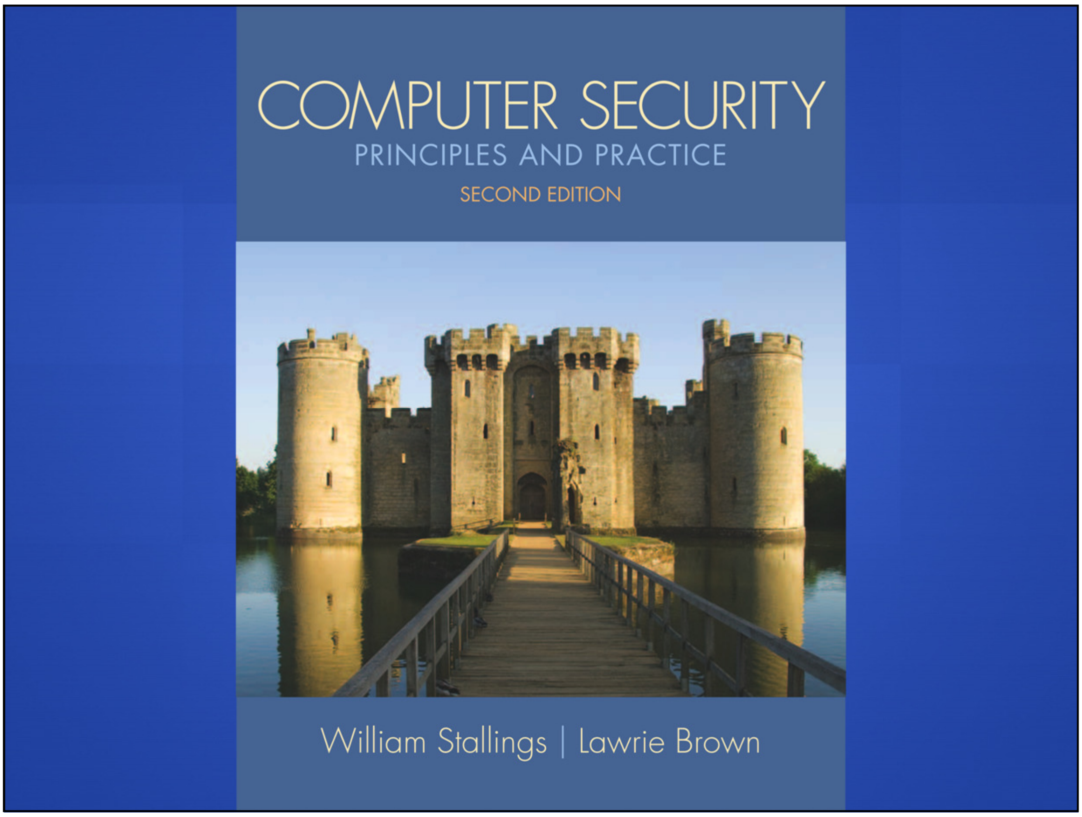
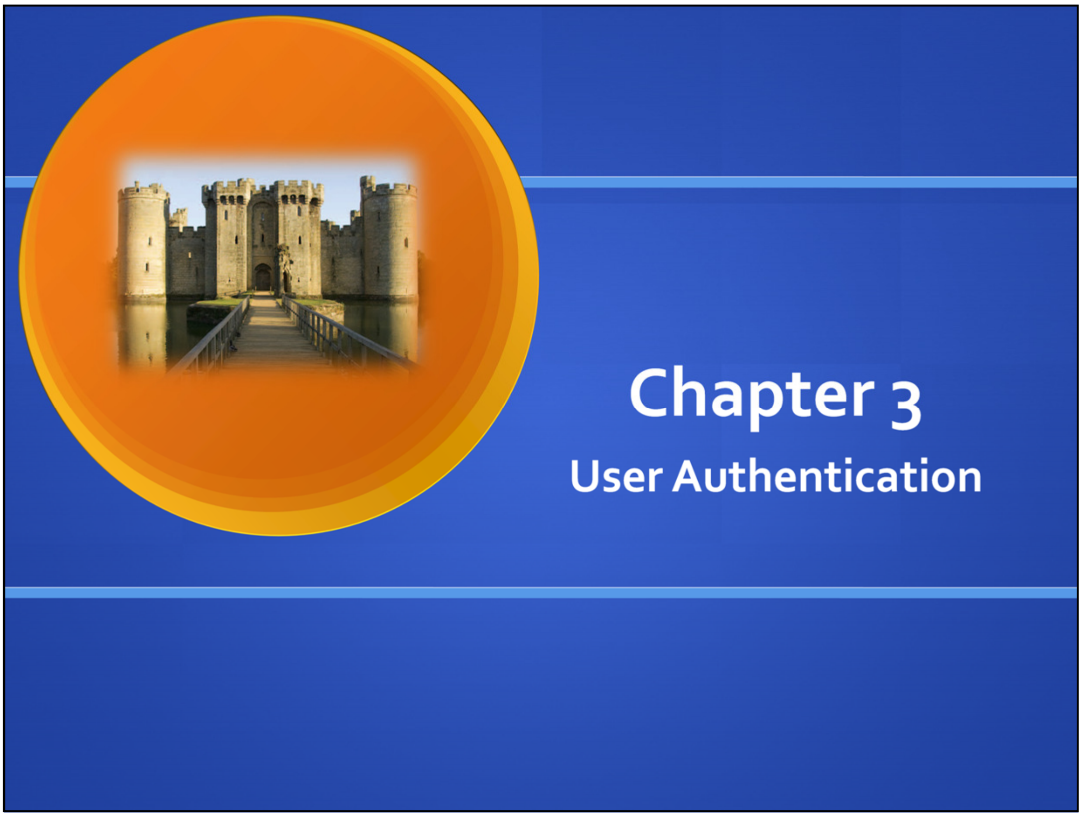


Lecture slides prepared for “Computer Security: Principles and Practice”, 2/e, by William Stallings and Lawrie Brown, Chapter 3 “User Authentication”.



Chapter 3

User Authentication

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.

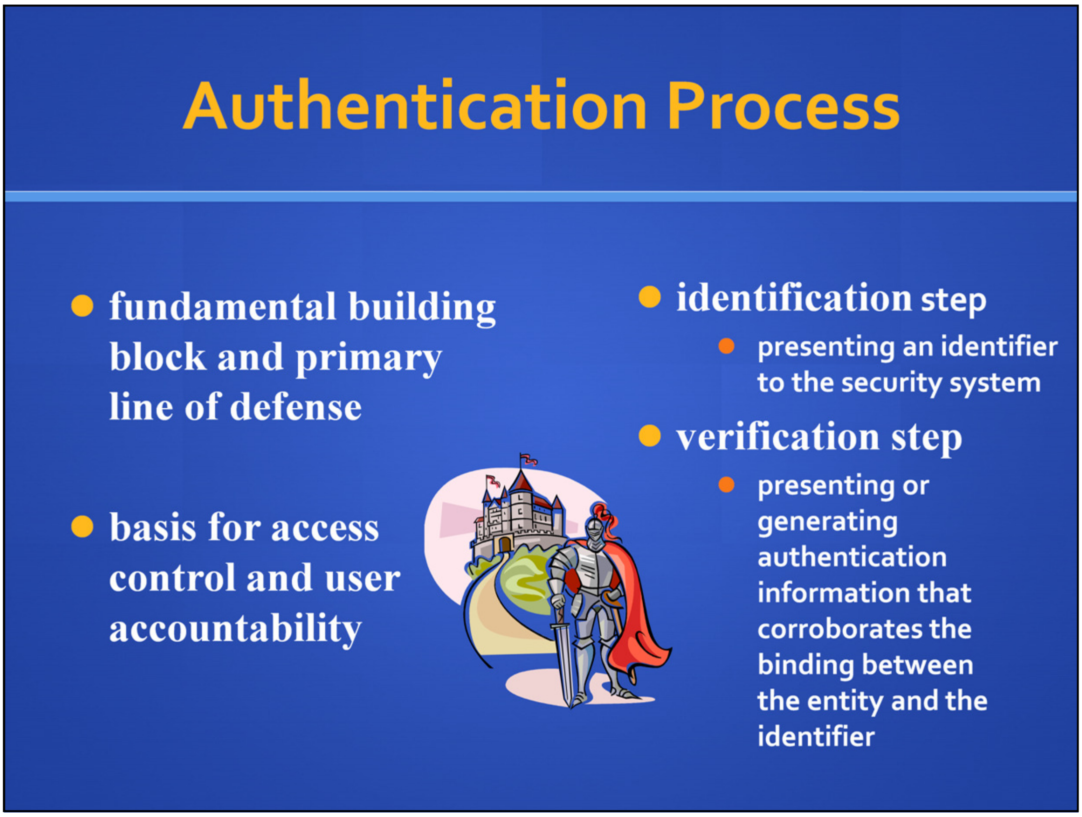
RFC 2828

RFC 2828 defines user authentication as:

“The process of verifying an identity claimed by or for a system entity.”

RFC 2828 defines user authentication as follows:

The process of verifying an identity claimed by or for a system entity.



Authentication Process

- fundamental building block and primary line of defense
- basis for access control and user accountability

- identification step
  - presenting an identifier to the security system
- verification step
  - presenting or generating authentication information that corroborates the binding between the entity and the identifier

An authentication process consists of two steps:

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

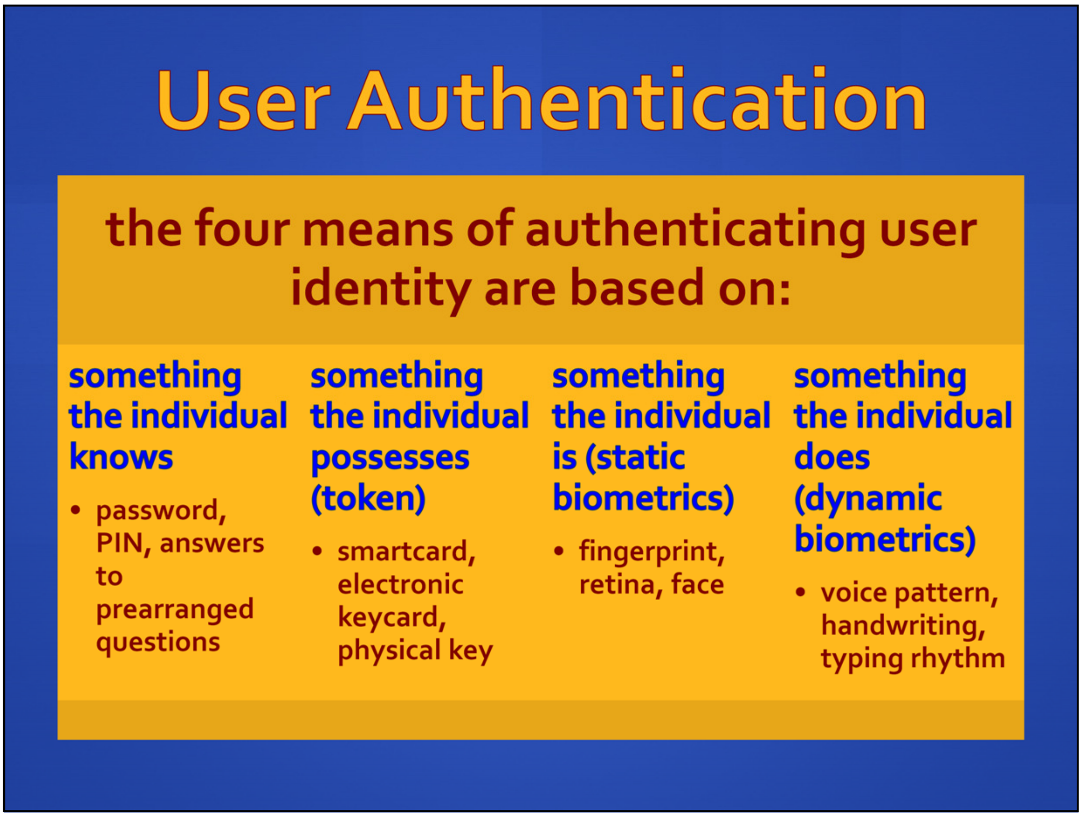
Verification step: Presenting or generating authentication information that corroborates the binding between the entity and the identifier.

For example, user Alice Toklas could have the user identifier ABTOKLAS. This information needs to be stored on any server or computer system that Alice wishes to use and could be known to system administrators and other users. A typical item of authentication information associated with this user ID is a password, which is kept secret (known only to Alice and to the system)2. If no one is able to obtain or guess Alice’s password, then the combination of Alice’s user ID and password enables administrators to set up Alice’s access permissions and audit her activity.

Because Alice’s ID is not secret, system users can send her e-mail, but because her password is secret, no one can pretend to be Alice.

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

Note that user authentication is distinct from message authentication. 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.



User Authentication

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:

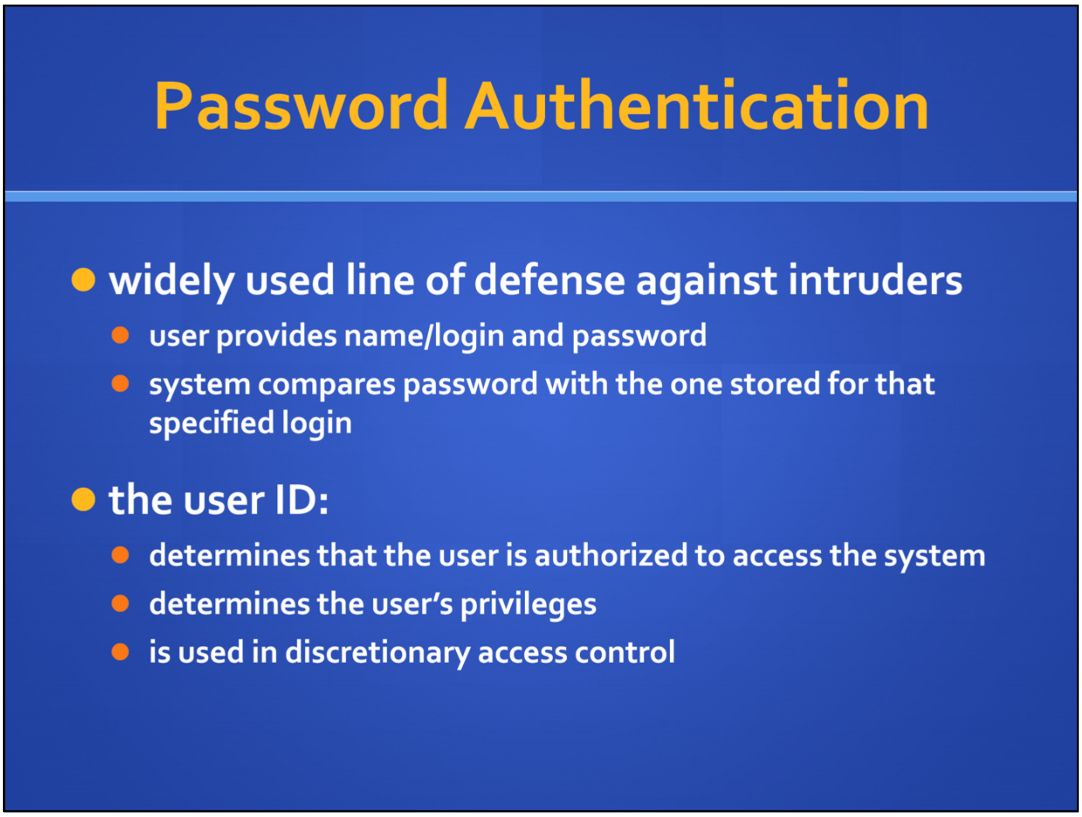
• **Something the individual knows:** Examples includes a password, a personal identification number (PIN), or answers to a prearranged set of questions.

• **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.



Password Authentication

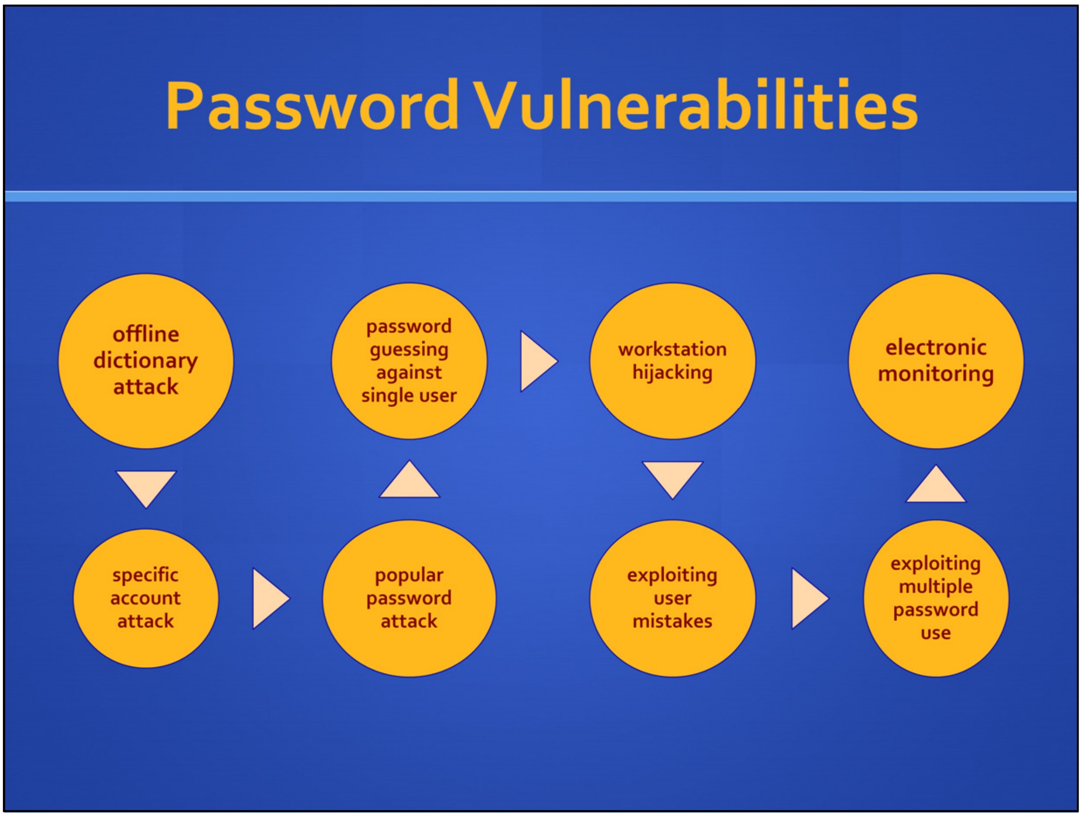
- widely used line of defense against intruders
  - user provides name/login and password
  - system compares password with the one stored for that specified login
- the user ID:
  - determines that the user is authorized to access the system
  - determines the user's privileges
  - is used in discretionary 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

password guessing against single user

workstation hijacking

electronic monitoring

specific account attack

popular password attack

exploiting user mistakes

exploiting multiple password use

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.

- **Specific account attack: The attacker targets a specific account and submits**
password guesses until the correct password is discovered.

- **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.

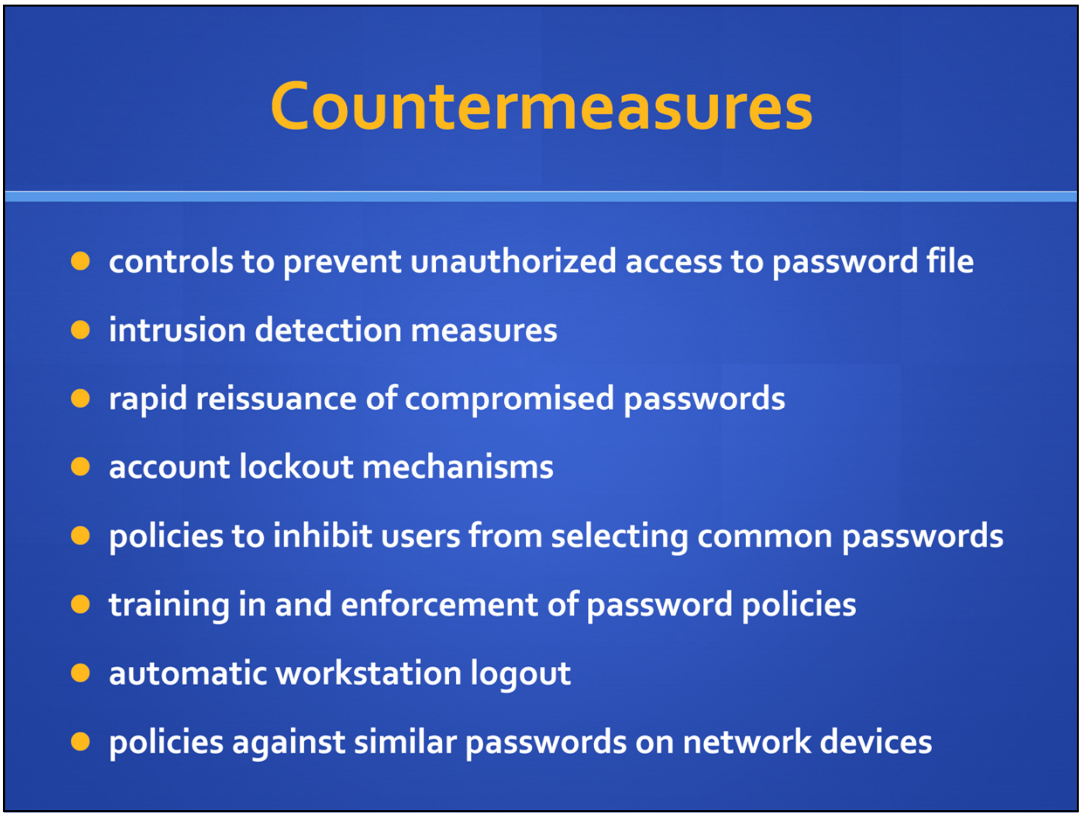
- **Password guessing against single user: The attacker attempts to gain knowledge** about the account holder and system password policies and uses that knowledge to guess the password.

- **Workstation hijacking; The attacker waits until a logged-in workstation is** unattended.

- **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.

- **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.

- **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



Countermeasures

- controls to prevent unauthorized access to password file
- intrusion detection measures
- rapid reissuance of compromised passwords
- account lockout mechanisms
- policies to inhibit users from selecting common passwords
- training in and enforcement of password policies
- automatic workstation logout
- policies against similar passwords on network devices

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, as described subsequently.

• **Offline dictionary attack:** 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 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:** 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.

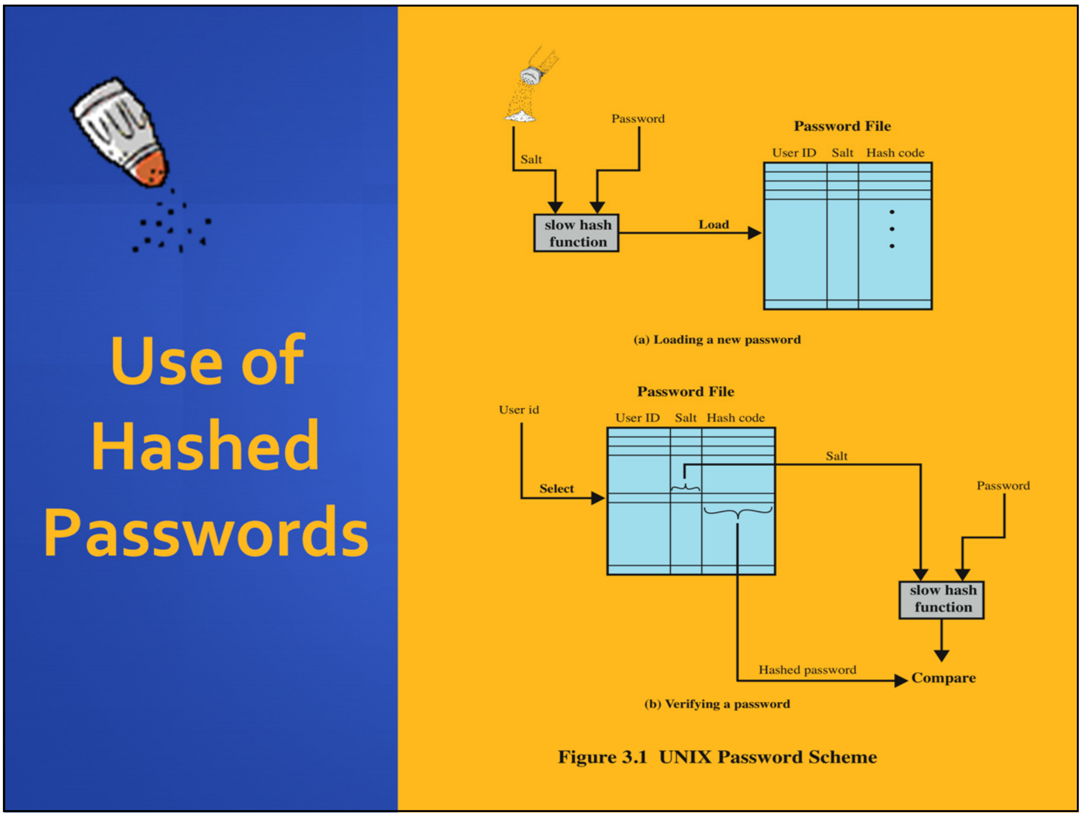
- **Password guessing against single user:** Countermeasures include training in and enforcement of password policies that make passwords difficult to guess. Such policies address the secrecy, minimum length of the password, character set, prohibition against using well-known user identifiers, and length of time before the password must be changed.

- **Workstation hijacking;** The 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:** Countermeasures include user training, intrusion detection, and simpler passwords combined with another authentication mechanism.

- **Exploiting multiple password use.** 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



Use of Hashed Passwords

Figure 3.1 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 3.1a). 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 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 3.1b). 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:

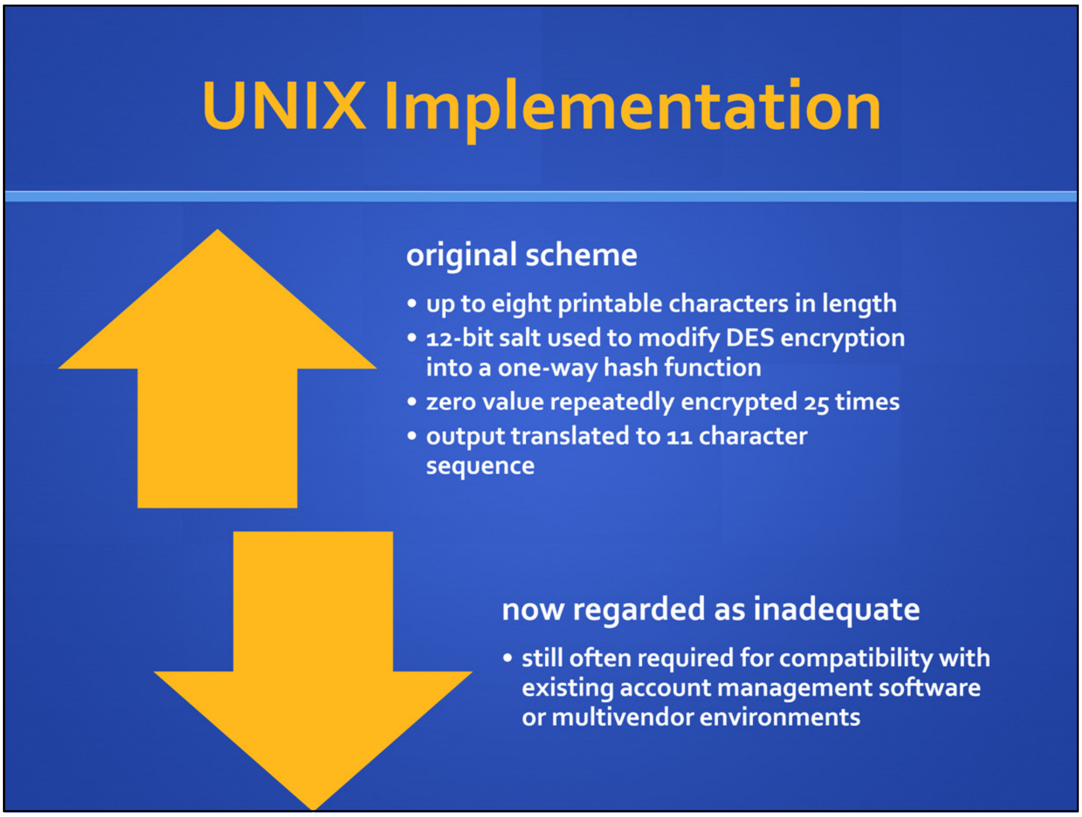
- It prevents duplicate passwords from being visible in the password file. Even if two users choose the same password, those passwords will be assigned different salt values. Hence, the hashed passwords of the two users will differ.

- 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 2^b,* 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.



UNIX Implementation

original scheme

- up to eight printable characters in length
- 12-bit salt used to modify DES encryption into a one-way hash function
- zero value repeatedly encrypted 25 times
- output translated to 11 character sequence

now regarded as inadequate

- still often required for compatibility with existing account management software or multivendor environments

Since the original development of UNIX, most implementations have relied on the following password scheme. Each user selects a password of up to eight printable characters in length. This is converted into a 56-bit value (using 7-bit ASCII) that serves as the key input to an encryption routine. The hash routine, known as crypt(3), is based on DES. A 12-bit salt value is used. The modified DES algorithm is executed with a data input consisting of a 64-bit block of zeros. The output of the algorithm then serves as input for a second encryption. This process is repeated for a total of 25 encryptions. The resulting 64-bit output is then translated into an 11-character sequence. The modification of the DES algorithm converts it into a one-way hash function. The crypt(3) routine is designed to discourage guessing attacks. Software implementations of DES are slow compared to hardware versions, and the use of 25 iterations multiplies the time required by 25.

This particular implementation is now considered woefully inadequate. For example, [PERR03] reports the results of a dictionary attack using a supercomputer.

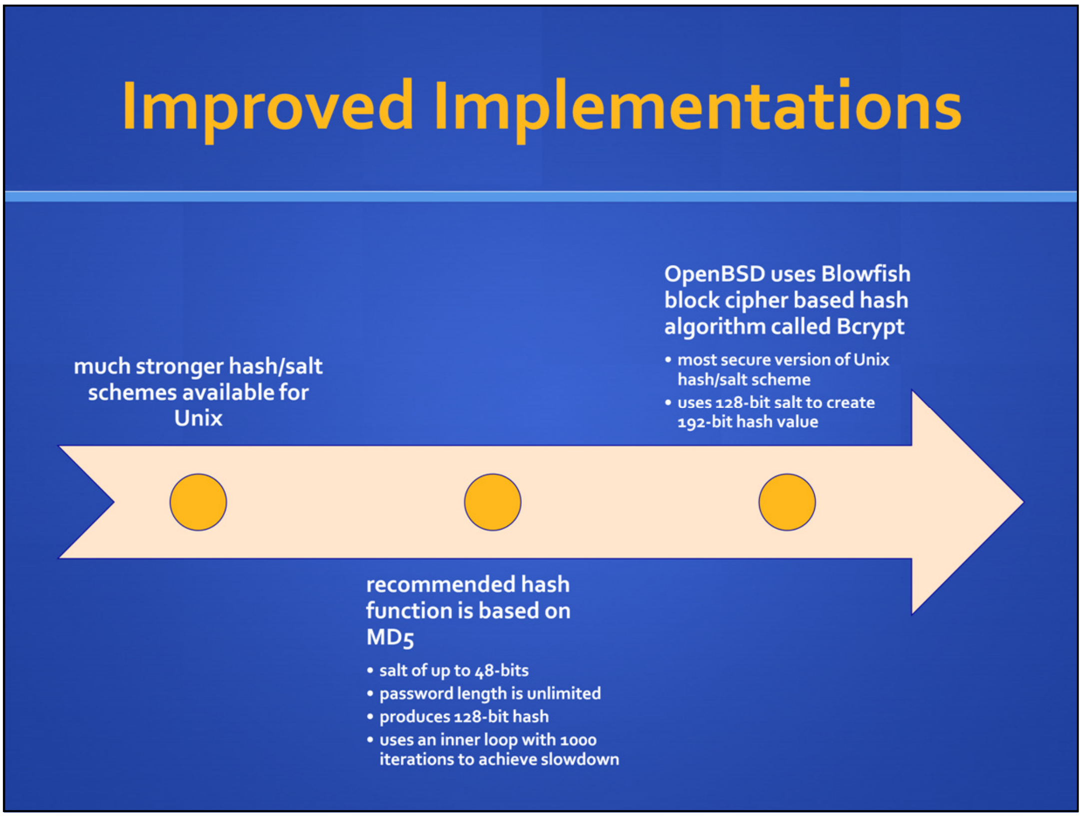
The attack was able to process over 50 million password guesses in about 80 minutes.

Further, the results showed that for about $10,000 anyone should be able to do the

same in a few months using one uniprocessor machine. Despite its known weaknesses,

this UNIX scheme is still often required for compatibility with existing account management

software or in multivendor environments.



Improved Implementations

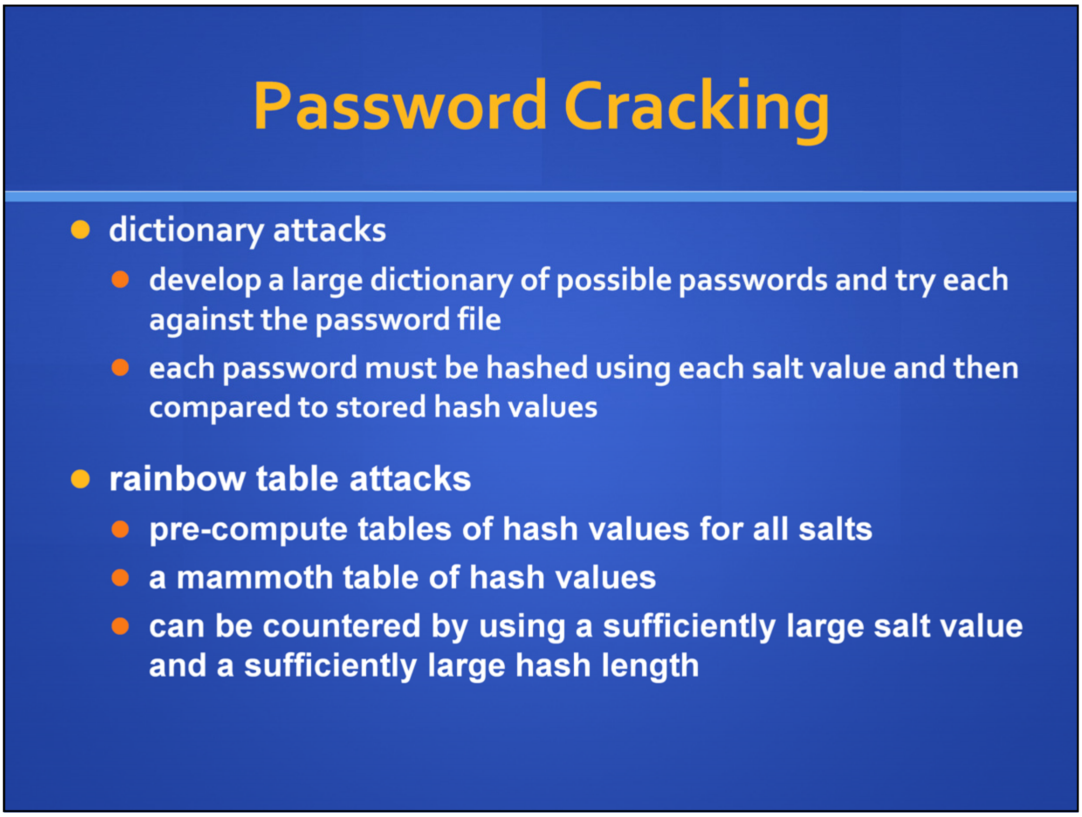
much stronger hash/salt schemes available for Unix

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

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.

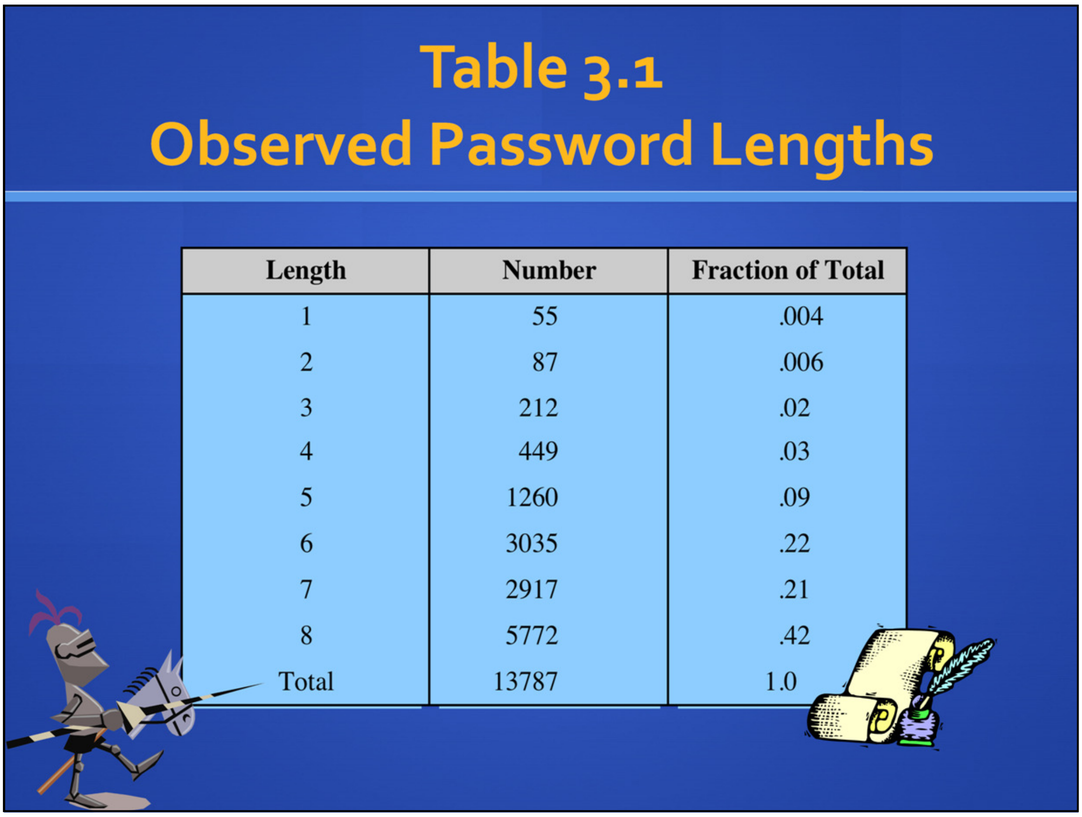


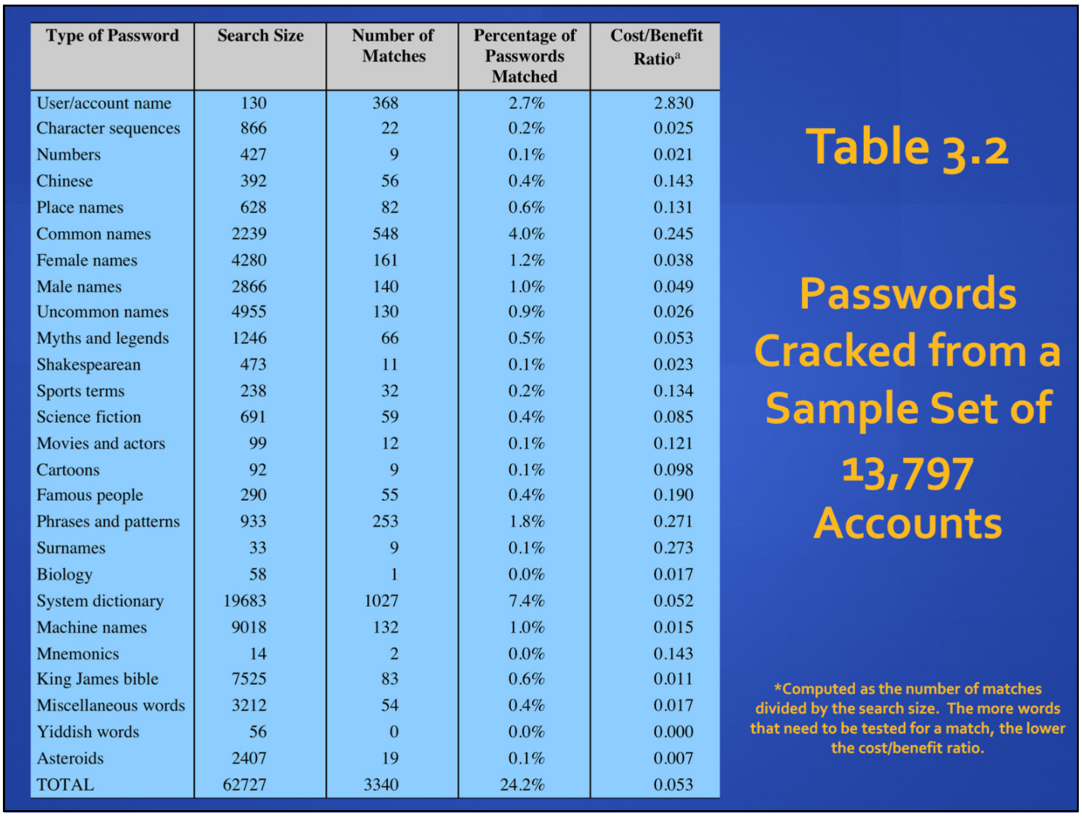
Table 3.1
Observed Password Lengths

Length	Number	Fraction of Total
1	55	.004
2	87	.006
3	212	.02
4	449	.03
5	1260	.09
6	3035	.22
7	2917	.21
8	5772	.42
Total	13787	1.0

Even the stupendous guessing rates referenced in the preceding section do not yet make it feasible for an attacker to use a dumb brute-force technique of trying all possible combinations of characters to discover a password. Instead, password crackers rely on the fact that some people choose easily guessable passwords.

Some users, when permitted to choose their own password, pick one that is absurdly short. The results of one study at Purdue University are shown in Table 3.1. The study observed password change choices on 54 machines, representing approximately 7000 user accounts. Almost 3% of the passwords were three characters or fewer in length. An attacker could begin the attack by exhaustively testing all possible passwords of length 3 or fewer. A simple remedy is for the system to reject any password choice of fewer than, say, six characters or even to require that all passwords be exactly eight characters in length. Most users would not complain about such a restriction.

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.



Type of Password	Search Size	Number of Matches	Percentage of Passwords Matched	Cost/Benefit Ratio[a]
User/account name	130	368	2.7%	2.830
Character sequences	866	22	0.2%	0.025
Numbers	427	9	0.1%	0.021
Chinese	392	56	0.4%	0.143
Place names	628	82	0.6%	0.131
Common names	2239	548	4.0%	0.245
Female names	4280	161	1.2%	0.038
Male names	2866	140	1.0%	0.049
Uncommon names	4955	130	0.9%	0.026
Myths and legends	1246	66	0.5%	0.053
Shakespearean	473	11	0.1%	0.023
Sports terms	238	32	0.2%	0.134
Science fiction	691	59	0.4%	0.085
Movies and actors	99	12	0.1%	0.121
Cartoons	92	9	0.1%	0.098
Famous people	290	55	0.4%	0.190
Phrases and patterns	933	253	1.8%	0.271
Surnames	33	9	0.1%	0.273
Biology	58	1	0.0%	0.017
System dictionary	19683	1027	7.4%	0.052
Machine names	9018	132	1.0%	0.015
Mnemonics	14	2	0.0%	0.143
King James bible	7525	83	0.6%	0.011
Miscellaneous words	3212	54	0.4%	0.017
Yiddish words	56	0	0.0%	0.000
Asteroids	2407	19	0.1%	0.007
TOTAL	62727	3340	24.2%	0.053

Table 3.2

Passwords Cracked from a Sample Set of 13,797 Accounts

*Computed as the number of matches divided by the search size. The more words that need to be tested for a match, the lower the cost/benefit ratio.

One demonstration of the effectiveness of guessing is reported in [KLEI90]. From a variety of sources, the author collected UNIX password files, containing nearly 14,000 encrypted passwords. The result, which the author rightly characterizes as frightening, is shown in Table 3.2. In all, nearly one-fourth of the passwords were guessed. The following strategy was used:

1. Try the user’s name, initials, account name, and other relevant personal information. In all, 130 different permutations for each user were tried.

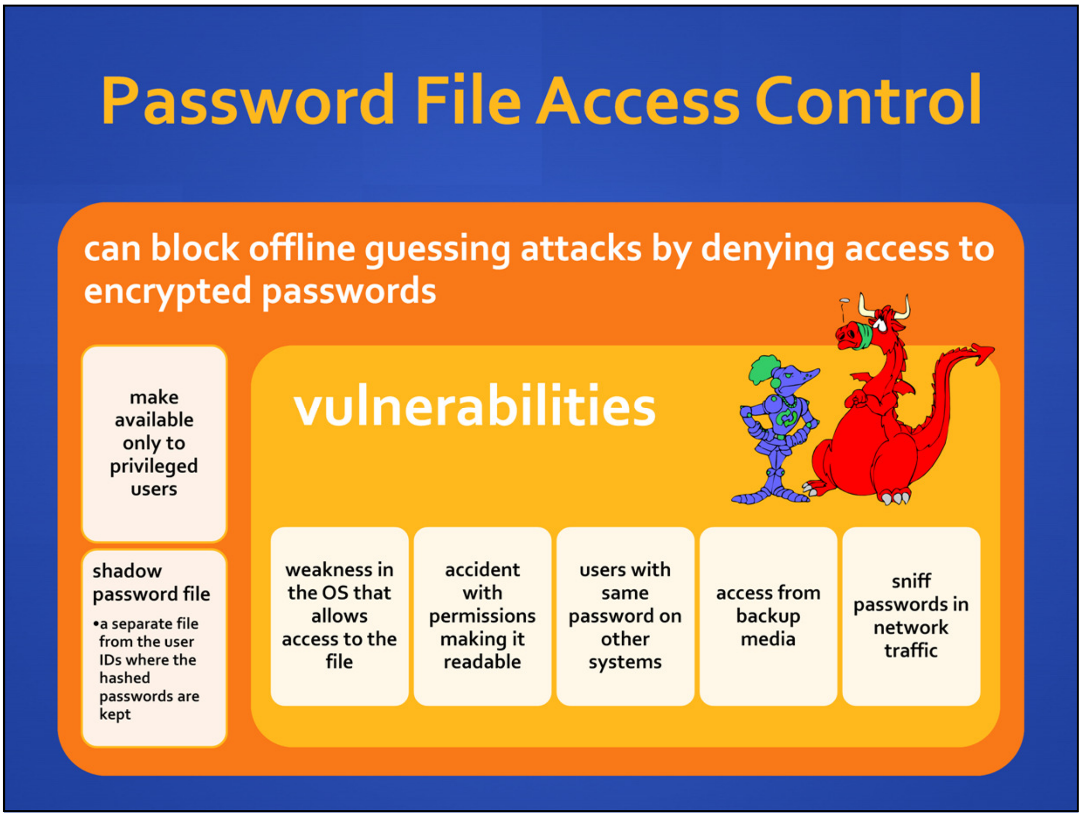
2. Try words from various dictionaries. The author compiled a dictionary of over 60,000 words, including the online dictionary on the system itself, and various other lists as shown.

3. Try various permutations on the words from step 2. This included making the first letter uppercase or a control character, making the entire word uppercase, reversing the word, changing the letter “o” to the digit “zero,” and so on.

These permutations added another 1 million words to the list.

4. Try various capitalization permutations on the words from step 2 that were not

considered in step 3. This added almost 2 million additional words to the list. Thus, the test involved in the neighborhood of 3 million words. Using the fastest Thinking Machines implementation listed earlier, the time to encrypt all these words for all possible salt values is under an hour. Keep in mind that such a thorough search could produce a success rate of about 25%, whereas even a single hit may be enough to gain a wide range of privileges on a system.



Password File Access Control

can block offline guessing attacks by denying access to encrypted passwords

- make available only to privileged users
- shadow password file
  - a separate file from the user IDs where the hashed passwords are kept

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

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

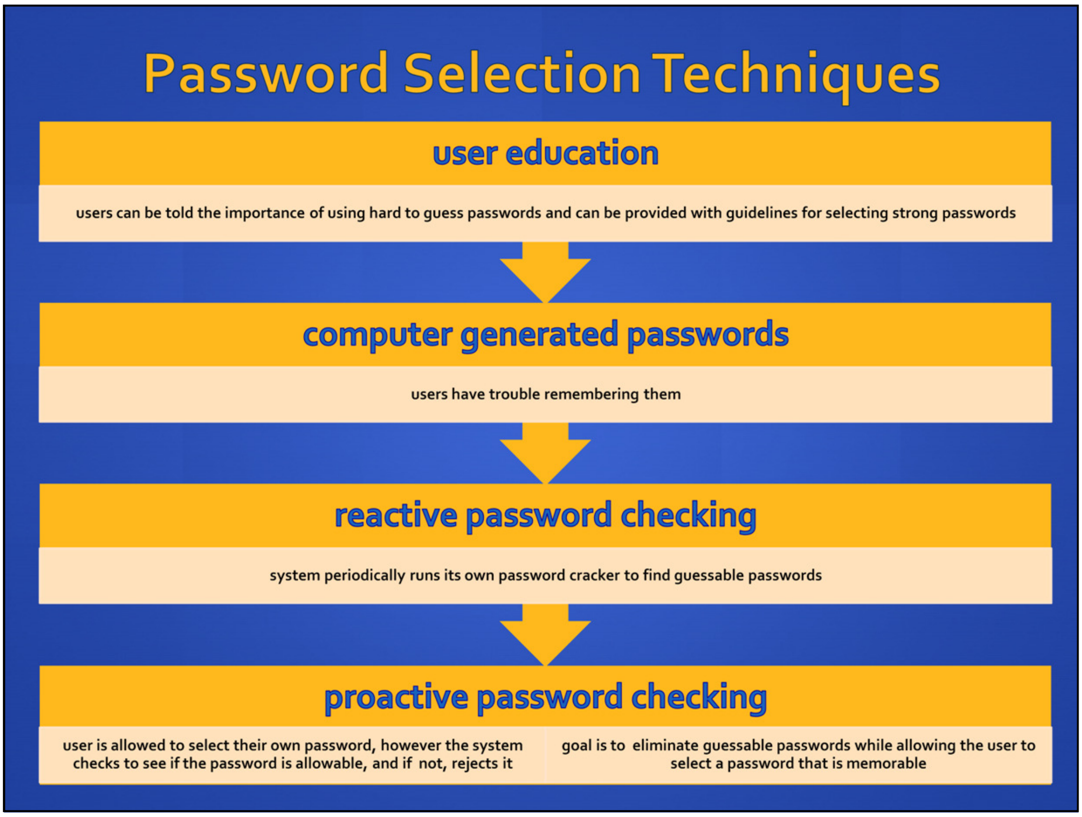
compromising
all the accounts.

- 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 Techniques

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

proactive password checking

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

The lesson from the two experiments just described (Tables 3.1 and 3.2) is that, 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
- Proactive password checking

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, don’t 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.

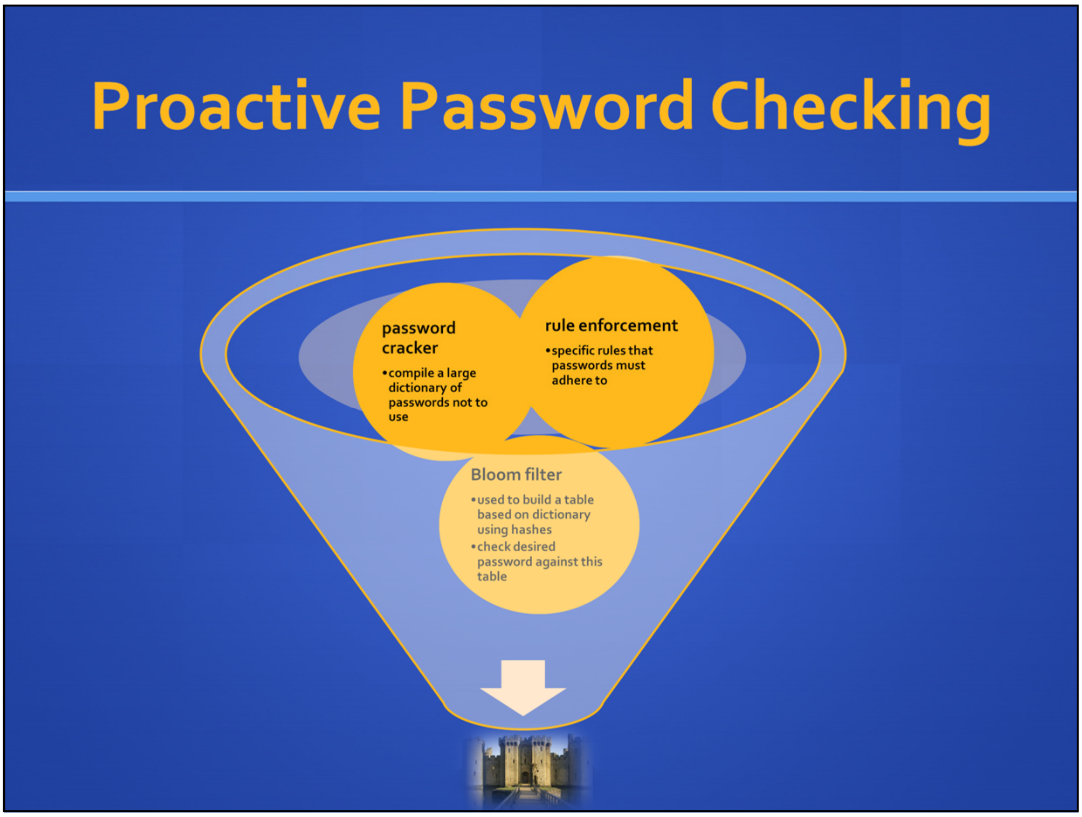
Computer-generated passwords also have problems. If the passwords are quite random in nature, users will not be able to remember them. Even if the password is pronounceable, the user may have difficulty remembering it and so be tempted to write it down. In general, computer-generated password schemes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

password cracker
- compile a large dictionary of passwords not to use

rule enforcement
- specific rules that passwords must adhere to

Bloom filter
- used to build a table based on dictionary using hashes
- check desired password against this table

The first approach is a simple system for rule enforcement.
For example, the following rules could be enforced:

- All passwords must be at least eight characters long.

- In the first eight characters, the passwords must include at least one each of uppercase, lowercase, numeric digits, and punctuation marks.

These rules could be coupled with advice to the user. 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. For example, the dictionary** used in the Purdue study [SPAF92a] occupies more than 30 megabytes of storage.

- **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 for developing an effective and efficient proactive password checker that is based on rejecting words on a list that has been implemented on a number of systems, including Linux. It is based on the use of a Bloom filter [BLOO70]. To begin, we explain the operation of the Bloom filter. A Bloom filter of order *k consists of a set of k* independent hash functions *H1(x), H2(x),c, Hk(x), where each function maps a* password into a hash value in the range 0 to *N – 1.*

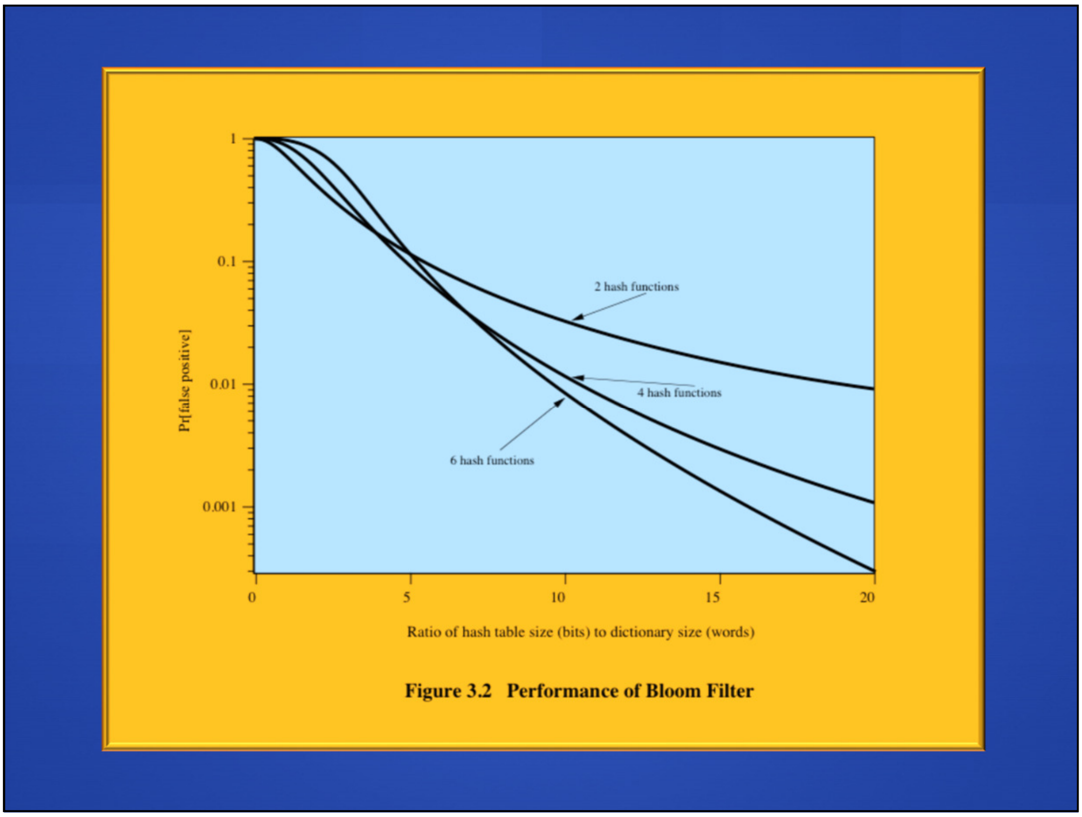


Figure 3.2 Performance of Bloom Filter

Figure 3.2 plots *P as a function of R for various values of k. Suppose we have* a dictionary of 1 million words and we wish to have a 0.01 probability of rejecting a password not in the dictionary. If we choose six hash functions, the required ratio is *R 9.6. Therefore, we need a hash table of 9.6 106 bits or about 1.2 MBytes* of storage. In contrast, storage of the entire dictionary would require on the order of 8 MBytes. Thus, we achieve a compression of almost a factor of 7. Furthermore, password checking involves the straightforward calculation of six hash functions and is independent of the size of the dictionary, whereas with the use of the full dictionary, there is substantial searching.

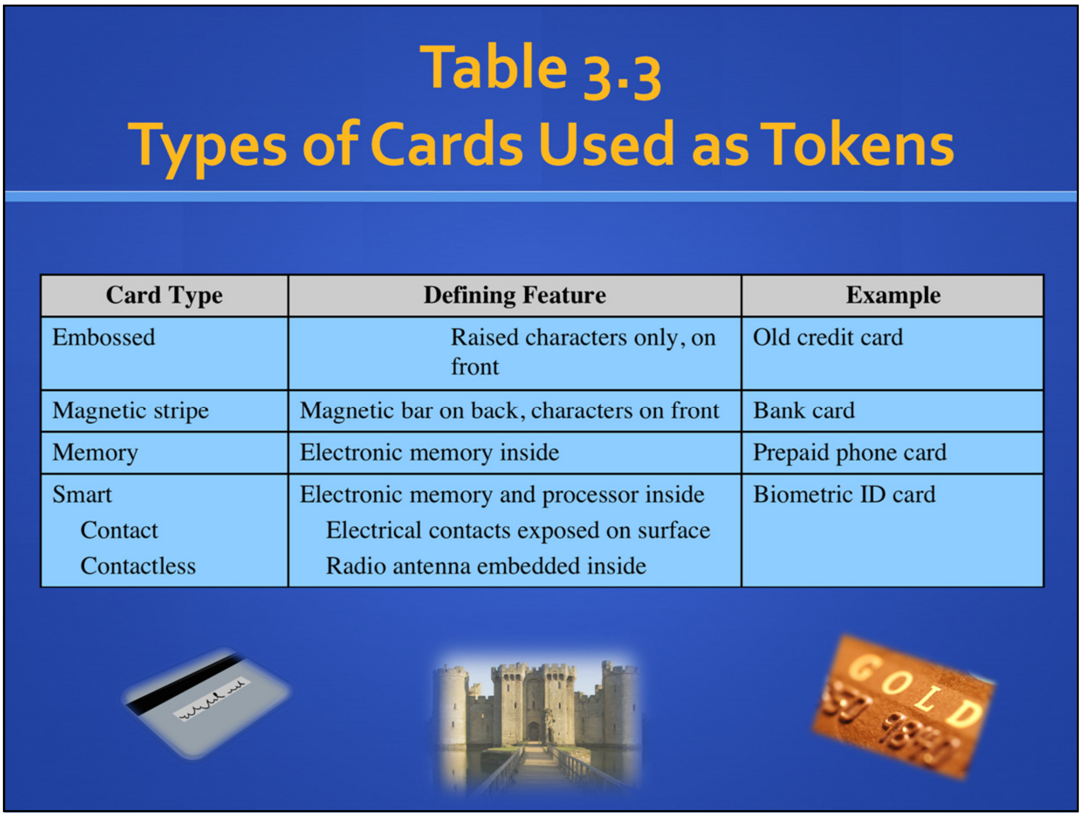
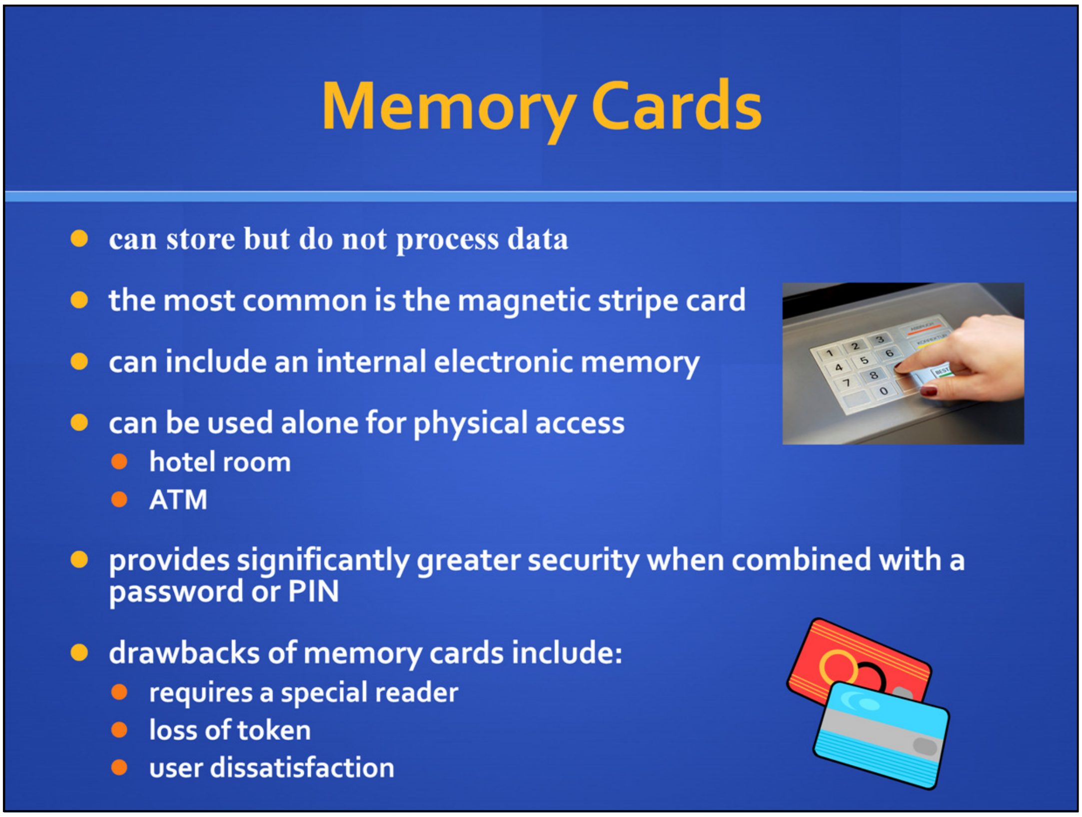


Table 3.3
Types of Cards Used as Tokens

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 are the following [NIST95]:

- **Requires special reader: This increases the cost of using the token and creates**

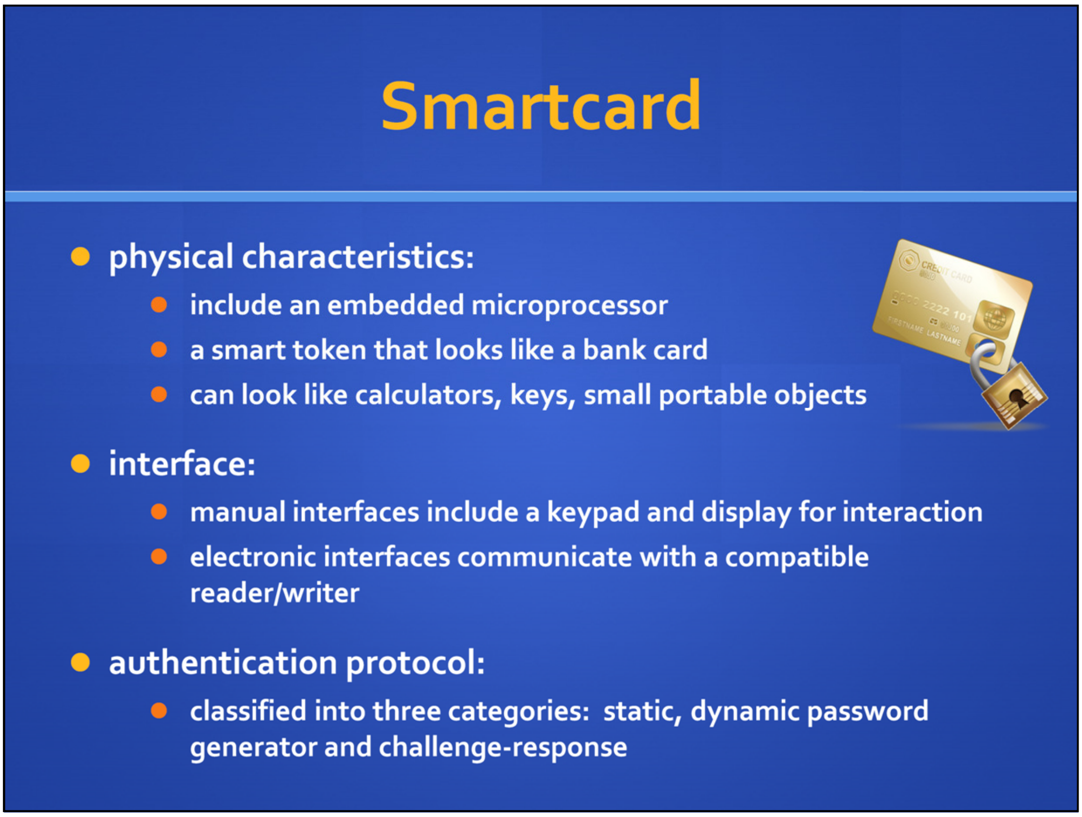
  the requirement to maintain the security of the reader’s hardware and software.

- **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.



Smartcard

- physical characteristics:
  - include an embedded microprocessor
  - a smart token that looks like a bank card
  - can look like calculators, keys, small portable objects
- interface:
  - manual interfaces include a keypad and display for interaction
  - electronic interfaces communicate with a compatible reader/writer
- authentication protocol:
  - classified into three categories: static, dynamic password generator and 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.

- **Interface: Manual interfaces include a keypad and display for human/token** interaction. Smart tokens with an electronic interface communicate with a compatible reader/writer.

- **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.

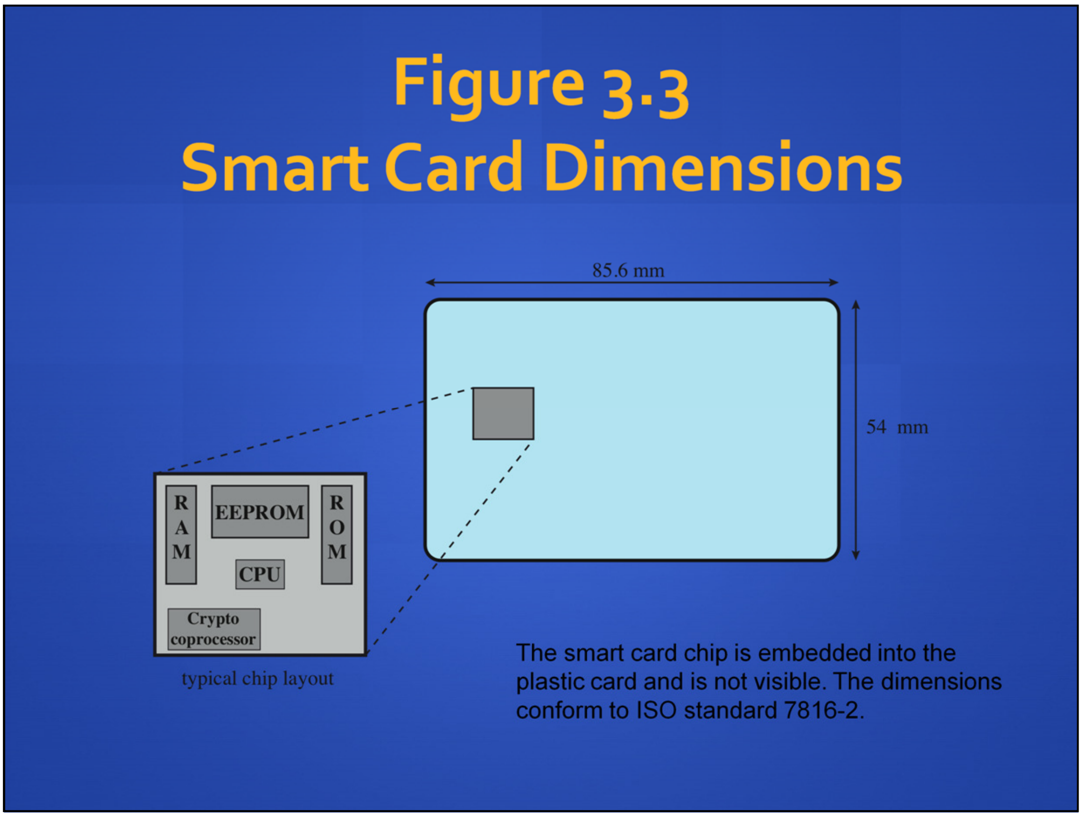


Figure 3.3
Smart Card Dimensions

The smart card chip is embedded into the plastic card and is not visible. The dimensions conform to ISO standard 7816-2.

For user authentication to computer, 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 (Figure 3.3). 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.

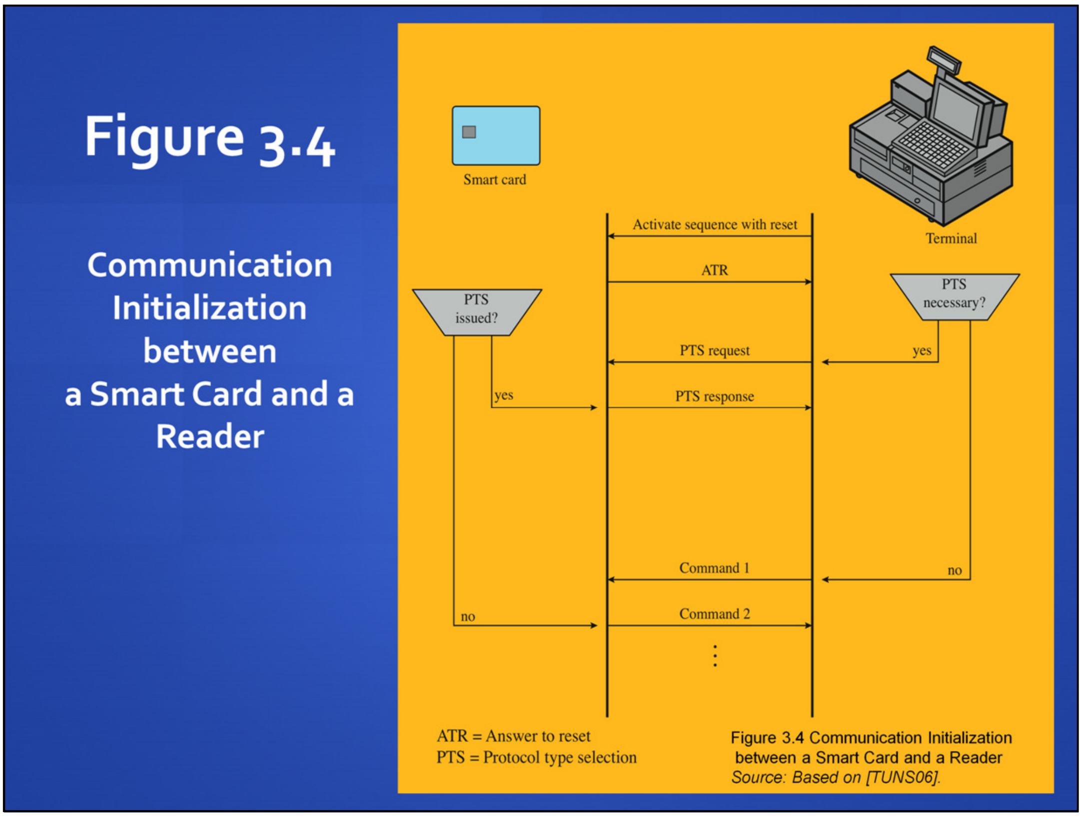


Figure 3.4

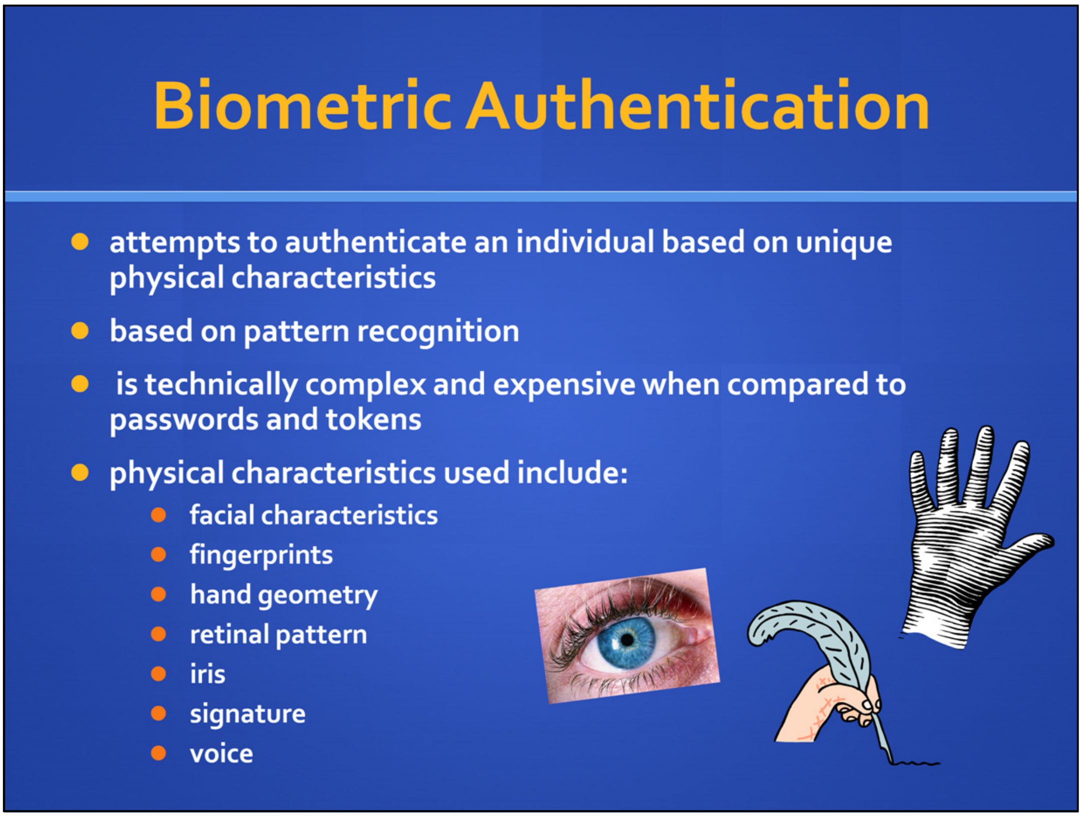
Communication Initialization between a Smart Card and a Reader

ATR = Answer to reset
PTS = Protocol type selection

Figure 3.4 Communication Initialization between a Smart Card and a Reader
Source: Based on [TUNS06].

Figure 3.4 illustrates the typical interaction between a smart card and a reader or computer system. Each time the card is inserted into a reader, a reset is initiated by the reader to initialize parameters such as clock value. After the reset function is performed, the card responds with answer to reset (ATR) message.

This message defines the parameters and protocols that the card can use and the functions it can perform. The terminal may be able to change the protocol used and other parameters via a protocol type selection (PTS) command. The cards PTS response confirms the protocols and parameters to be used. The terminal and card can now execute the protocol to perform the desired application.



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.

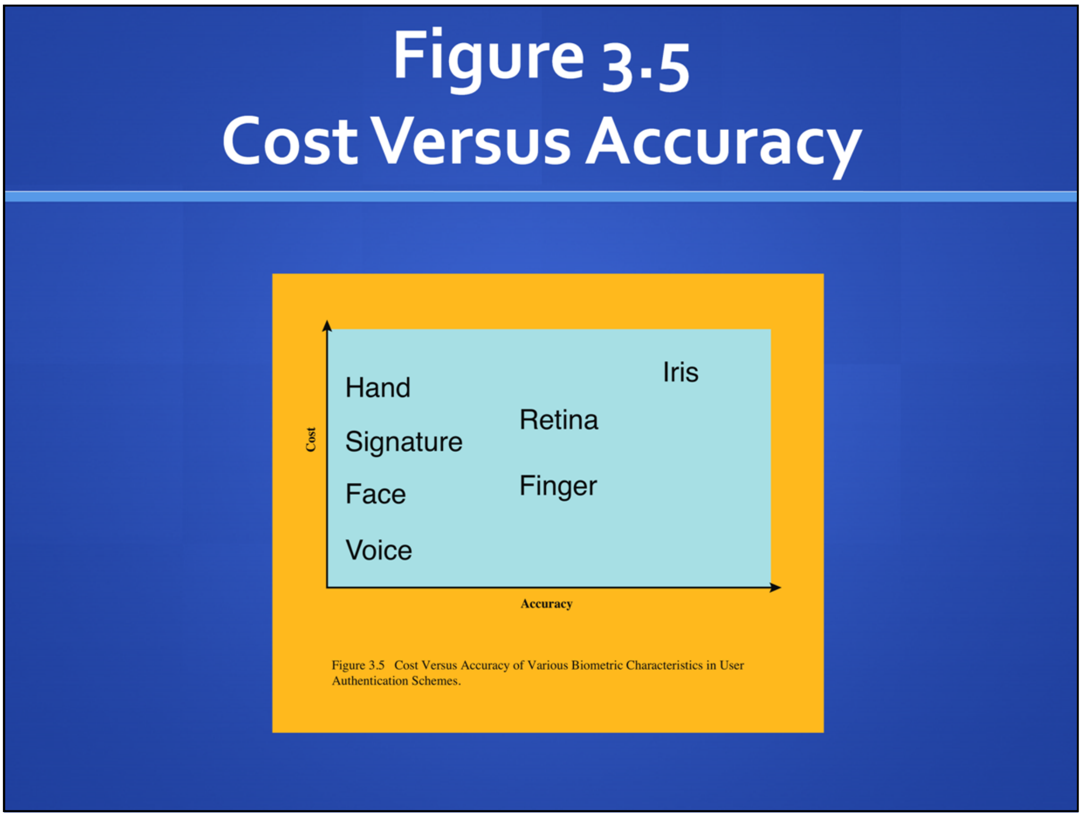


Figure 3.5
Cost Versus Accuracy

Figure 3.5 Cost Versus Accuracy of Various Biometric Characteristics in User Authentication Schemes.

Figure 3.5 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.

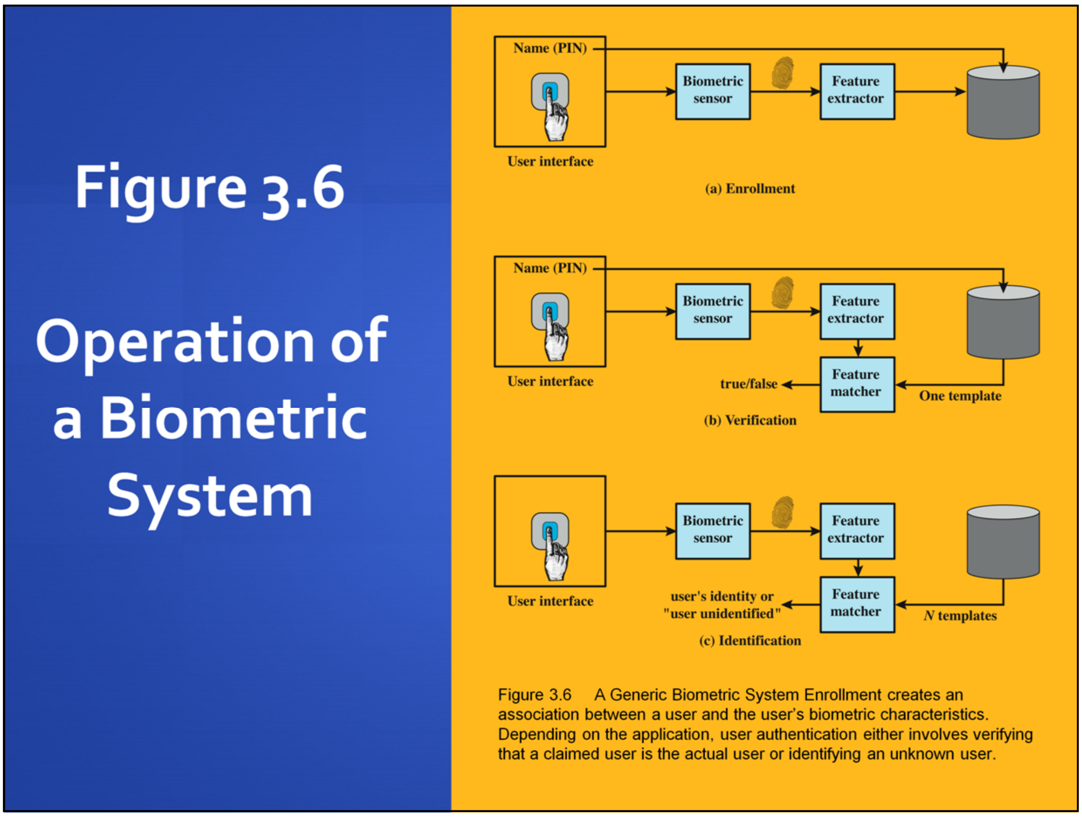


Figure 3.6

Operation of a Biometric System

Figure 3.6 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 3.6 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.

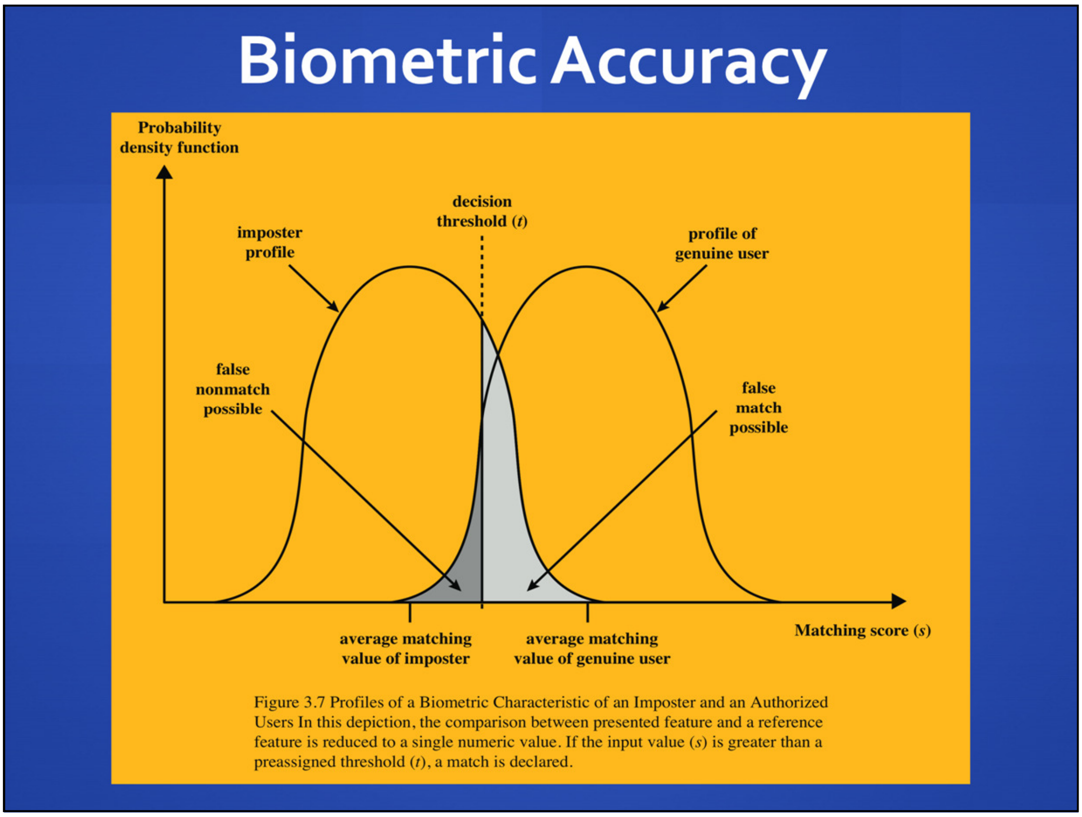
Depending on application, user authentication on a biometric system involves either **verification or identification. Verification is analogous to a user**

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.

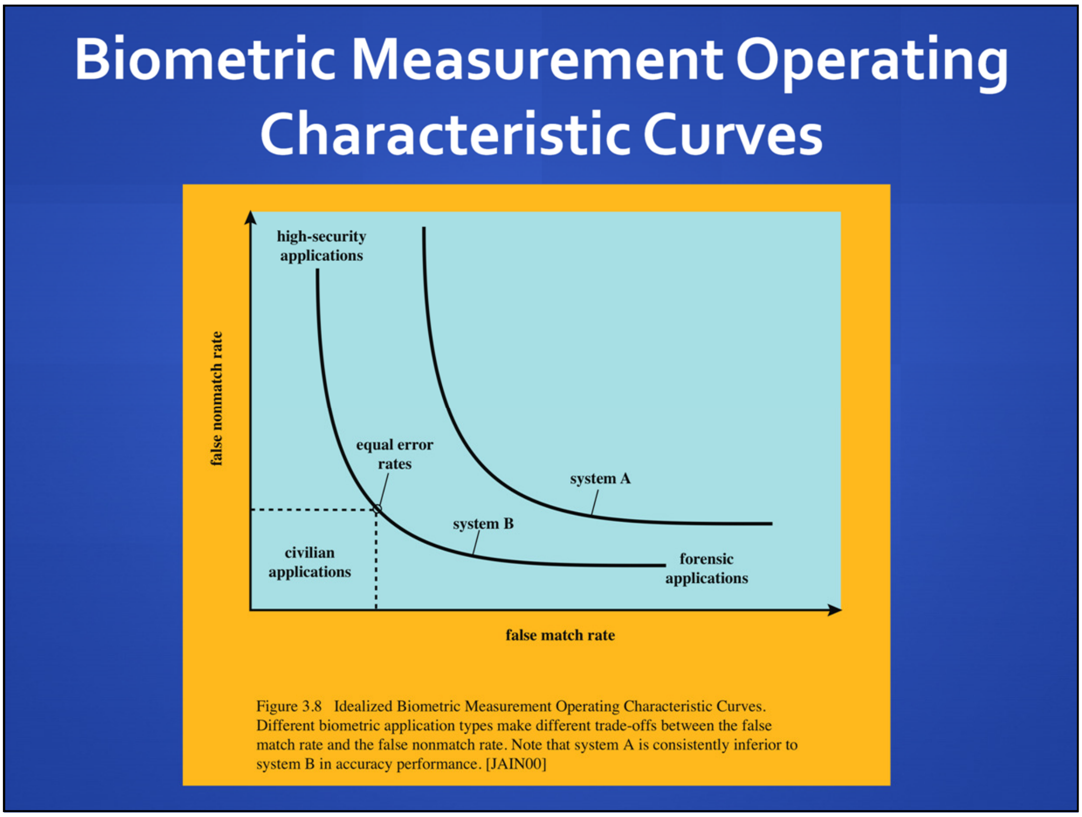


Biometric Accuracy

Figure 3.7 Profiles of a Biometric Characteristic of an Imposter and an Authorized Users In this depiction, the comparison between presented feature and a reference feature is reduced to a single numeric value. If the input value (s) is greater than a preassigned threshold (t), a match is declared.

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.

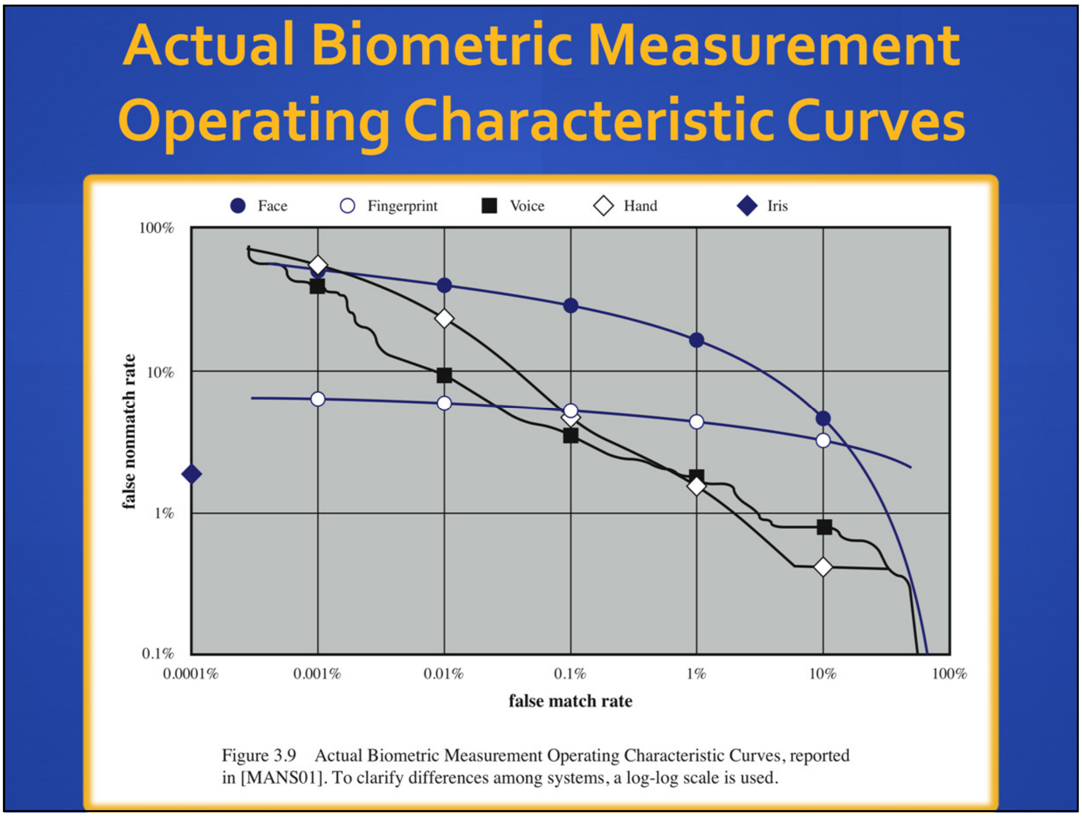
Figure 3.7 illustrates the dilemma posed to the system. If a single user is tested by the system numerous times, the matching score *s will vary, with a probability* density function typically forming a bell curve, as shown. For example, in the case of a fingerprint, results may vary due to sensor noise; changes in the print due to swelling, dryness, and so on; finger placement; and so on. On average, any other individual should have a much lower matching score but again will exhibit a bell-shaped probability density function. The difficulty is that the range of matching scores produced by two individuals, one genuine and one an imposter, compared to a given reference template, are likely to overlap. In Figure 3.7 a threshold value is selected thus that if the presented value s Ú t a match is assumed, and for *s 6 t, a mismatch is assumed.* The shaded part to the right of *t indicates a range of values for which a false match is* possible, and the shaded part to the left indicates a range of values for which a false nonmatch is possible. The area of each shaded part represents the probability of a false match or nonmatch, respectively. By moving the threshold, left or right, the probabilities can be altered, but note that a decrease in false match rate necessarily results in an increase in false nonmatch rate, and vice versa.



Biometric Measurement Operating Characteristic Curves

Figure 3.8 Idealized Biometric Measurement Operating Characteristic Curves. Different biometric application types make different trade-offs between the false match rate and the false nonmatch rate. Note that system A is consistently inferior to system B in accuracy performance. [JAIN00]

For a given biometric scheme, we can plot the false match versus false nonmatch rate, called the operating characteristic curve. Figure 3.8 shows representative curves for two different systems. A reasonable tradeoff is to pick a threshold *t that corresponds* to a point on the curve where the rates are equal. A high-security application may require a very low false match rate, resulting in a point farther to the left on the curve. 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.



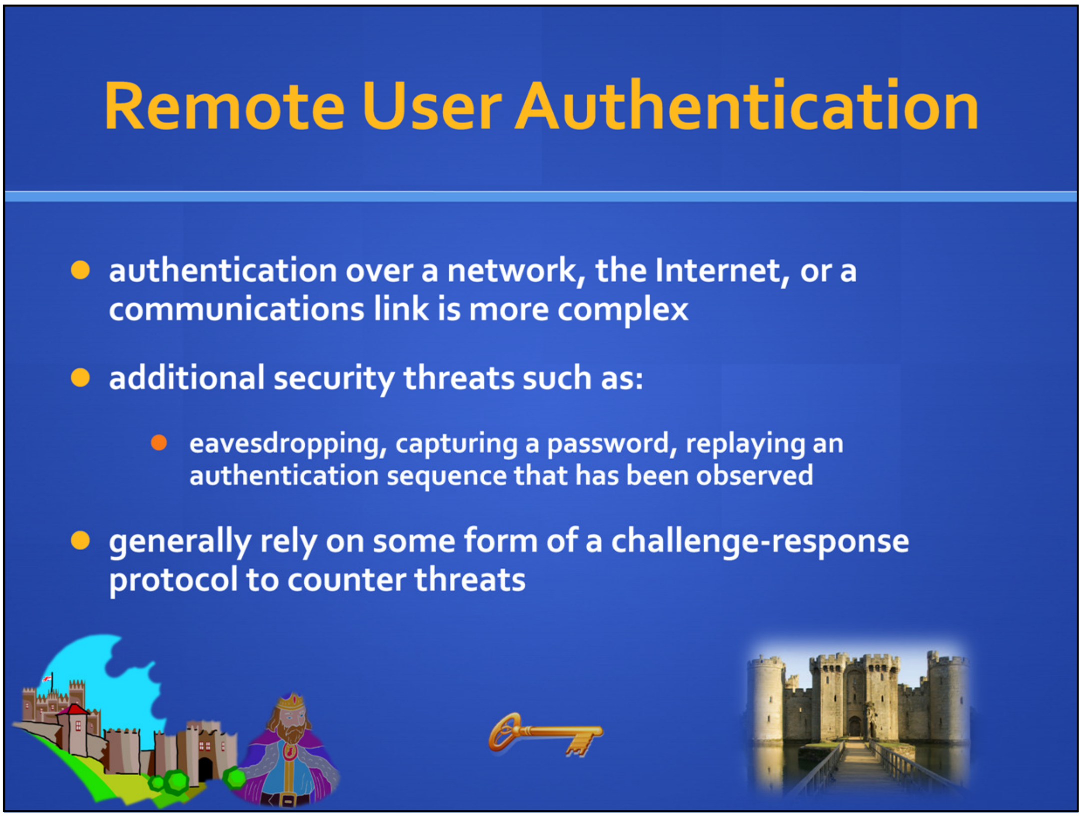
Actual Biometric Measurement Operating Characteristic Curves

Figure 3.9 Actual Biometric Measurement Operating Characteristic Curves, reported in [MANS01]. To clarify differences among systems, a log-log scale is used.

Figure 3.9 shows characteristic curves developed from actual product testing. The

iris system had no false matches in over 2 million cross-comparisons. Note that over

a broad range of false match rates, the face biometric is the worst performer.



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.

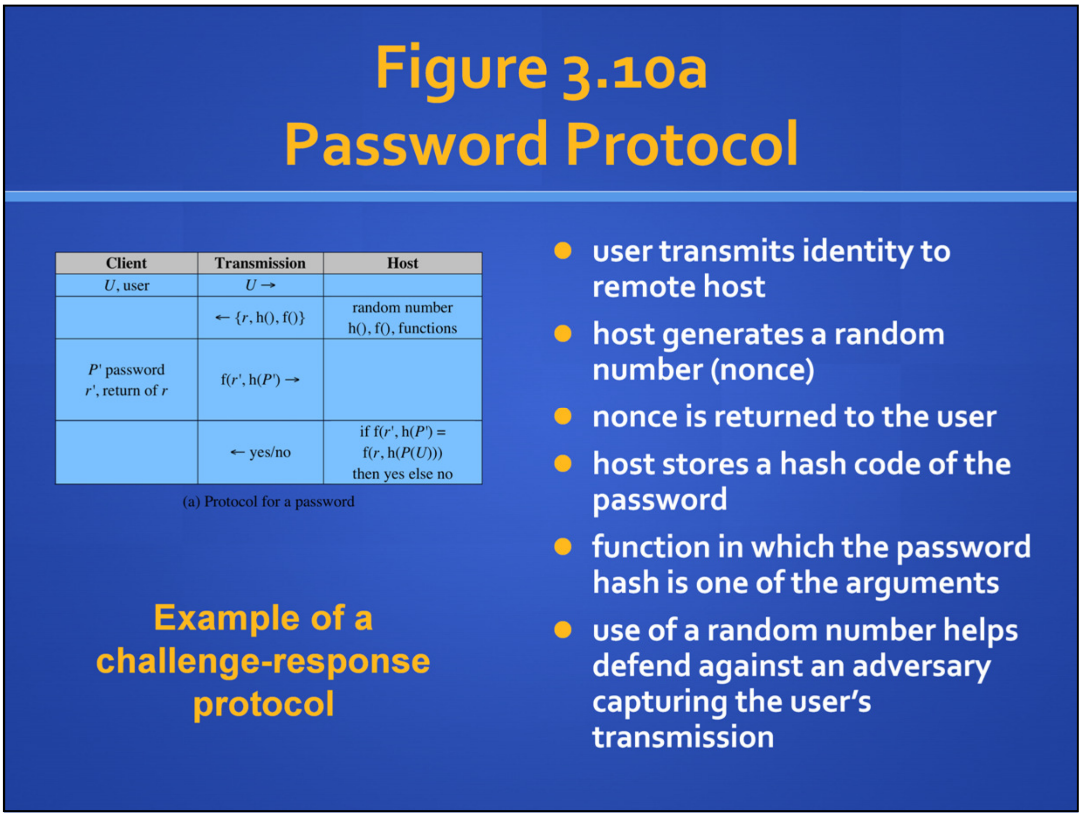


Figure 3.10a Password Protocol

Client	Transmission	Host
U, user	$U \rightarrow$	
	$\leftarrow \{r, \mathrm{h}(), \mathrm{f}()\}$	random number h(), f(), functions
P' password r', return of r	$\mathrm{f}(r', \mathrm{h}(P')) \rightarrow$	
	← yes/no	if $\mathrm{f}(r', \mathrm{h}(P') =$ $\mathrm{f}(r, \mathrm{h}(P(U)))$ then yes else no

(a) Protocol for a password

Example of a challenge-response protocol

- user transmits identity to remote host
- host generates a random number (nonce)
- nonce is returned to the user
- host stores a hash code of the password
- function in which the password hash is one of the arguments
- use of a random number helps defend against an adversary capturing the user's transmission

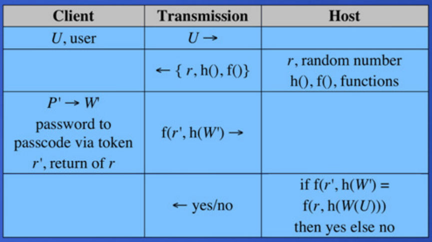
Figure 3.10a provides a simple example of a challenge-response protocol for authentication via password. Actual protocols are more complex, such as Kerberos, discussed in Chapter 23. In this example, a user first transmits his or her identity to the remote host. The host generates a random number *r, 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 3.10b
Token Protocol

- user transmits identity to the remote host
- host returns a random number and identifiers
- token either stores a static passcode or generates a one-time random passcode
- user activates passcode by entering a password
- password is shared between the user and token and does not involve the remote host



Client	Transmission	Host
U, user	$U \rightarrow$	
	$\leftarrow \{ r, \mathrm{h}(), \mathrm{f}() \}$	r, random number h(), f(), functions
$P' \rightarrow W'$ password to passcode via token r', return of r	$\mathrm{f}(r', \mathrm{h}(W') \rightarrow$	
	$\leftarrow$ yes/no	if $\mathrm{f}(r', \mathrm{h}(W') =$ $\mathrm{f}(r, \mathrm{h}(W(U)))$ then yes else no

(b) Protocol for a token

Example of a token protocol

Figure 3.10b 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.

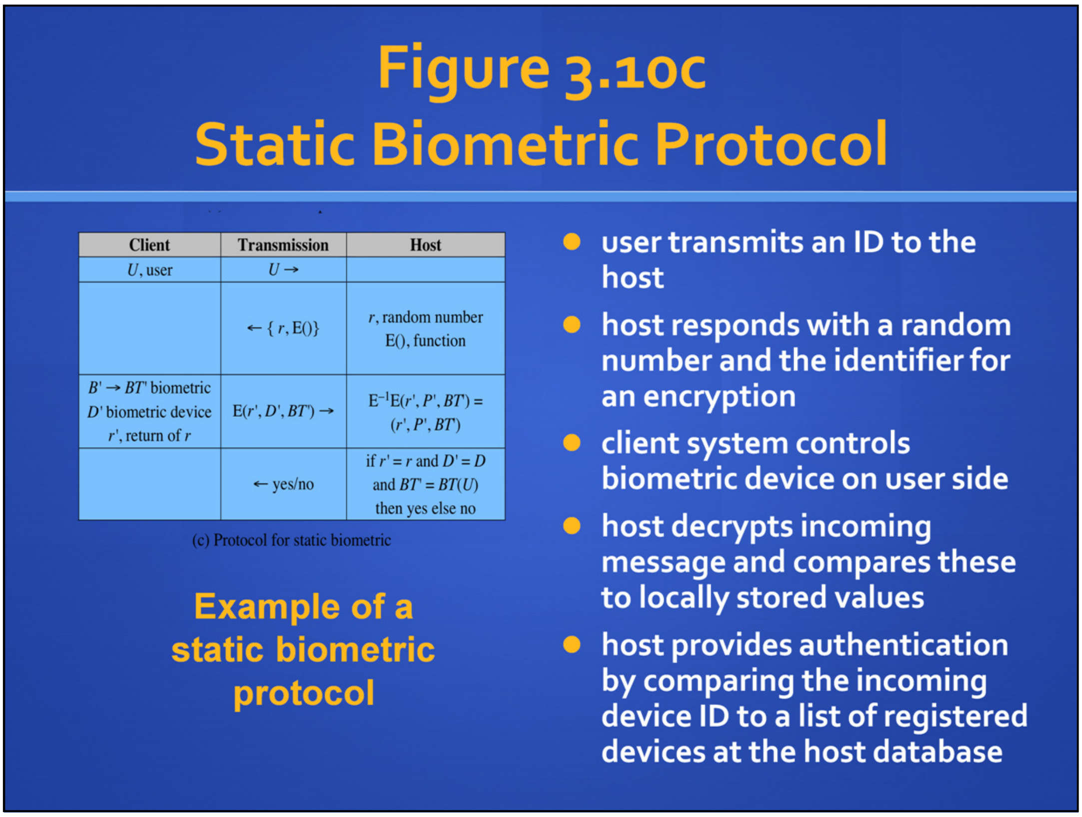


Figure 3.10c
Static Biometric Protocol

Client	Transmission	Host
U, user	$U \rightarrow$	
	$\leftarrow \{ r, \mathrm{E}() \}$	r, random number E(), function
$B' \rightarrow BT'$ biometric D' biometric device r', return of r	$\mathrm{E}(r', D', BT') \rightarrow$	$\mathrm{E}^{-1}\mathrm{E}(r', P', BT') =$ (r', P', BT')
	← yes/no	if $r' = r$ and $D' = D$ and $BT' = BT(U)$ then yes else no

(c) Protocol for static biometric

Example of a static biometric protocol

- user transmits an ID to the host
- host responds with a random number and the identifier for an encryption
- client system controls biometric device on user side
- host decrypts incoming message and compares these to locally stored values
- host provides authentication by comparing the incoming device ID to a list of registered devices at the host database

Figure 3.10c 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.

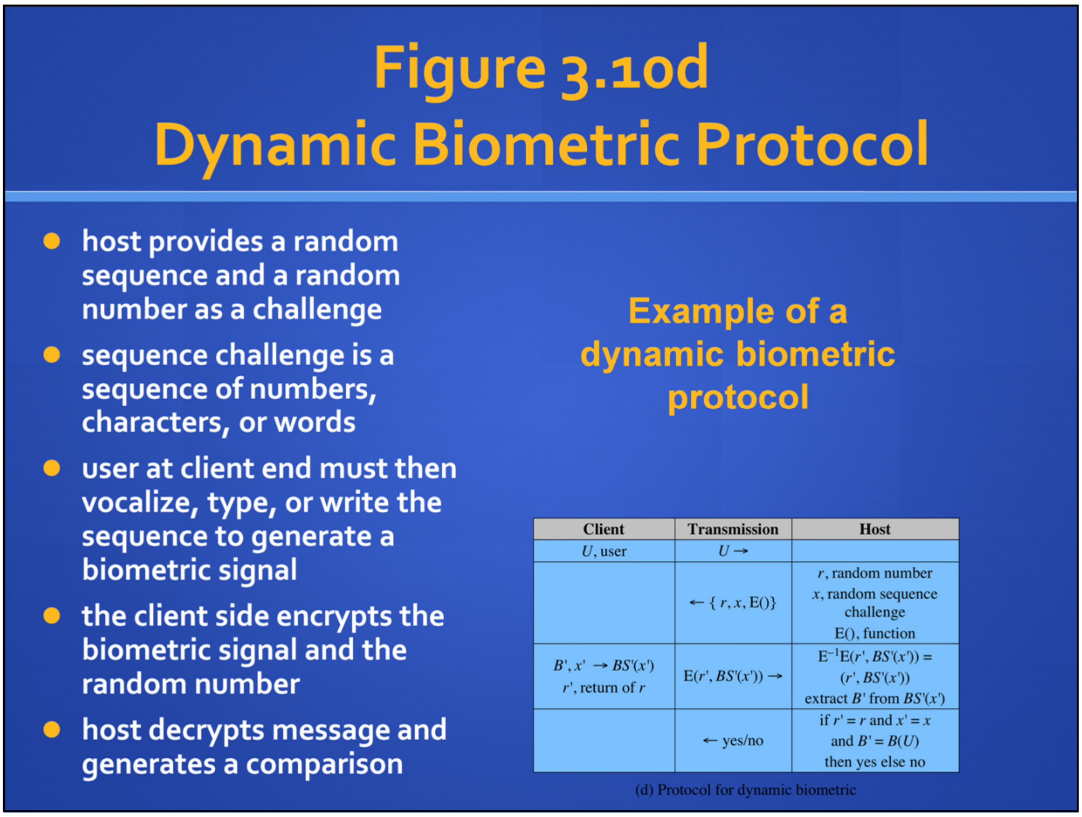


Figure 3.10d
Dynamic Biometric Protocol

- host provides a random sequence and a random number as a challenge
- sequence challenge is a sequence of numbers, characters, or words
- user at client end must then vocalize, type, or write the sequence to generate a biometric signal
- the client side encrypts the biometric signal and the random number
- host decrypts message and generates a comparison

Example of a dynamic biometric protocol

Client	Transmission	Host
U, user	$U \rightarrow$	
	$\leftarrow \{ r, x, \mathrm{E}() \}$	r, random number x, random sequence challenge E(), function
$B', x' \rightarrow BS'(x')$ r', return of r	$\mathrm{E}(r', BS'(x')) \rightarrow$	$\mathrm{E}^{-1}\mathrm{E}(r', BS'(x')) = (r', BS'(x'))$ extract B' from $BS'(x')$
	$\leftarrow$ yes/no	if $r' = r$ and $x' = x$ and $B' = B(U)$ then yes else no

(d) Protocol for dynamic biometric

Figure 3.10d 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.

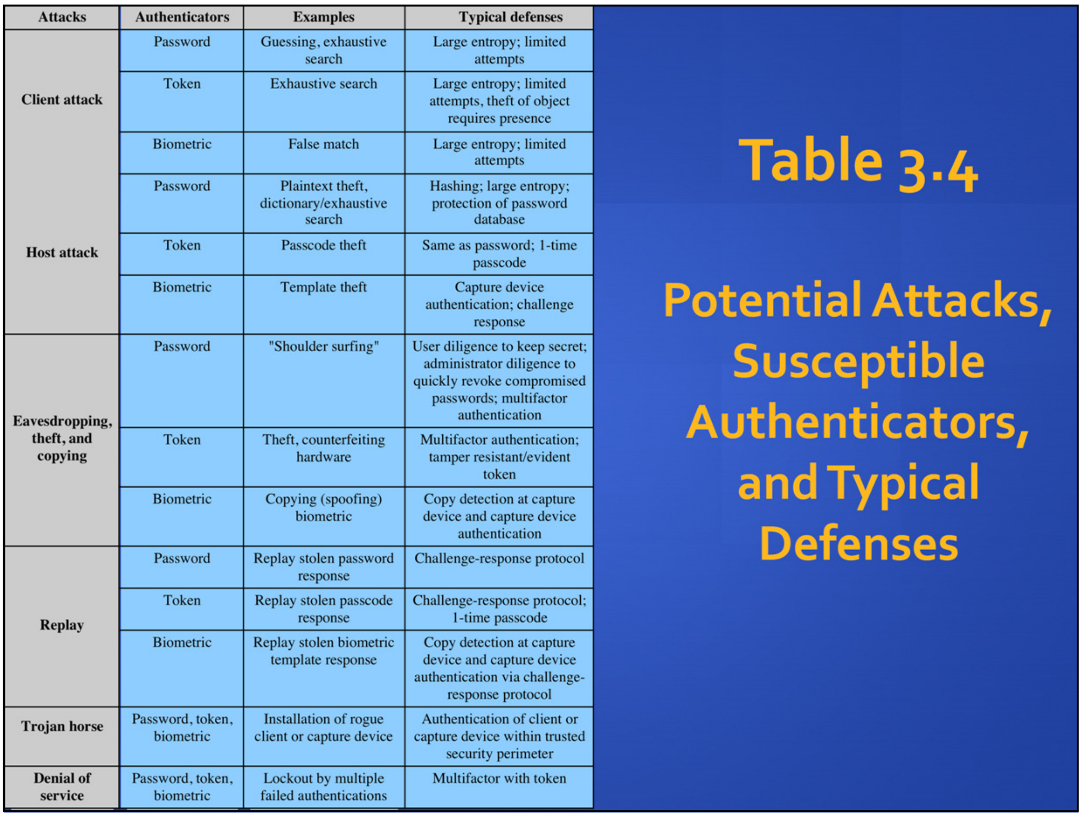


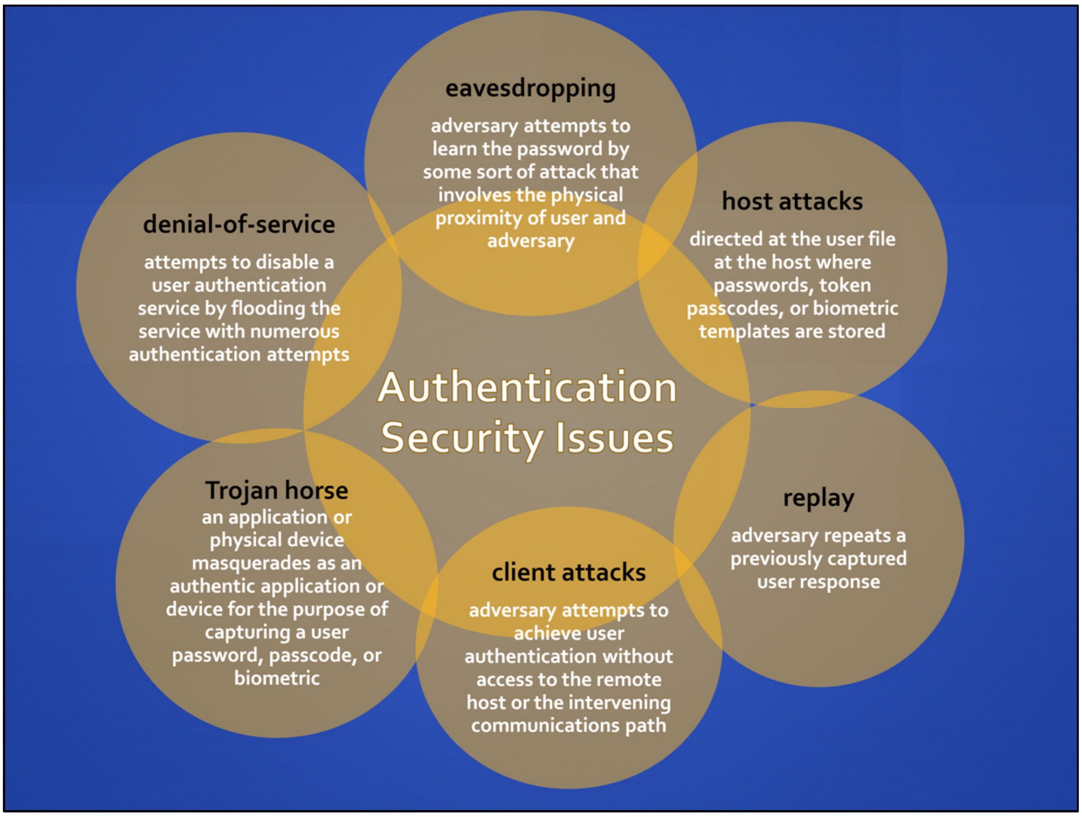
Table 3.4

Potential Attacks, Susceptible Authenticators, and Typical Defenses

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
Eavesdropping, 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
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

As with any security service, user authentication, particularly remote user authentication, is subject to a variety of attacks. Table 3.4, 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.



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.

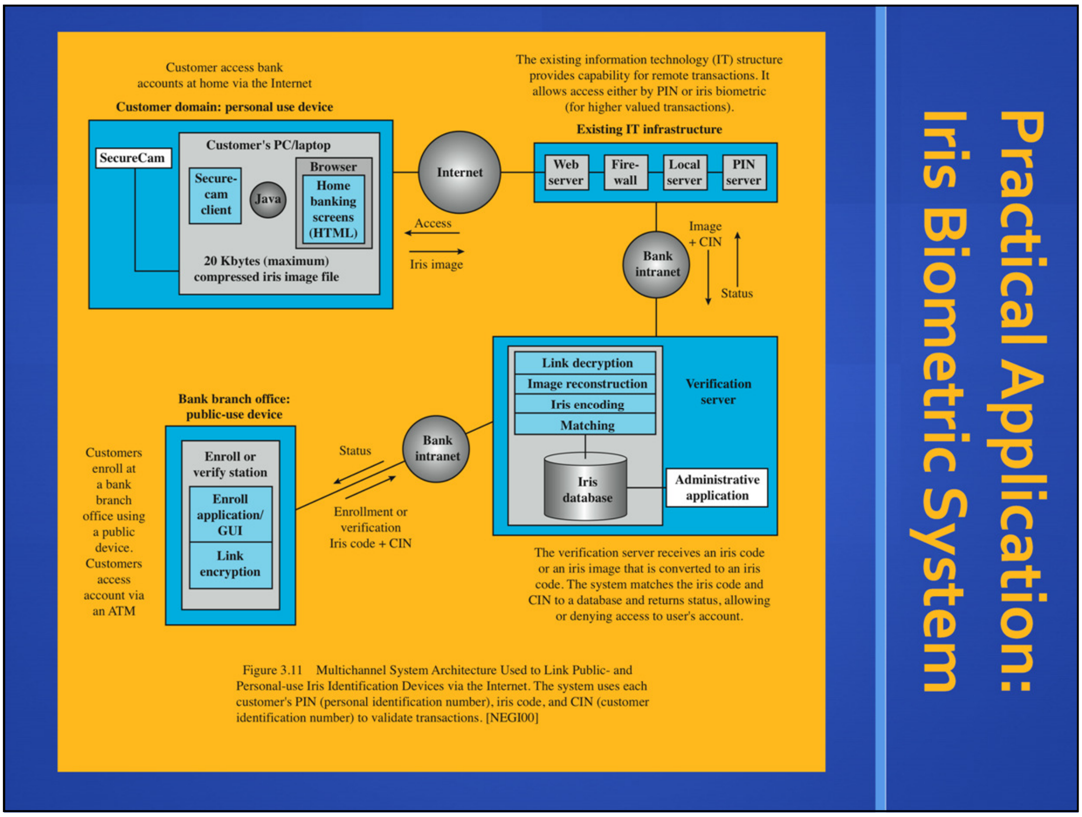
Eavesdropping in the context of passwords refers to an adversary's attempt to learn the password by observing the user, finding a written copy of the password, or some similar attack that involves the physical proximity of user and adversary. Another form of eavesdropping is keystroke logging (keylogging), 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.



Practical Application: Iris Biometric System

Figure 3.11 Multichannel System Architecture Used to Link Public- and Personal-use Iris Identification Devices via the Internet. The system uses each customer's PIN (personal identification number), iris code, and CIN (customer identification number) to validate transactions. [NEGI00]

As an example of a biometric user authentication system, we look at an iris biometric system that was developed for use in the banking industry [NEGI00] for authentication of debit card users. Figure 3.11 shows a generic version of this system, which is now in use commercially in a number of locations worldwide. There is considerable interest commercially in the use of an iris biometric system for this application because of its exceptional accuracy (see Figure 3.9) and because the biometric itself can be acquired without the individual having to come into physical contact with the biometric acquisition device [COVE03].

The system described in this section is designed to operate with automated teller machines (ATMs) in public places as well as with personal use devices that can be installed at home. For ATMs, a wide-angle camera finds the head of the person to be identified. A zoom lens then targets in on the user’s iris and takes a digital photo. A template of concentric lines is laid on the iris image and a number of specific points are recorded and the information converted into a digital code.

For personal-use systems, a low-cost camera device involves more cooperative action on the part of the user to focus and capture the biometric.

A customer must initially enroll through a public-use ATM device owned by the bank. The biometric is converted into a numeric iris code. This code and the customer identification number (CIN) are encrypted and transmitted over the bank's intranet to a verification server. The verification server then performs the user authentication function. A user may employ a personal-use device to access the system via the Internet. The image information plus the CIN are transmitted securely over the Internet to the bank's Web server. From there, the data are transmitted over the bank's intranet to the verification server. In this case, the verification server does the conversion of iris image to iris code.

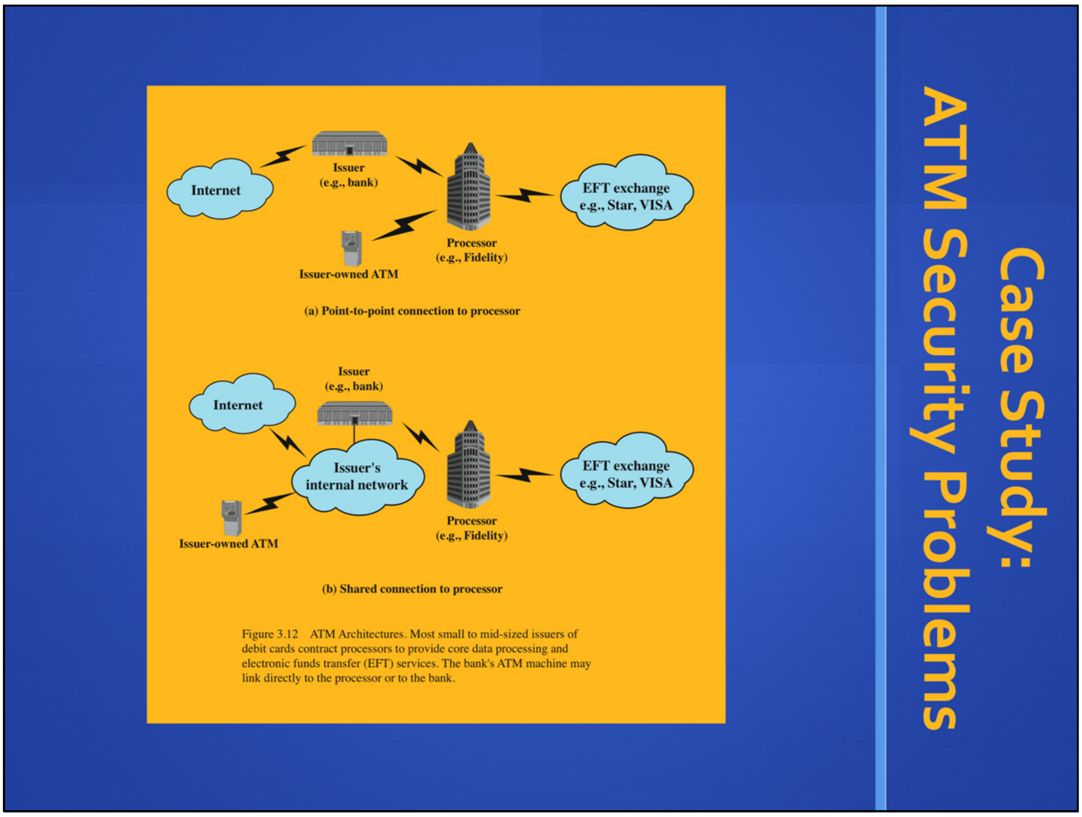
Initial field trials of the system showed very high acceptance rate of customers preferring this method to other user authentication techniques, such as PIN codes. The specific results reported in [NEGI00] are as follows:

- 91% prefer iris identification to PIN or signature.
- 94% would recommend iris identification to friends and family.
- 94% were comfortable or very comfortable with the system.

These results are very encouraging, because of the inherent advantage of iris biometric systems over passwords, PINs, and tokens. Unlike other biometric parameters, iris biometric systems, properly implemented, have virtually zero false match rate. And whereas passwords can be guessed, and passwords, PINs, and tokens can be stolen, this is not the case with a user's iris pattern. Combined with a challenge-response protocol to assure real-time acquisition of the iris pattern, iris biometric authentication is highly attractive.

The field trials referenced earlier were conducted in 1998 with the Nationwide Building Society in Swindon, England. The bank subsequently put the system into full-time operation. Following this, a number of other banks throughout the world adopted this iris biometric system.

An instructive epilogue to this case study is the fate of the Nationwide Building Society system. The system was in use at its Swindon headquarters branch for 5 years, until 2003, and the bank planned to deploy the system nationwide in all its branches. It was anticipated that the cost of the system would drop to competitive levels, but this did not happen. Nationwide found that the iris recognition system made up 25% of the cost of individual ATM units. Thus, in 2003, Nationwide cancelled the system, although it continues to pursue biometric alternatives. The lesson here is that the technology industry needs to be careful it does not damage the future of genuinely useful technologies like biometrics by pushing for its use where there isn't a rock-solid business case.



Case Study: ATM Security Problems

(a) Point-to-point connection to processor

(b) Shared connection to processor

Figure 3.12 ATM Architectures. Most small to mid-sized issuers of debit cards contract processors to provide core data processing and electronic funds transfer (EFT) services. The bank's ATM machine may link directly to the processor or to the bank.

Redspin, Inc., an independent auditor, recently released a report describing a security vulnerability in ATM (automated teller machine) usage that affects a number of small to mid-size ATM card issuers. This vulnerability provides a useful case study illustrating that cryptographic functions and services alone do not guarantee security; they must be properly implemented as part of a system.

We begin by defining terms used in this section:

• **Cardholder: An individual to whom a debit card is issued. Typically, this** individual is also responsible for payment of all charges made to that card.

• **Issuer: An institution that issues debit cards to cardholders. This institution**
is responsible for the cardholder's account and authorizes all transactions. Banks and credit unions are typical issuers.

• **Processor: An organization that provides services such as core data processing**
(PIN recognition and account updating), electronic funds transfer (EFT), and so

on to issuers. EFT allows an issuer to access regional and national networks that connect point of sale (POS) devices and ATMs worldwide. Examples of processing companies include Fidelity National Financial and Jack Henry & Associates.

Customers expect 24/7 service at ATM stations. For many small to mid-sized issuers, it is more cost-effective for contract processors to provide the required data processing and EFT/ATM services. Each service typically requires a dedicated data connection between the issuer and the processor, using a leased line or a virtual leased line.

Prior to about 2003, the typical configuration involving issuer, processor, and ATM machines could be characterized by Figure 3.12a. The ATM units linked directly to the processor rather than to the issuer that owned the ATM, via leased or virtual leased line. The use of a dedicated link made it difficult to maliciously intercept transferred data. To add to the security, the PIN portion of messages transmitted from ATM to processor was encrypted using DES (Data Encryption Standard). Processors have connections to EFT (electronic funds transfer) exchange networks to allow cardholders access to accounts from any ATM. With the configuration of Figure 3.12a, a transaction proceeds as follows. A user swipes her card and enters her PIN. The ATM encrypts the PIN and transmits it to the processor as part of an authorization request. The processor updates the customer’s information and sends a reply.

In the early 2000s, banks worldwide began the process of migrating from an older generation of ATMs using IBM’s OS/2 operating system to new systems running Windows. The mass migration to Windows has been spurred by a number of factors, including IBM’s decision to stop supporting OS/2 by 2006, market pressure from creditors such as MasterCard International and Visa International to introduce stronger Triple DES, and pressure from U.S. regulators to introduce new features for disabled users. Many banks, such as those audited by Redspin, included a number of other enhancements at the same time as the introduction of Windows and triple DES, especially the use of TCP/IP as a network transport.

Because issuers typically run their own Internet-connected local area networks (LANs) and intranets using TCP/IP, it was attractive to connect ATMs to these issuer networks and maintain only a single dedicated line to the processor, leading to the configuration illustrated in Figure 3.12b. This configuration saves the issuer expensive monthly circuit fees and enables easier management of ATMs by the issuer. In this configuration, the information sent from the ATM to the processor traverses the issuer's network before being sent to the processor. It is during this time on the issuer's network that the customer information is vulnerable.

The security problem was that with the upgrade to a new ATM OS and a new communications configuration, the only security enhancement was the use of triple DES rather than DES to encrypt the PIN. The rest of the information in the ATM request message is sent in the clear. This includes the card number, expiration date, account balances, and withdrawal amounts. A hacker tapping into the bank's network, either from an internal location or from across the Internet potentially would have complete access to every single ATM transaction.
The situation just described leads to two principal vulnerabilities:

- **Confidentiality: The card number, expiration date, and account balance can** be used for online purchases or to create a duplicate card for signature-based transactions.

- **Integrity: There is no protection to prevent an attacker from injecting or** altering data in transit. If an adversary is able to capture messages en route, the adversary can masquerade as either the processor or the ATM. Acting as the processor, the adversary may be able to direct the ATM to dispense money without the processor ever knowing that a transaction has occurred. If an adversary captures a user's account information and encrypted PIN, the account is compromised until the ATM encryption key is changed, enabling the adversary to modify account balances or effect transfers.

Redspin recommended a number of measures that banks can take to counter these threats. Short-term fixes include segmenting ATM traffic from the rest of the network either by implementing strict firewall rule sets or physically dividing the networks altogether. An additional short-term fix is to implement network-level encryption between routers that the ATM traffic traverses.

Long-term fixes involve changes in the application-level software. Protecting confidentiality requires encrypting all customer-related information that traverses the network. Ensuring data integrity requires better machine-to-machine authentication between the ATM and processor and the use of challenge-response protocols to counter replay attacks.

Summary

- four means of authenticating a user's identity
  - something the individual knows
  - something the individual possesses
  - something the individual is
  - something the individual does
- vulnerability of passwords
  - 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
- hashed password and salt value
- password file access control
- password selection strategies
  - user education
  - computer generated passwords
  - reactive password checking
  - proactive password checking
- Bloom filter
- token based authentication
  - memory cards
  - smart cards
- biometric authentication
- remote user authentication
  - password protocol
  - token protocol
  - static biometric protocol
  - dynamic biometric protocol

Chapter 3 summary.