entries.

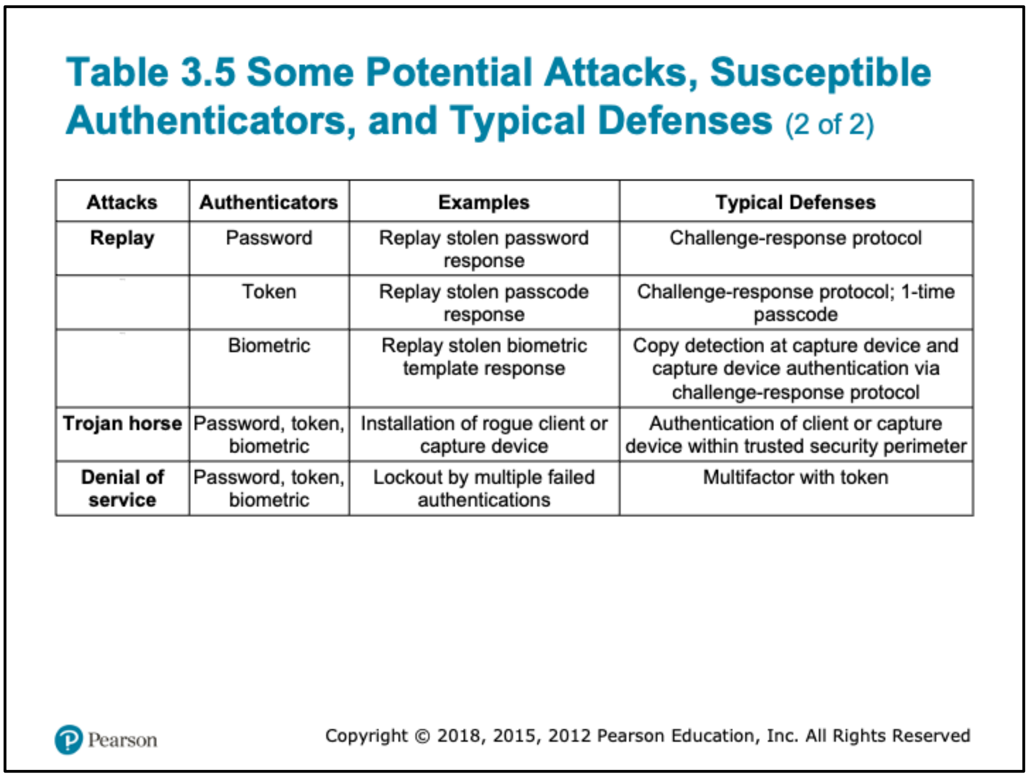
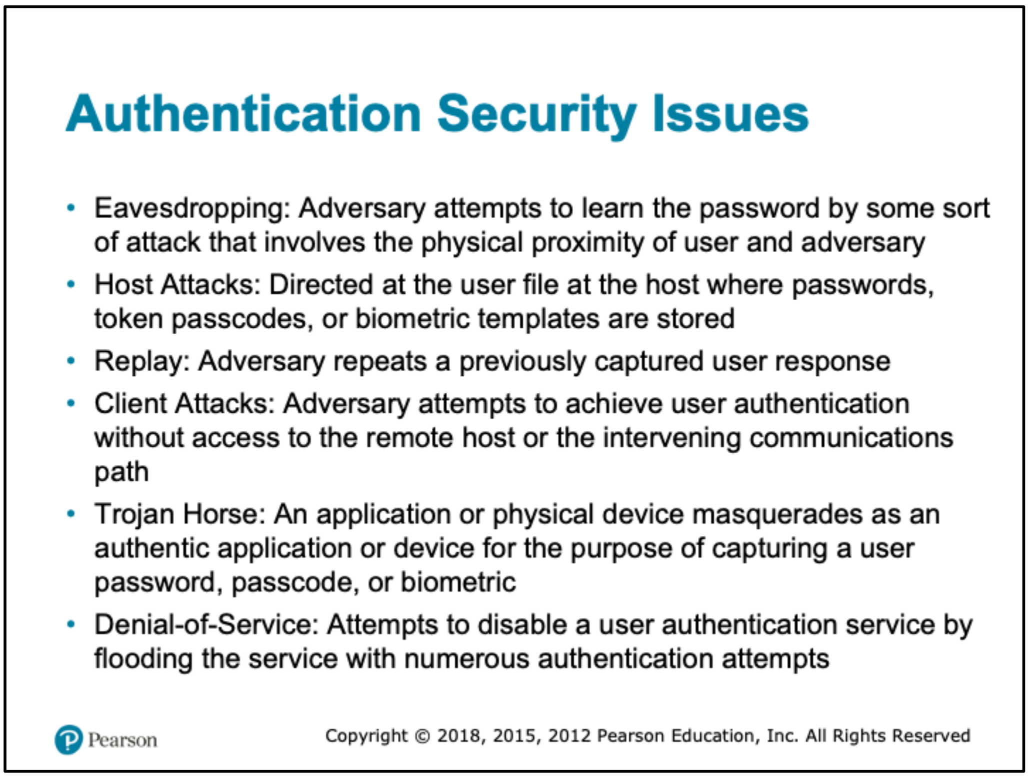


Table 3.5 Some Potential Attacks, Susceptible Authenticators, and Typical Defenses (2 of 2)

Attacks	Authenticators	Examples	Typical Defenses
Replay	Password	Replay stolen password response	Challenge-response protocol
	Token	Replay stolen passcode response	Challenge-response protocol; 1-time passcode
	Biometric	Replay stolen biometric template response	Copy detection at capture device and capture device authentication via challenge-response protocol
Trojan horse	Password, token, biometric	Installation of rogue client or capture device	Authentication of client or capture device within trusted security perimeter
Denial of service	Password, token, biometric	Lockout by multiple failed authentications	Multifactor with token

Pearson



Authentication Security Issues

- Eavesdropping: Adversary attempts to learn the password by some sort of attack that involves the physical proximity of user and adversary
- Host Attacks: Directed at the user file at the host where passwords, token passcodes, or biometric templates are stored
- Replay: Adversary repeats a previously captured user response
- Client Attacks: Adversary attempts to achieve user authentication without access to the remote host or the intervening communications path
- Trojan Horse: An application or physical device masquerades as an authentic application or device for the purpose of capturing a user password, passcode, or biometric
- Denial-of-Service: Attempts to disable a user authentication service by flooding the service with numerous authentication attempts

Client attacks are those in which an adversary attempts to achieve user authentication without access to the remote host or to the intervening communications path. The adversary attempts to masquerade as a legitimate user. For a password-based system, the adversary may attempt to guess the likely user password. Multiple guesses may be made. At the extreme, the adversary sequences through all possible passwords in an exhaustive attempt to succeed. One way to thwart such an attack is to select a password that is both lengthy and unpredictable. In effect, such a password has large entropy; that is, many bits are required to represent the password. Another countermeasure is to limit the number of attempts that can be made in a given time period from a given source.

A token can generate a high-entropy passcode from a low-entropy PIN or password, thwarting exhaustive searches. The adversary may be able to guess or acquire the PIN or password but must additionally acquire the physical token to succeed.

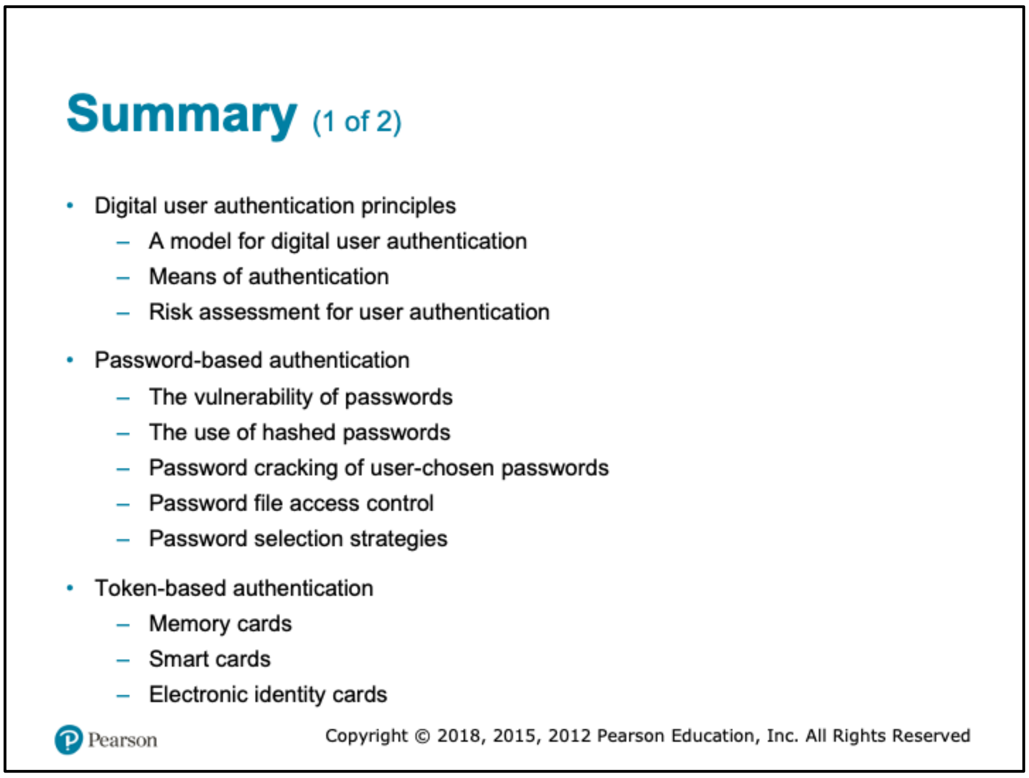
Host attacks are directed at the user file at the host where passwords, token passcodes, or biometric templates are stored. Section 3.2 discusses the security considerations with respect to passwords. For tokens, there is the additional defense of using one-time passcodes, so that passcodes are not stored in a host passcode file. Biometric features of a user are difficult to secure because they are physical features of the user. For a static feature, biometric device authentication adds a measure of protection. For a dynamic feature, a challenge-response protocol enhances security.

Eavesdropping in the context of passwords refers to an adversary's attempt to learn the password by observing the user, finding a written copy of the password, or some similar attack that involves the physical proximity of user and adversary. Another form of eavesdropping is keystroke logging (keylogging), in which malicious hardware or software is installed so that the attacker can capture the user's keystrokes for later analysis. A system that relies on multiple factors (e.g., password plus token or password plus biometric) is resistant to this type of attack. For a token, an analogous threat is theft of the token or physical copying of the token. Again, a multifactor protocol resists this type of attack better than a pure token protocol. The analogous threat for a biometric protocol is **copying** or imitating the biometric parameter so as to generate the desired template. Dynamic biometrics are less susceptible to such attacks. For static biometrics, device authentication is a useful countermeasure.

Replay attacks involve an adversary repeating a previously captured user response. The most common countermeasure to such attacks is the challenge-response protocol.

In a **Trojan horse** attack, an application or physical device masquerades as an authentic application or device for the purpose of capturing a user password, passcode, or biometric. The adversary can then use the captured information to masquerade as a legitimate user. A simple example of this is a rogue bank machine used to capture user ID/password combinations.

A **denial-of-service** attack attempts to disable a user authentication service by flooding the service with numerous authentication attempts. A more selective attack denies service to a specific user by attempting logon until the threshold is reached that causes lockout to this user because of too many logon attempts. A multifactor authentication protocol that includes a token thwarts this attack, because the adversary must first acquire the token.

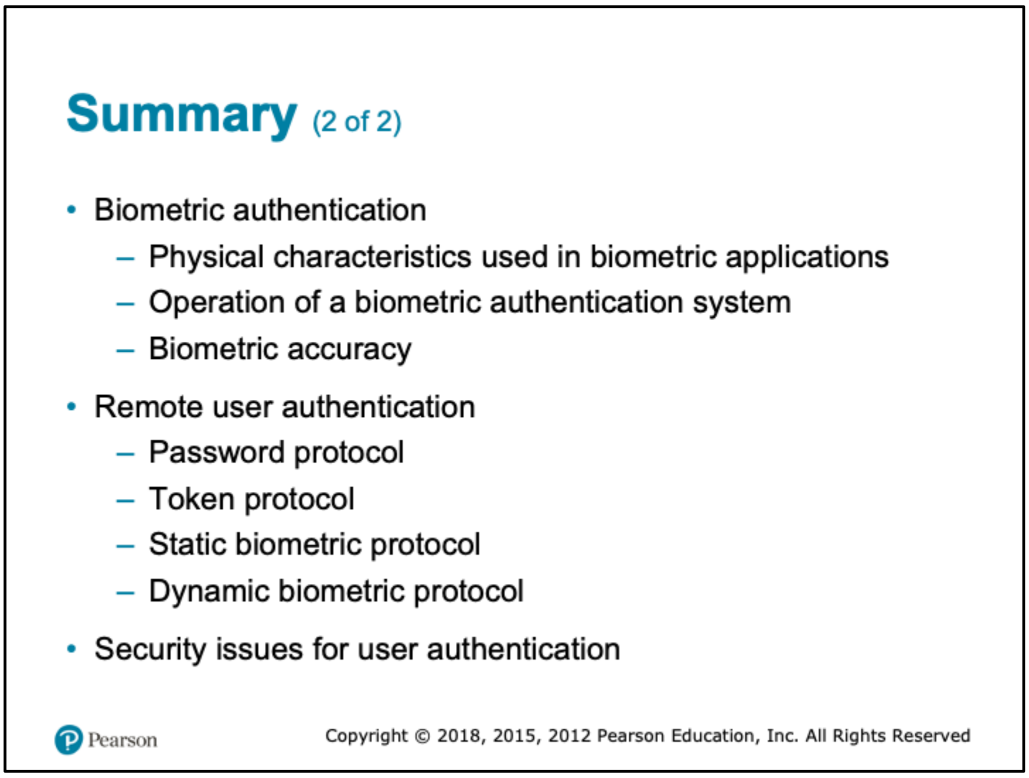


Summary (1 of 2)

- Digital user authentication principles
  - A model for digital user authentication
  - Means of authentication
  - Risk assessment for user authentication
- Password-based authentication
  - The vulnerability of passwords
  - The use of hashed passwords
  - Password cracking of user-chosen passwords
  - Password file access control
  - Password selection strategies
- Token-based authentication
  - Memory cards
  - Smart cards
  - Electronic identity cards

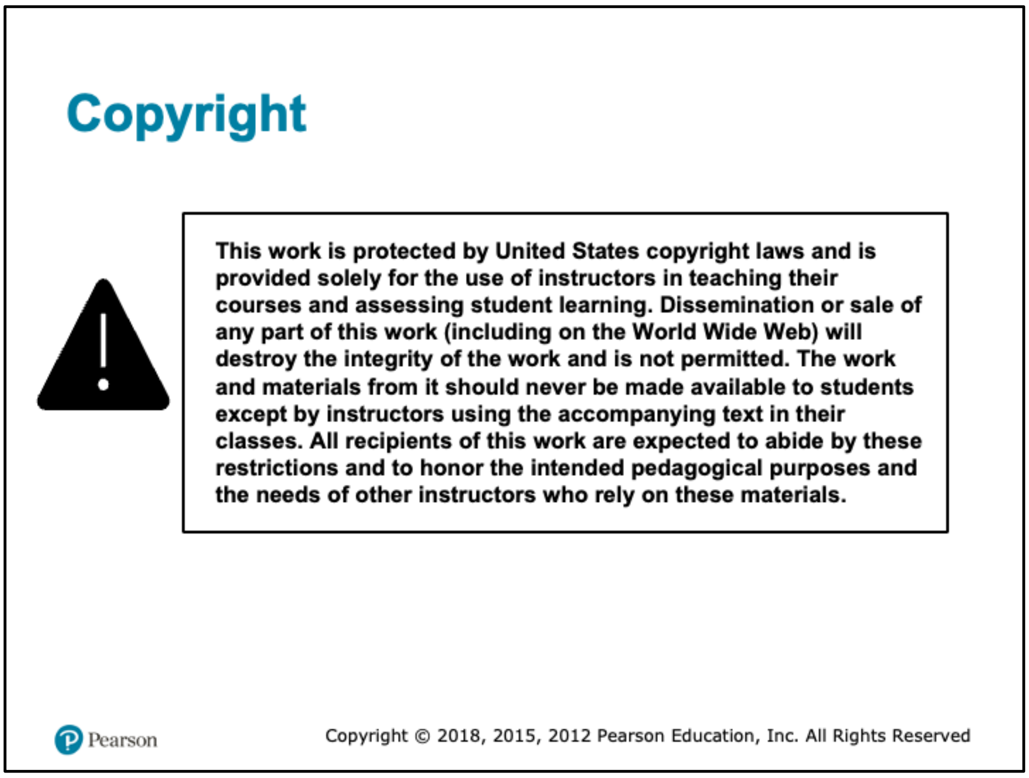
Pearson

Chapter 3 summary.



Summary (2 of 2)

- Biometric authentication
  - Physical characteristics used in biometric applications
  - Operation of a biometric authentication system
  - Biometric accuracy
- Remote user authentication
  - Password protocol
  - Token protocol
  - Static biometric protocol
  - Dynamic biometric protocol
- Security issues for user authentication



Copyright