User Authentication

- The fundamental building block and the primary line of defense in many computer security scenarios
- The basis for most types of access control and for user accountability

Learning Objectives

- Discuss the **four general means of authenticating** a user's identity
- Explain the mechanism by which **hashed passwords** are used for user authentication
- Understand the use of the **Bloom filter** in password management
- Present an overview of **token-based user authentication**
- Discuss the issues involved and the approaches for **remote user authentication**
- Summarize some of the key security issues for user authentication

Index

- Digital User Authentication Principles
- Password-Based Authentication
- Token-Based Authentication
- Biometric Authentication
- Remote User Authentication
- Security Issues for User Authentication
- Practical Application: An Iris Biometric System
- Case Study: Security Problems for ATM Systems

Index

- Digital User Authentication Principles
- Password-Based Authentication
- Token-Based Authentication
- Biometric Authentication
- Remote User Authentication
- Security Issues for User Authentication
- Practical Application: An Iris Biometric System
- Case Study: Security Problems for ATM Systems

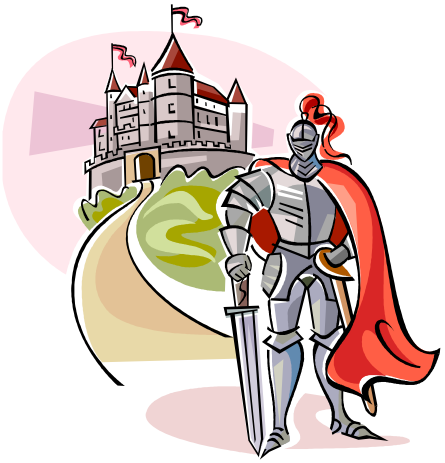
User Authentication

- 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
- Identification is the means by which a user provides a claimed identity to the system
- Verification (or authentication) is the means of establishing the validity of the user's claim

A Definition of Digital User Authentication

Digital User Authentication:
The process of establishing confidence in user identities that are presented electronically to an information system

The NIST SP 800-63-3
[Digital Authentication Guideline, October 2016]
(NIST = U.S. National Institute of Standards and Technology)



Security requirements for identification and authentication services (SP-800-171)

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

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

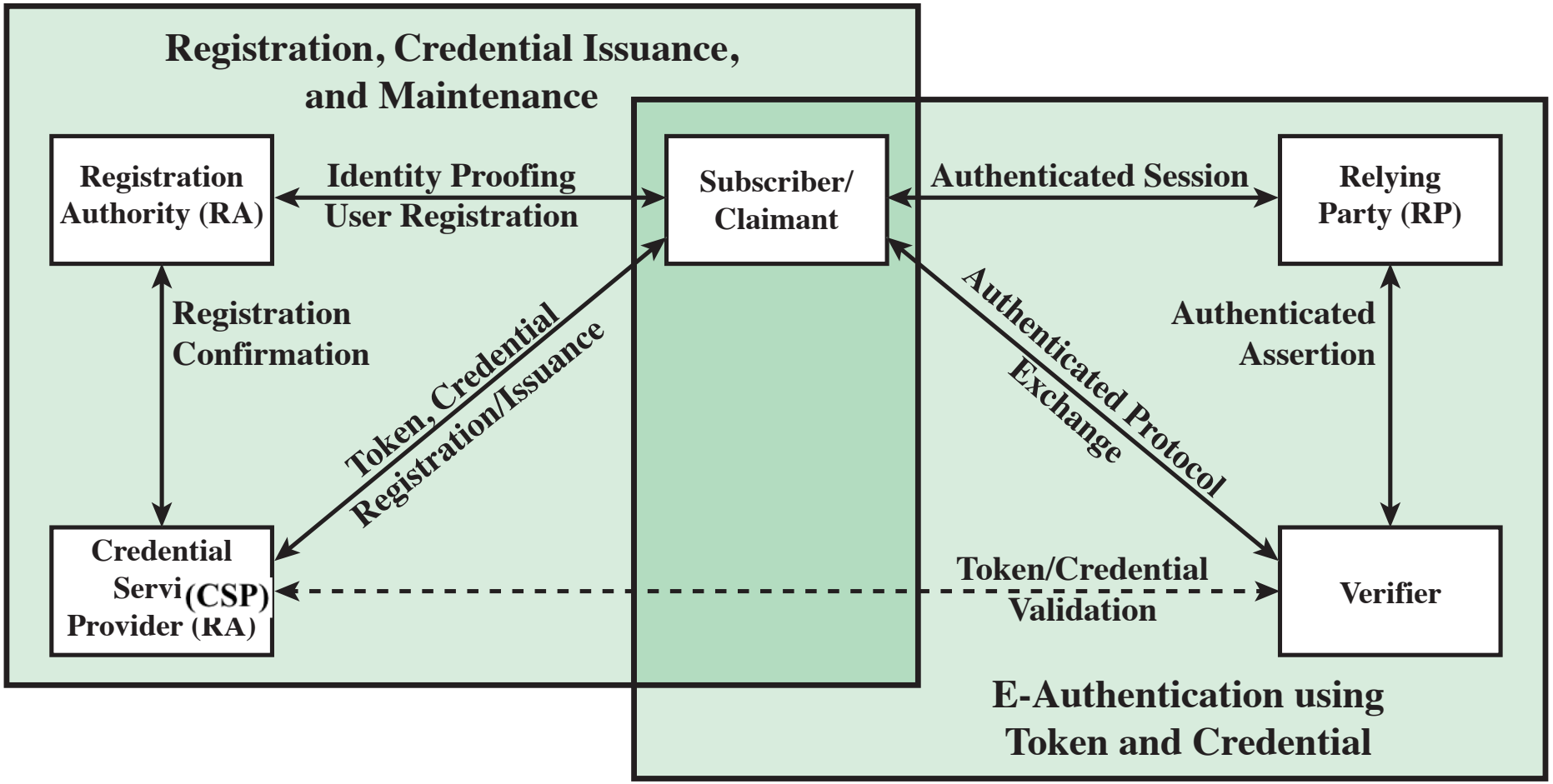
E-Authentication Architectural Model for Digital User Authentication

NIST SP 800-63-3 (*Electronic Authentication Guideline*, October 2016)

- First, the user must register with the system
  - An applicant applies to a **registration authority** to become a **subscriber** of a **credential service provider (CSP)**
  - The CSP then engages in an exchange with the subscriber and issues some sort of electronic credential
    - This is data structure that authoritatively binds an identity and additional attributes to a **token** (e.g. an encryption key or an encrypted password) possessed by the subscriber
- Then, the actual authentication process can take place
  - When the **claimant** (the party to be authenticated) presents a token to the **verifier** (the party verifying that identity) an *authentication transaction* takes place
  - The verifier passes on an **assertion** about the identity of the subscriber to the **relying party (RP)** which uses it to make *access control* or *authorization decisions*

E-Authentication Architectural Model for Digital User Authentication

NIST SP 800-63-3 (*Electronic Authentication Guideline*, October 2016)



Means of Authentication

The four general 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

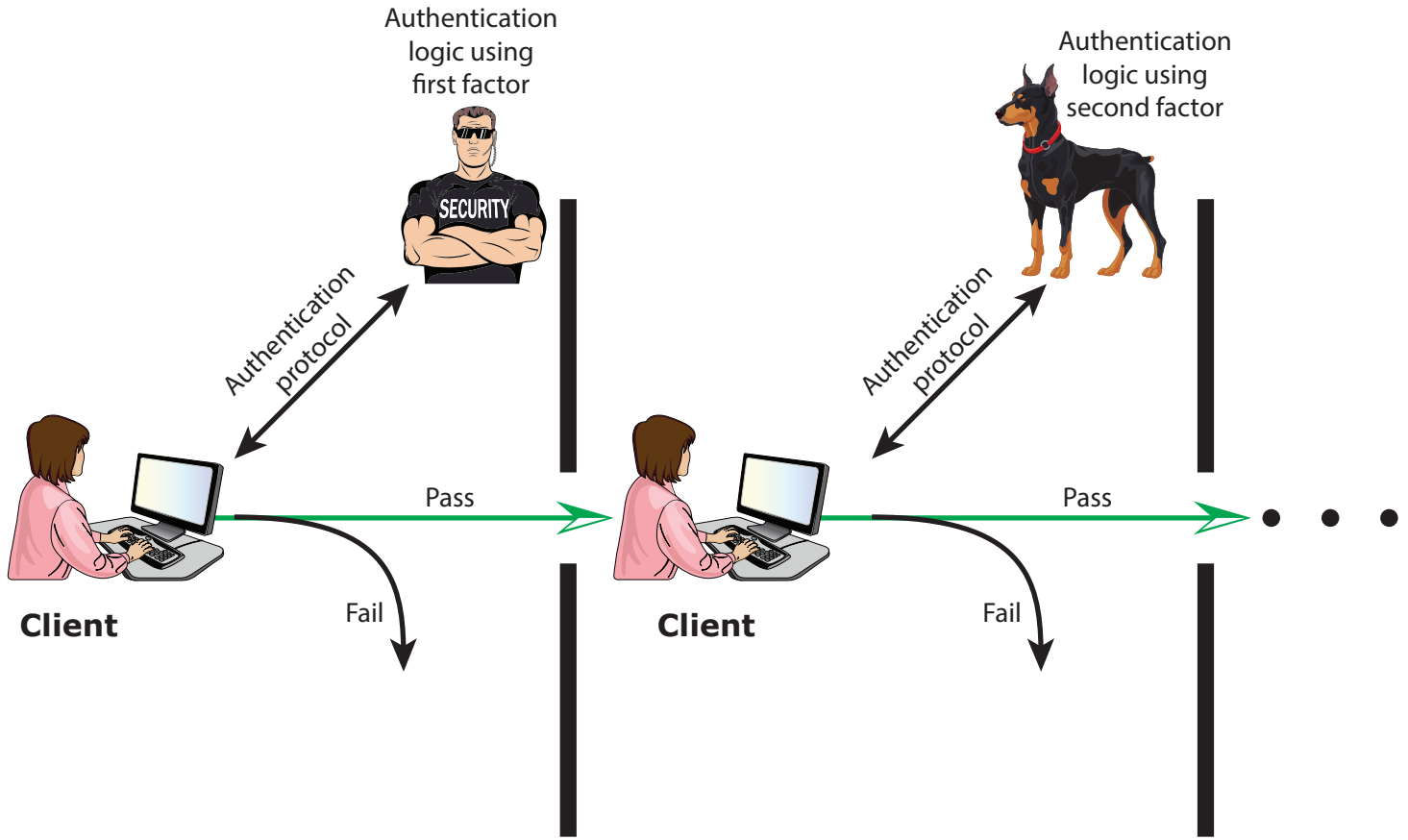
Something the individual does (dynamic biometrics)

- Voice pattern, handwriting, typing rhythm

Means of Authentication: considerations

- All of these methods, if properly implemented and used, can provide secure user authentication
- However, **each method has problems**
  - Passwords, tokens
    - 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
    - There is a significant administrative overhead on systems for managing and securing password and token information
  - Biometric authenticators
    - Suffer from a variety of problems, including dealing with false positives and false negatives, user acceptance, cost, and convenience

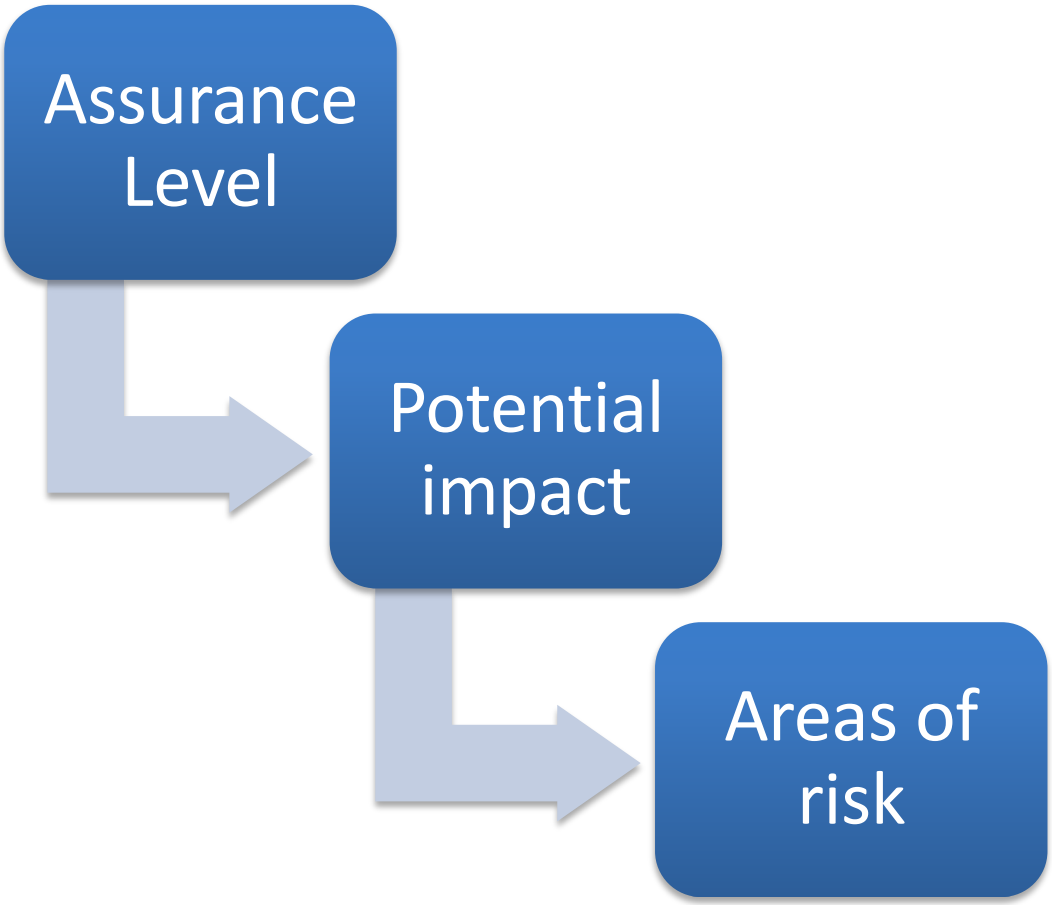
Multifactor Authentication



- 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 …, and so on

Risk Assessment for User Authentication

There are three separate concepts:



Assurance Level

Describes an organization's **degree of certainty** that a user has presented a credential that refers to his or her identity

More specifically is defined as:

The degree of confidence in the **process used to establish the identity** of the individual to whom the credential was issued

The degree of confidence that **the individual who uses the credential** is the individual to whom the credential was issued

Four levels of assurance in the asserted identity's validity

- **Level 1:** Little or no confidence: typical authentication techniques require a *user-supplied ID* and *password*
  - A consumer registering to participate in a discussion at a company web site discussion board
- **Level 2:** Some confidence: some sort of *secure authentication protocol* needs to be used, together with one of the four basic means of authentication
  - Wide range of business with the public where organizations require an initial identity assertion
- **Level 3:** High confidence: typical authentication techniques require *multifactor authentication*
  - Clients or employees accessing restricted services of high value but not the highest value
    - A patent attorney electronically submits confidential patent information to the U.S. Patent and Trademark Office
    - Improper disclosure would give competitors a competitive advantage
- **Level 4:** Very high confidence: typical authentication techniques require the use of *multiple factors* as well as *in-person registration*
  - Clients or employees accessing restricted services of very high value or for which improper access is very harmful
    - A law enforcement official accesses a law enforcement database containing criminal records
    - Unauthorized access could raise privacy issues and/or compromise investigations

Potential Impact

FIPS 199 defines three levels of potential impact on organizations or individuals should there be a breach of security (in our context, a failure in user authentication):

- **Low**: An authentication error could be expected to have a limited adverse effect on organizational operations, organizational assets, or individuals
  - E.g. the error might cause a degradation in mission capability to an extent and duration that the organization is able to perform its primary functions, but the effectiveness of the functions is noticeably reduced
- **Moderate**: an authentication error could be expected to have a serious adverse effect
  - E.g. the error might cause a significant degradation in mission capability to an extent and duration that the organization is able to perform its primary functions, but the effectiveness of the functions is significantly reduced
- **High**: An authentication error could be expected to have a severe or catastrophic adverse effect
  - E.g. the error might cause a severe degradation in or loss of mission capability to an extent and duration that the organization is not able to perform one or more of its primary functions

Areas of Risk

- For a given asset, an organization needs to determine
  - the different ***risk areas***, or **categories of impact**, that are of concern
  - for each area, the *level of potential impact* should an authentication failure occur
- E.g. consider the **financial loss** if there is an authentication error that results in unauthorized access to a database; depending on the nature of the database, the potential impact could be:
  - **Low:** At worst, an insignificant unrecoverable financial loss to any party, or an insignificant organization liability
  - **Moderate:** At worst, a serious unrecoverable financial loss to any party, or a serious organization liability
  - **High:** severe or catastrophic unrecoverable financial loss to any party, or severe or catastrophic organization liability
- The **analyst** can then pick an assurance level that meets or exceeds the requirements for assurance in each area

Areas of Risk

- The mapping between the potential impact and the appropriate level of assurance that is satisfactory to deal with the potential impact depends on the risk area
- A possible mapping for various risk areas that an organization may be exposed to is below
  - According to it, if any of the risk areas has a potential impact of high, or if the personal safety area has a potential impact of moderate or high, then assurance level 4 should be implemented

	Assurance Level based on Potential Impact			
Risk Areas for Authentication Errors	1	2	3	4
Inconvenience, distress, or damage to reputation	Low	Mod	Mod	High
Financial loss or organization liability	Low	Mod	Mod	High
Harm to organization programs or interests	None	Low	Mod	High
Unauthorized release of sensitive information	None	Low	Mod	High
Personal safety	None	None	Low	Mod/High
Civil or criminal violations	None	Low	Mod	High

Index

- Digital User Authentication Principles
- Password-Based Authentication
- Token-Based Authentication
- Biometric Authentication
- Remote User Authentication
- Security Issues for User Authentication
- Practical Application: An Iris Biometric System
- Case Study: Security Problems for ATM Systems

Password Authentication

- **Widely used** line of defense against intruders
  - User provides an identifier (ID) and a password
  - System compares password with the one stored for that specified ID
- The password serves to authenticate the ID of the individual logging on to the system
- The **ID provides security** in the following ways
  - Determines whether the user is authorized to access the system
  - Determines the user's privileges (e.g. supervisory or "superuser", normal user, guest/anonymous accounts)
  - Is used in discretionary access control
- The most commonly used user authentication technique, despite its many security vulnerabilities

Password Vulnerabilities

- Typically, a system that uses password-based authentication maintains a password file indexed by user ID
  - One technique is to store not the user's password but the value resulting from application of a one-way hash function to the password
- We can identify the following main forms of attack against password-based authentication
  - 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

Some attack strategies and countermeasures

- **Offline dictionary attack:** the attacker bypasses the access control system, 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 (e.g. 5) of failed login 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
  - Countermeasures include policies to inhibit the selection by users of common passwords, scanning the IP addresses of authentication requests

Some attack strategies and countermeasures

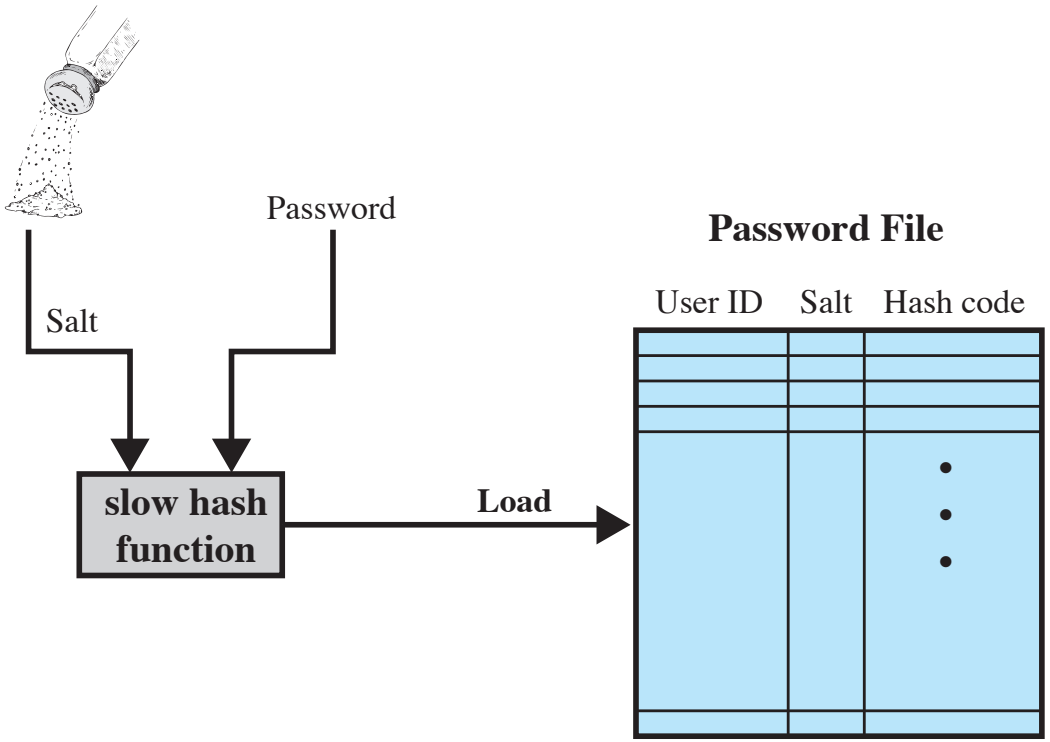
- **Password guessing against single user:** the attacker attempts to gain knowledge about the account holder and system password policies and uses that knowledge to guess the password
  - Countermeasures include training in and enforcement of password policies that make passwords difficult to guess
  - Such policies address the secrecy, minimum length of the password, character set, prohibition against using well-known user identifiers, length of time before the password must be changed
- **Exploiting user mistakes**: passwords written down because difficult to remember, or intentionally shared to enable a colleague to share files, or preconfigured with a shipped computer system, or obtained by using social engineering tactics that trick the user into revealing the password
  - Countermeasures include user training, intrusion detection, simpler passwords combined with an additional authentication mechanism

Hashed Passwords: UNIX Password Scheme

- A widely used password security technique is the use of **hashed passwords** and a **salt value** (i.e. a pseudorandom or random number)
- Found on virtually all UNIX variants and many other OSs
- Shown to be secure against a variety of cryptanalytic attacks

Loading a new password

- The user selects or is assigned a password
- This password is combined with a fixed-length salt value
- Password and salt are input to a hash algorithm designed to be slow to execute for thwarting attacks
- The hashed password is then stored, together with a plaintext copy of the salt, in the password file for the corresponding user ID

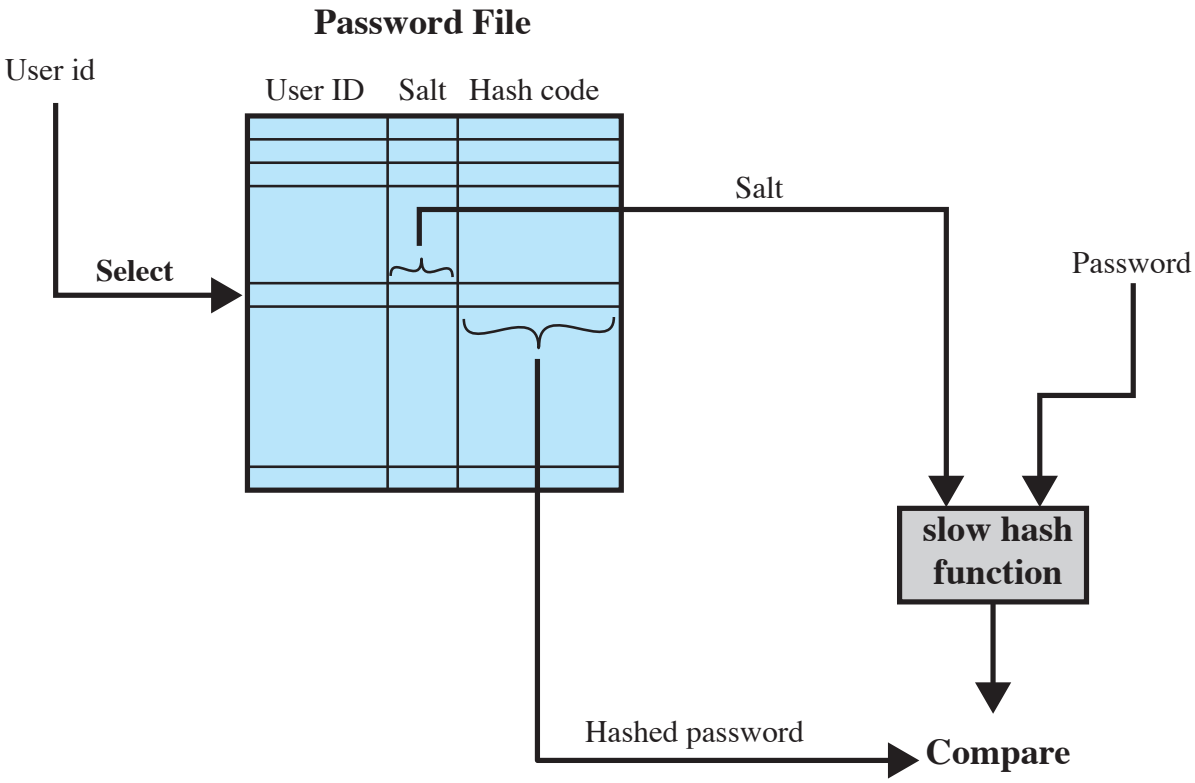


Hashed Passwords: UNIX Password Scheme

- A widely used password security technique is the use of **hashed passwords** and a **salt value** (i.e. a pseudorandom or random number)
- Found on virtually all UNIX variants and many other OSs
- Shown to be secure against a variety of cryptanalytic attacks

Verifying a password

- When a user attempts to log on to the system, the user provides an ID and a password
- The OS uses the ID to index into the password file and retrieve the plaintext salt and the hashed password
- The salt and user-supplied password are used as input to the hash algorithm
- If the result matches the stored value, the password is accepted

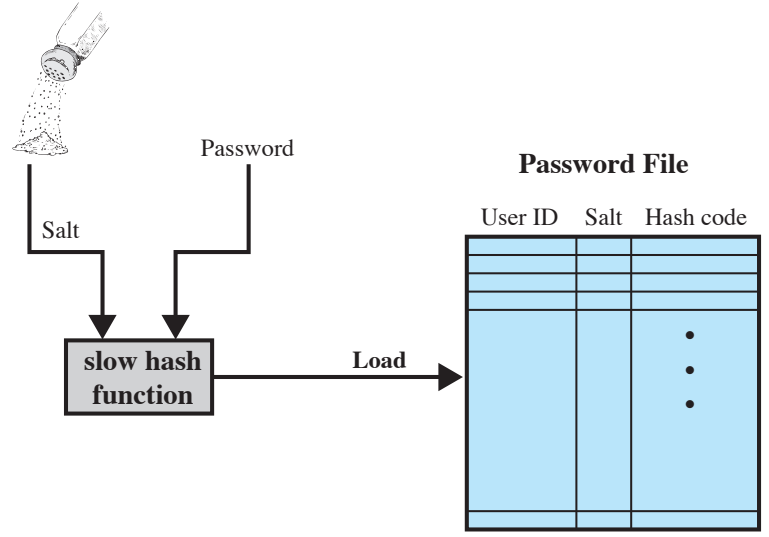


Purposes of using a salt value

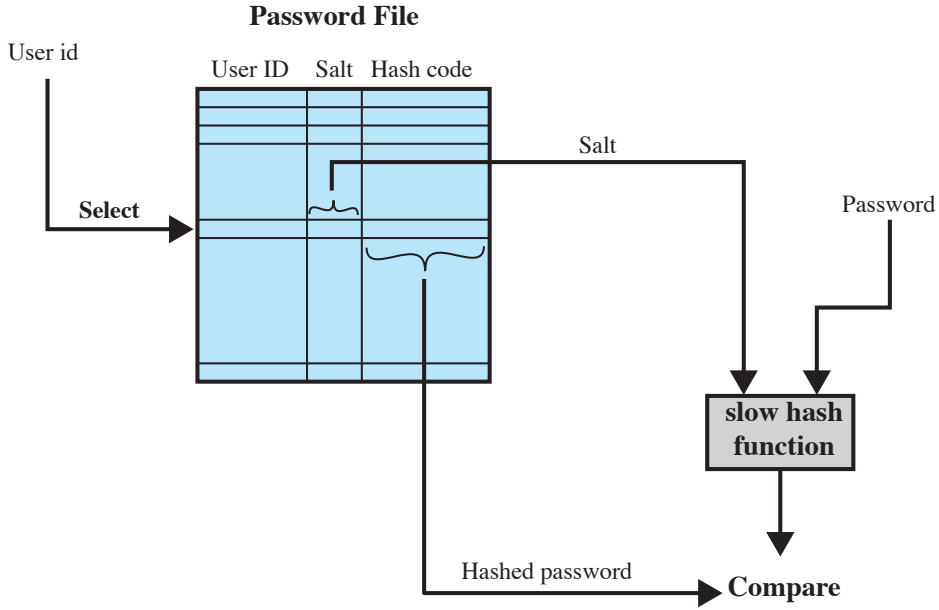
- Prevents duplicate passwords from being visible in the password file
  - Even if two users choose the same password, those passwords will be assigned different salt values, hence the hashed passwords of the two users will differ
- Makes nearly impossible to find out whether a user with passwords on two or more systems has used the same password on all of them
- Increases the difficulty of dictionary-based attacks (see later)
  - For a salt of length b bits, the number of possible passwords is potentially increased by a factor of 2^b, increasing the difficulty of guessing a large number of passwords in a dictionary attack
- The salt does not need to be secret
  - The randomization of the password hashes using salts makes the some attacks (e.g. the rainbow attack) too expensive to be worthwhile

Possible threats to UNIX Password Scheme

- 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
- If an opponent is able to obtain a copy of the password file, then a password 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

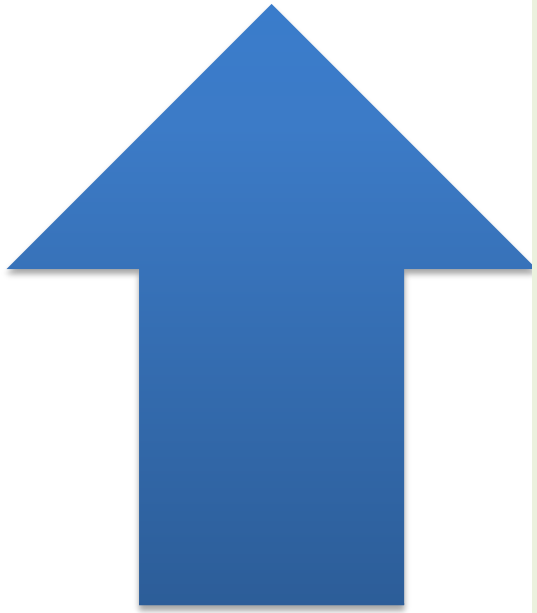


(a) Loading a new password



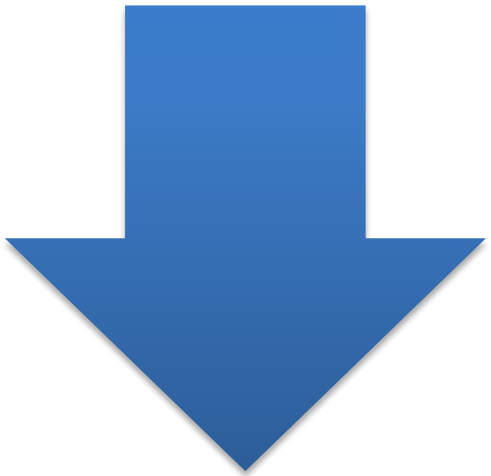
(b) Verifying a password

Original UNIX Implementation



Original scheme

- Each user selects a password of up to 8 printable characters
- This is converted into a 56-bit value (using 7-bit ASCII) that serves as the key input to a hash routine based on DES
- The hash routine also takes a 12-bit salt value and is executed with a data input consisting of a 64-bit block of 0s
- Encryption is repeated 25 times by taking the output of an iteration as input of the next one
- The resulting 64-bit output is then translated into an 11-character sequence



Now regarded as inadequate

- Due to dictionary attacks (supercomputers are able to process tens of millions password guesses in about one hour)
- Still often required for compatibility with existing account management software or in multivendor environment

Improved UNIX Implementations

- There are other, much stronger, hash/salt schemes available for Unix
- For many UNIX systems, the recommended hash function is based on the **MD5** secure hash algorithm
  - 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 a hash function, called **Bcrypt**, based on the Blowfish symmetric block cipher
  - Probably the most secure version of Unix hash/salt scheme
  - Bcrypt allows passwords of up to 55 characters in length
  - Uses 128-bit salt to create 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

Password Cracking: Traditional Approaches

Dictionary attacks

- Develop a large dictionary of possible passwords and try each against the password file
- In contrast to a brute force attack, a dictionary attack tries only those possibilities which are deemed most likely to succeed
- Each password must be hashed using each salt value and then compared with stored hash values
- If no match, try variations (e.g. backwards spelling of words, additional numbers or special characters, or sequence of characters)

Rainbow table attacks

- Trade-off space for time
- Pre-compute a mammoth table of hash values for a large number of dictionary words and salts combinations
- Requires a considerable amount of preparation time, but allows the actual attack to be executed faster
- 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
- Remedy: require that all passwords be, at least, 8 characters long

John the Ripper (http://openwall.com/john/pro/)

- Open-source password cracker first developed in 1996
- Uses a combination of brute-force and dictionary techniques

Using salts to counter dictionary attacks

- *Salts make dictionary attacks and brute-force attacks for cracking large numbers of passwords much slower*
- The use of salt values doesn't make a single hashed password stronger, rather it fixes up some weaknesses that appear when dealing with a collection of hashed passwords
  - While the attacker may be able to crack one password, cracking a large number of them will be unfeasible

Using salts to counter dictionary attacks

- If the attacker somehow gets *a single hashed password*, then adding a salt doesn't really slow the attacker down (assuming that he gets the salt as well; which is reasonable since the salt is commonly stored with the hashed password)
  - Without the salt, the attacker can guess various passwords, hash them, and see if any of the obtained hashes matches the target value
  - With the salt, the attacker simply includes that salt when he hashes his guesses, and so that doesn't make his job any harder
  - The attacker can still do a dictionary search on a single hashed password, but he has to do the computation after he has obtained the salt for the specific hashed password and he *can't reuse* that effort to attack a hashed password with a different salt
- However, if the attacker gets a *list of hashed passwords*, then adding a salt does make the attacker's job harder
  - Without the salt, the attacker can guess various passwords, hash them, and see if any of them hashes to a value on the list
    - If he has a collection of 1000 hashed passwords, he has just checked 1000 passwords for each single hash
  - On the other hand, if each hashed password includes a salt, and if each salt is different, then to check to see if a guessed password is correct, he has to run 1000 hashes (with each hash using a different salt)

Using salts to counter raibow table attacks

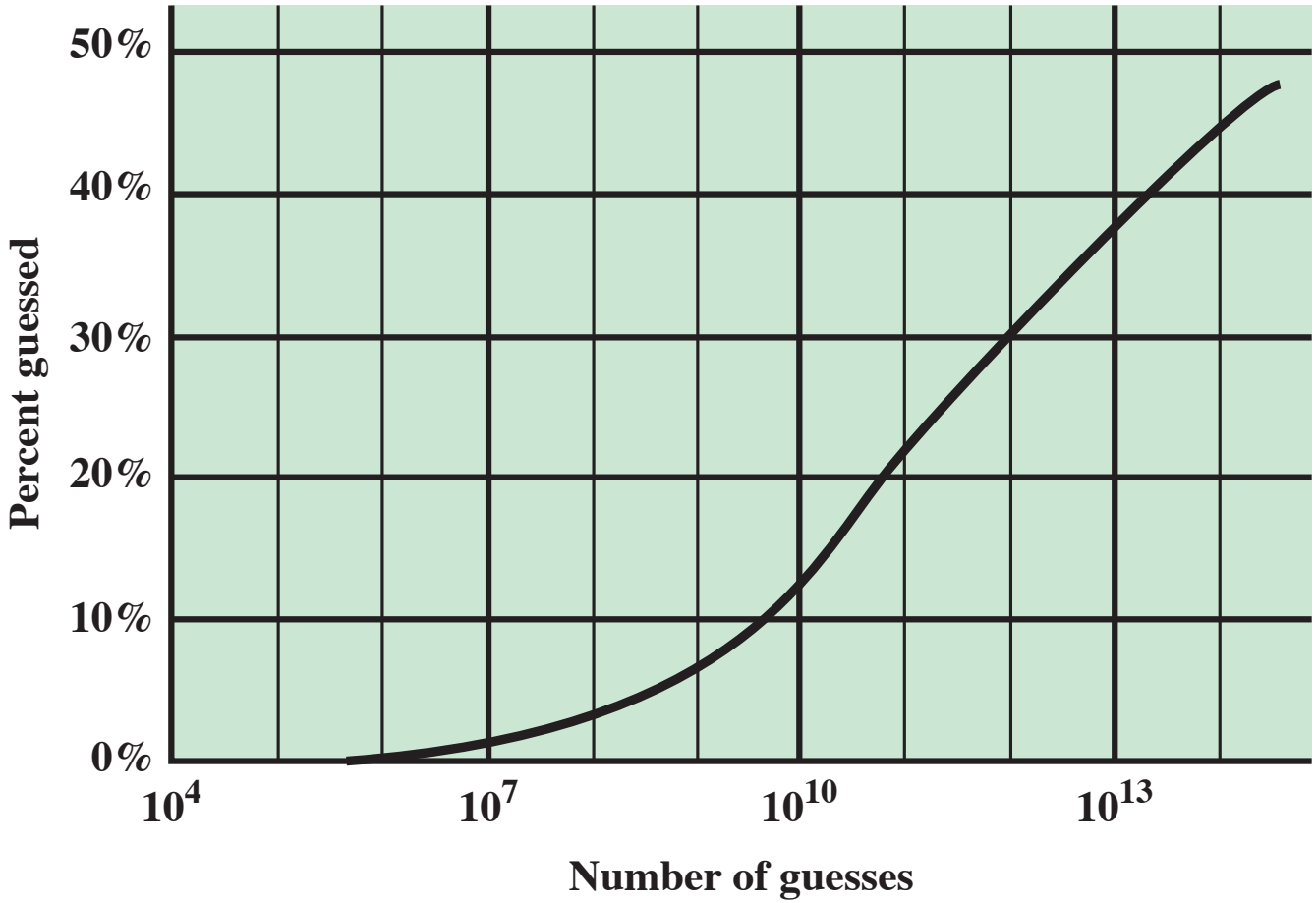
- A salt makes the use of precomputed tables, like e.g. rainbow tables, impractical because
  - the salt is another thing that the precompute method would need to guess, and
  - if the salt is of reasonable size, it can make the size of the rainbow table required for a successful attack prohibitively large
- Salts *mitigate rainbow table attacks* by forcing attackers to re-compute the tables using the salts
  - An attacker won't know in advance what the salt will be, so they can't pre-compute a rainbow table

Password Cracking: Modern Approaches

- Complex password policy
  - Forcing users to pick stronger passwords
- Password-cracking techniques have also improved
  - The processing capacity available for password cracking (e.g. GPU) has increased dramatically
  - Sophisticated algorithms (e.g. based on probabilities of letters in natural language) to generate potential passwords are now used
  - Numerous sets of leaked password files have become available for analysis and are used as training data
    - **Probabilistic approaches** have been developed that order the guesses according to their likelihood, based on e.g. the frequency of their character-classes in the training data

Percentage of Passwords Guessed after a given Number of Guesses

- E.g., a study has analysed the passwords used by over 25,000 students at a US research university with a complex password policy
- The analysts used the probabilistic approach and a database consisting of a collection of leaked password files
- The graph shows the percentage of passwords that have been recovered as a function of the number of guesses



Over 10% of the passwords are recovered after only 10^{10} guesses
After 10^{13} guesses, almost 40% of the passwords are recovered

Password File Access Control

Can block offline guessing attacks by denying access to encrypted passwords

Hashed passwords kept in a separate file from the user Ids (**shadow password file**), only accessible to privileged users

Vulnerabilities

- **Weakness in the OS that allows to bypass the access control system**
- **Accident of protection might render the password file readable**
- **Users with the same password on other systems**
- **Access to a backup of the password file in an archival disk**
- **Sniff user IDs and passwords in network traffic**

Password Selection Strategies

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

- Indeed, 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 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

Goal: to eliminate guessable passwords while allowing users to select passwords that are memorable

Four Basic 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
  - A good technique: use the first letter of each word of a phrase e.g. MsPi24yo = “My sister Peg is 24 years old”
- **Computer generated passwords**
  - Users have trouble remembering them
- **Reactive password checking**
  - System periodically runs its own password cracker program to find guessable passwords
  - In case, cancels any guessed passwords and notifies the user
- **Complex password policy (or proactive password checking)**
  - User can select their own password, but the system checks to see if the password is allowable
  - The aim is to strike a balance between user acceptability and password strength

Approaches to proactive password checking

- **Rule enforcement**. E.g., 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
  - Rule enforcement can be automated
  - It may not be sufficient to thwart password crackers since this scheme alerts crackers as to which passwords not to try
- **Password checker**. 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. Problems:
  - *Space*: The dictionary must be very large (tens of MB) to be effective
  - *Time*: The time required to search a large dictionary may be large

  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

Proactive password checking: Bloom Filter

- **Bloom filter**. An effective technique for rejecting words in a dictionary that has been implemented on a number of systems

A *Bloom filter of order k* is of a set of *k* independent hash functions

$$H_1(x), H_2(x), .. , H_k(x),$$

where each function maps a word in the dictionary into a hash value in the range *0* to *N –1,* i.e.

$$H_i(X_j) = y \quad \text{for} \quad 1 \le i \le k;\ 1 \le j \le D;\ 0 \le y \le N\text{-}1$$

where

X_j = *j*th word in password dictionary

D = number of words in password dictionary

Proactive password checking: Bloom Filter

- **Bloom filter**. An effective technique for rejecting words in a dictionary that has been implemented on a number of systems

A. The following procedure is then applied to the dictionary:
   1. A bit array (*hash table*) of *N* bits is used, with all bits initially set to *0*
   2. For each word in the dictionary, its *k* hash values are calculated, and the corresponding bits in the bit array are set to *1*
      E.g., if $H_i(X_j) = 67$ for some (i,j), then the sixty-seventh bit of the bit array is set to *1*; if the bit already has the value *1*, it remains at *1*
B. When a new password is presented to the checker, its *k* hash values are calculated
   - If all the corresponding bits of the bit array are equal to 1, then the password is rejected (because it is "possibly in the dictionary")
   - All words in the dictionary will be hence rejected
   - There will also be some false positives (that is, *words that are not in the dictionary but that produce a match in the bit array*)

Bloom Filter: false positive example

- Consider a scheme with two hash functions: $H_1()$ and $H_2()$
- Suppose that the words *undertaker* and *hulkhogan* are in the dictionary, but *xG%#jj98* is not
- Further suppose that

  $H_1(undertaker) = 25, \quad H_1(hulkhogan) = 83, \quad H_1(xG\%\#jj98) = 665$

  $H_2(undertaker) = 998, \quad H_2(hulkhogan) = 665, \quad H_2(xG\%\#jj98) = 998$
- If the password *xG%#jj98* is presented to the system, it will be rejected even though it is not in the dictionary
- Bloom Filters by Examples: https://llimllib.github.io/bloomfilter-tutorial/

Bloom Filter: minimizing false positive

- The hash scheme should be designed to minimize false positives, otherwise it will be difficult for users to select passwords
- It can be shown that the probability *P* of a false positive can be approximated by

$$P \approx (1 - e^{-kD/N})^k = (1 - e^{-k/R})^k$$

where

k = number of hash functions

N = number of bits in the bit array

D = number of words in the dictionary

R = N/D, ratio of bit array size (bits) to dictionary size (words)

- Thus, P (the probability of a false positive)
  - decreases as N (the number of bits in the array) increases, and
  - increases as D (the number of words in dictionary) increases

Probability of false positive

- Assume that a hash function selects each array position with equal probability
  - Thus the probability that a given array position is not selected is

$$1 - 1/N$$

- After all *D* words have been added to the Bloom filter, let *q* be the fraction of the *N* bits that are set to *0* (i.e. the number of bits still set to *0* is then *qN*)
  - The expected value of *q* is the probability that a given array position is left untouched by each of the *k* hash functions for each of the *D* words, which is

$$E[q] = (1 - 1/N)^{kD}$$

- Then, when testing membership of an element, for the array position given by any of the *k* hash functions the probability that the bit is found set to *1* is *1 − q*
  - So the probability that all k hash functions find their bit set to *1* is $(1 - q)^k$
- Thus, when testing membership of an element, the probability that all of the *k* array positions computed by the hash functions are set to *1* is

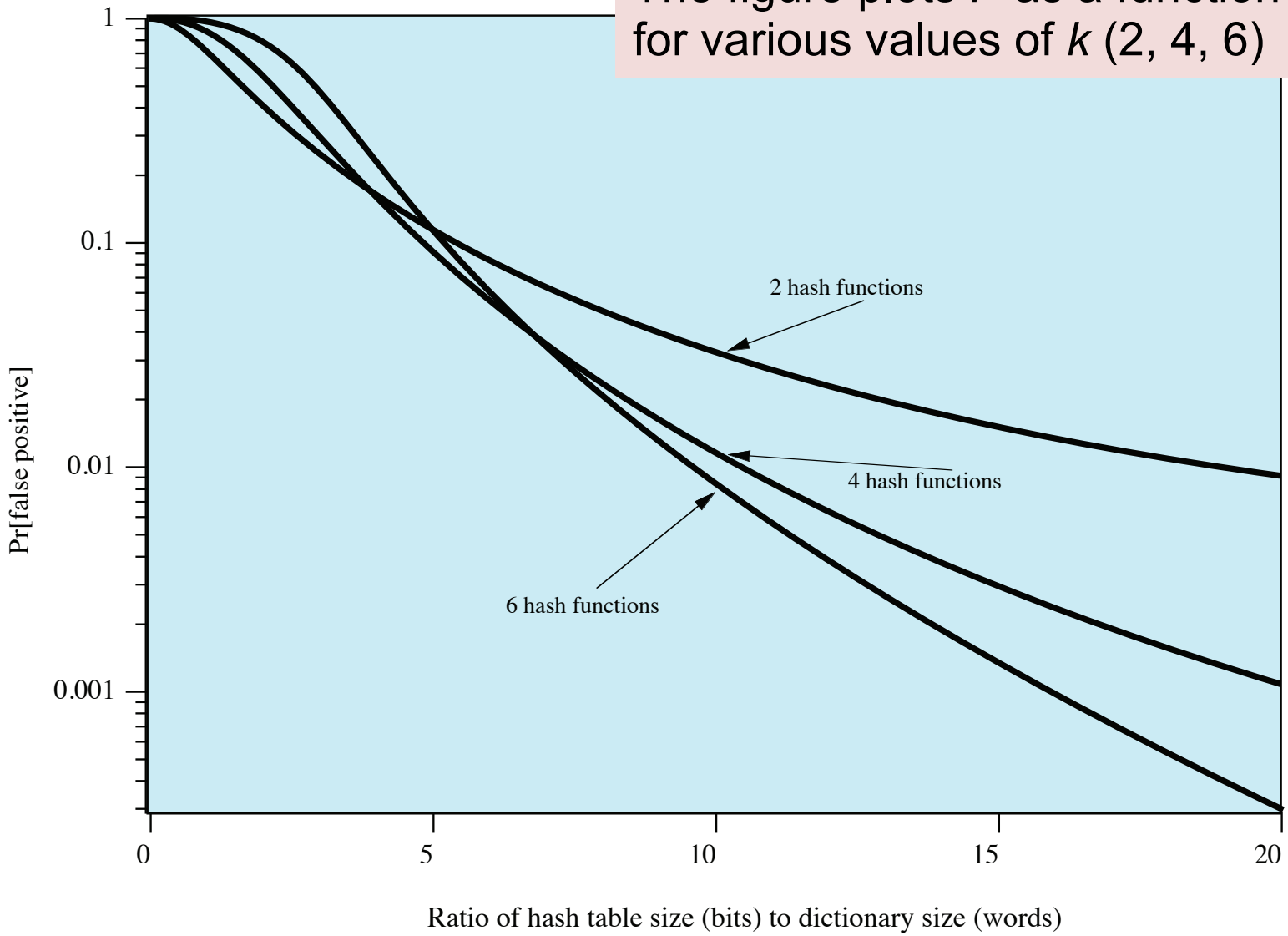
$$(1 - (1 - 1/N)^{kD})^k$$

  often given as

$$(1 - (1 - 1/N)^{kD})^k \approx (1 - e^{-kD/N})^k = (1 - e^{-k/R})^k$$

Performance of Bloom Filter

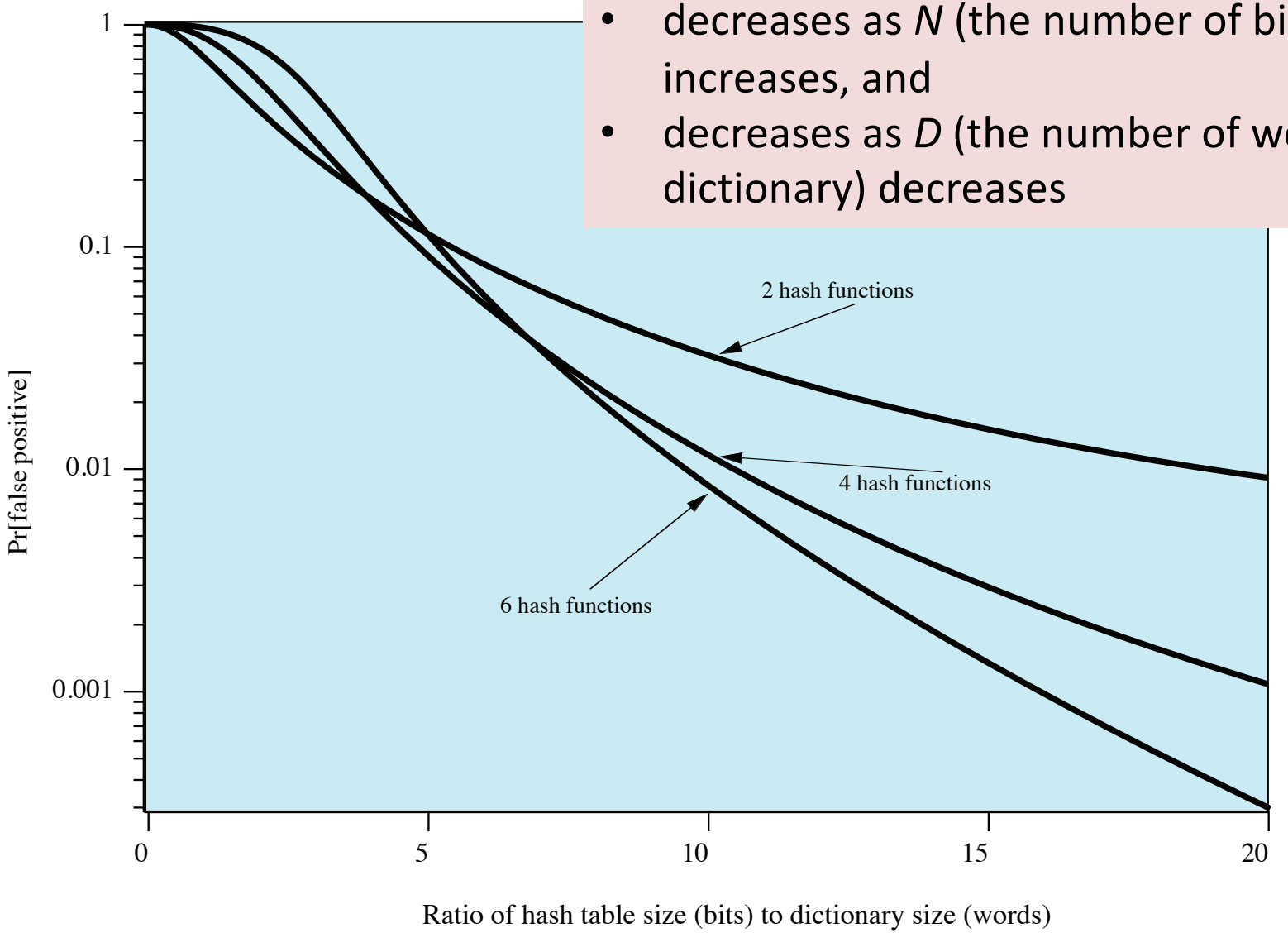
The figure plots *P* as a function of *R* for various values of *k* (2, 4, 6)



Performance of Bloom Filter

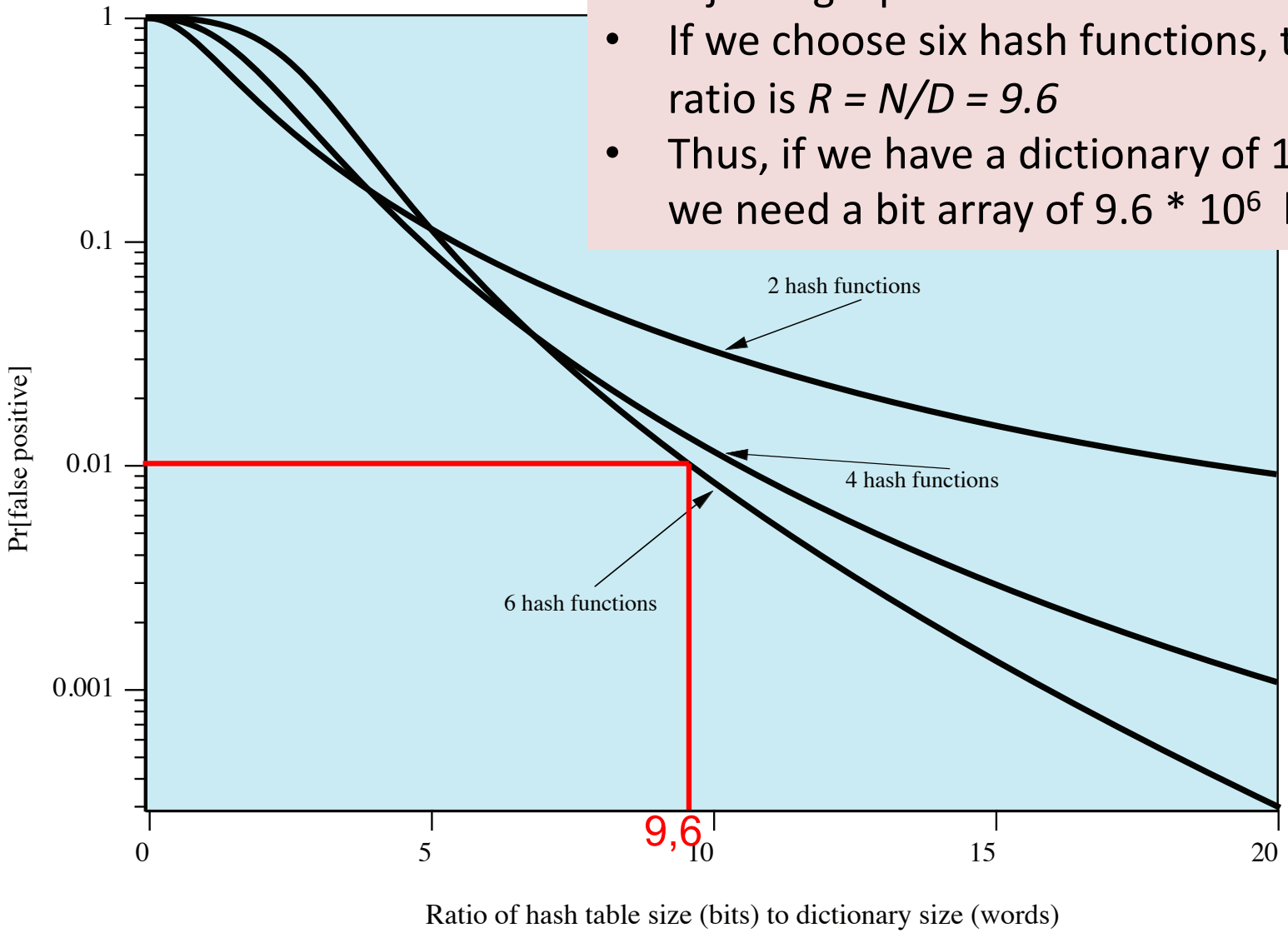
The probability of false positives

- decreases as $R = N/D$ increases, i.e.
- decreases as N (the number of bits in the array) increases, and
- decreases as D (the number of words in dictionary) decreases



Performance of Bloom Filter

- Suppose 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 = N/D = 9.6*
- Thus, if we have a dictionary of 1 million words, we need a bit array of 9.6 * 10^6 bits ≈ 1,2 MB



Performance of Bloom Filter

- Suppose 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 $R = N/D = 9.6$
- Thus, if we have a dictionary of 1 million words, we need a bit array of $9.6 * 10^6$ bits or about 1.2 MB of storage

- In contrast, assuming that each word requires 8 bytes, storage of the entire dictionary would require on the order of 8 MB
- **Advantage**: we achieve a compression of almost a factor of 7
- **Advantage**: 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 costs

Index

Token-based authentication

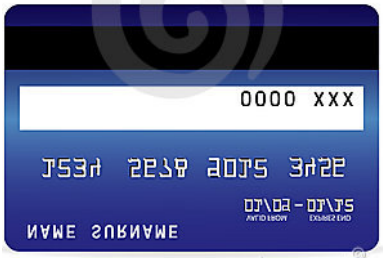
- Objects that a user possesses for the purpose of user authentication are called **tokens**
- We examine two types of tokens that are widely used
  - Memory cards
  - Smart tokens
- Both have the appearance and size of bank cards

Types of Cards Used as Tokens

Card Type	Defining Features	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 Electric contacts exposed on surface Radio antenna embedded inside	Biometric ID card

Old credit card

Bank card



Prepaid phone card

Biometric ID card

Memory Cards

- Can store but do not process data
- The most common is the bank card with a magnetic stripe on the back which can store only a simple security code
  - It can be read (and unfortunately reprogrammed) by an inexpensive card reader
- Can be used alone for physical access
  - Hotel rooms
- For authentication, a user provides both the memory card and some form of password or personal identification number (PIN)
  - A typical application is an automatic teller machine (ATM)
- When combined with a password or PIN, a memory card provides significantly greater security than a password alone
- **Drawbacks** of memory cards include:
  - Require a special reader
  - Loss of token: temporarily prevents its owner from gaining system access; if the token is found, stolen, or forged, then an adversary now need only determine the PIN to gain unauthorized access
  - User dissatisfaction: its use for computer access may be deemed inconvenient

Smart Tokens

These can be categorized along four dimensions that are not mutually exclusive

- *Physical characteristics*: 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*: include a keypad and display for human/token interaction
- *Electronic interface*: to communicate with a compatible reader/writer.
  A token may have one or both of the following types of interface:
  - Contact: must be inserted into a smart reader
  - Contactless: requires only close proximity to a reader. Both the reader and the token have an antenna, and the two communicate using radio frequencies
- *Authentication protocol*: it can be classified into three categories:
  - *Static*: the user authenticates himself or herself to the token and then the token authenticates the user to the computer
  - *Dynamic*: the token generates a unique password periodically (e.g., every minute); this password is then entered into the computer system for user authentication
  - *Challenge-response*: the computer system generates a challenge, such as a random string of numbers while 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 (as in digital signature)

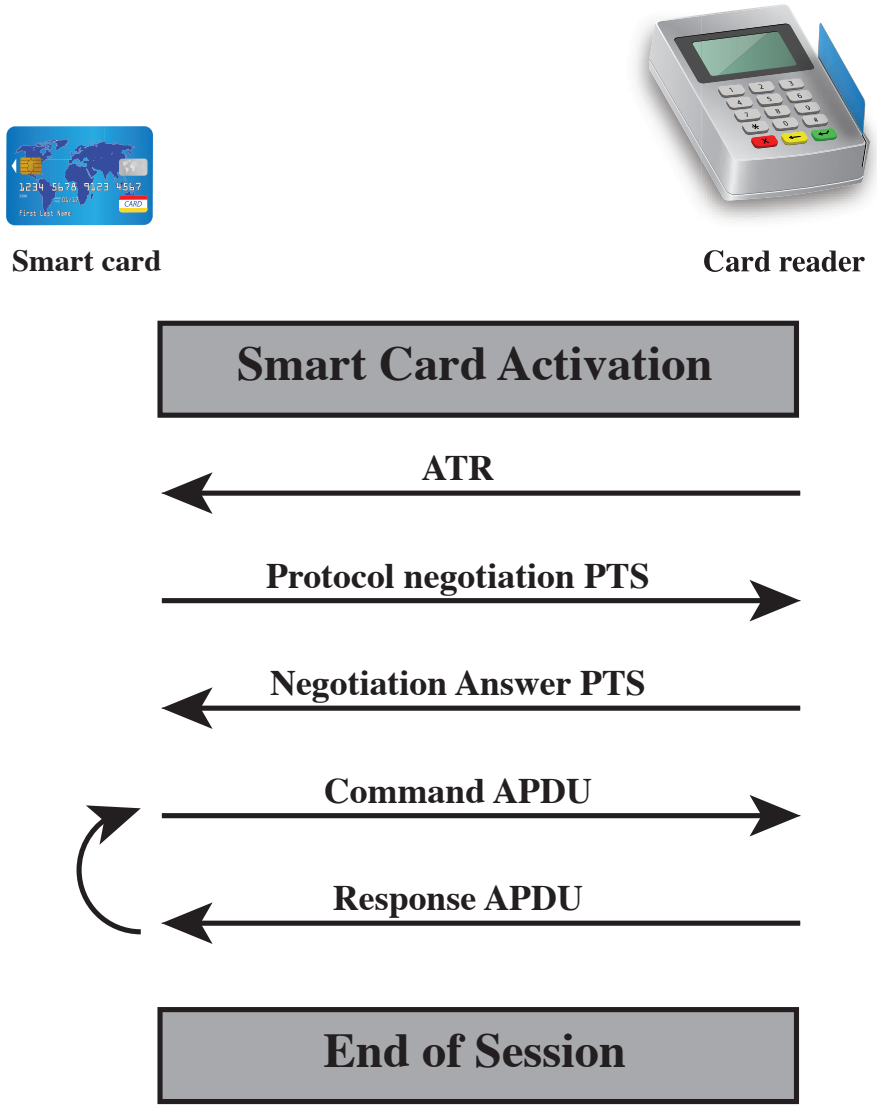
Smart Cards

- The most important category of smart token for user authentication
  - Has the appearance of a credit card
  - Has an electronic interface
  - May use any of the smart token protocols
- A smart card contains an entire microprocessor
  - Processor
  - Memory
  - I/O ports

- May also incorporate a special co-processing circuit for cryptographic operation to speed the task of
  - encoding and decoding messages
  - generating digital signatures to validate the information transferred
- 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

Smart Card/Reader Interaction

- Each time the card is inserted into a reader, a **reset** is initiated by the reader to initialize parameters, e.g. clock value
- After the reset function is performed, the card responds with an **answer to reset** (ATR) message defining 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 card's PTS response confirms the protocols and parameters to be used
- Terminal and card can now execute the **application protocol data unit** (APDU) to perform the desired application



APDU = application protocol data unit
ATR = Answer to reset
PTS = Protocol type selection

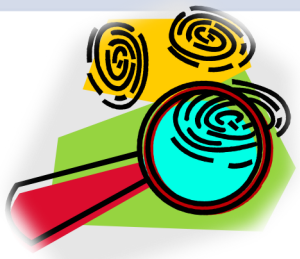
Electronic Identity Cards (eID)

Use of a smart card as a national identity card for citizens

It is a smart card that has been verified by the national government as valid and authentic

Can serve the same purposes as other national ID cards, for access to government and commercial services

Can provide strong proof of identity and can be used in a wider variety of applications



The 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): a six-digit decimal random number printed on the face of the card, usable as a password
- 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)

Use of a smart card as a national identity card for citizens

It is a smart card that has been verified by the national government as valid and authentic

Can serve the same purposes as other national ID cards, for access to government and commercial services

Can provide stronger proof of identity and can be used in a wider variety of applications

The most advanced deployment is the German card
neuer Personalausweis

Electronic Functions and Data for eID Cards

Three separate electronic functions, each with its own protected dataset

Function	Purpose	PACE Password	Data	Uses
ePass (mandatory)	Authorized offline inspection systems read the data (like electronic passport); it is not used over a network	CAN or MRZ	Face image; two fingerprint images (optional), MRZ data	Offline biometric identity verification reserved for government access
eID (activation optional)	Online applications read the data or access functions as authorized	eID PIN	Family and given names; artistic name and doctoral degree: date and place of birth; address; expiration date	Identification; age verification; on-line user authentication; revocation query
	Offline inspection systems read the data and update the address and community ID	CAN or MRZ		
eSign (certificate optional)	A certification authority installs the signature certificate online	eID PIN	Private key; X.509 certificate	Digital signature generation
	Citizens make electronic signature with eSign PIN	CAN		

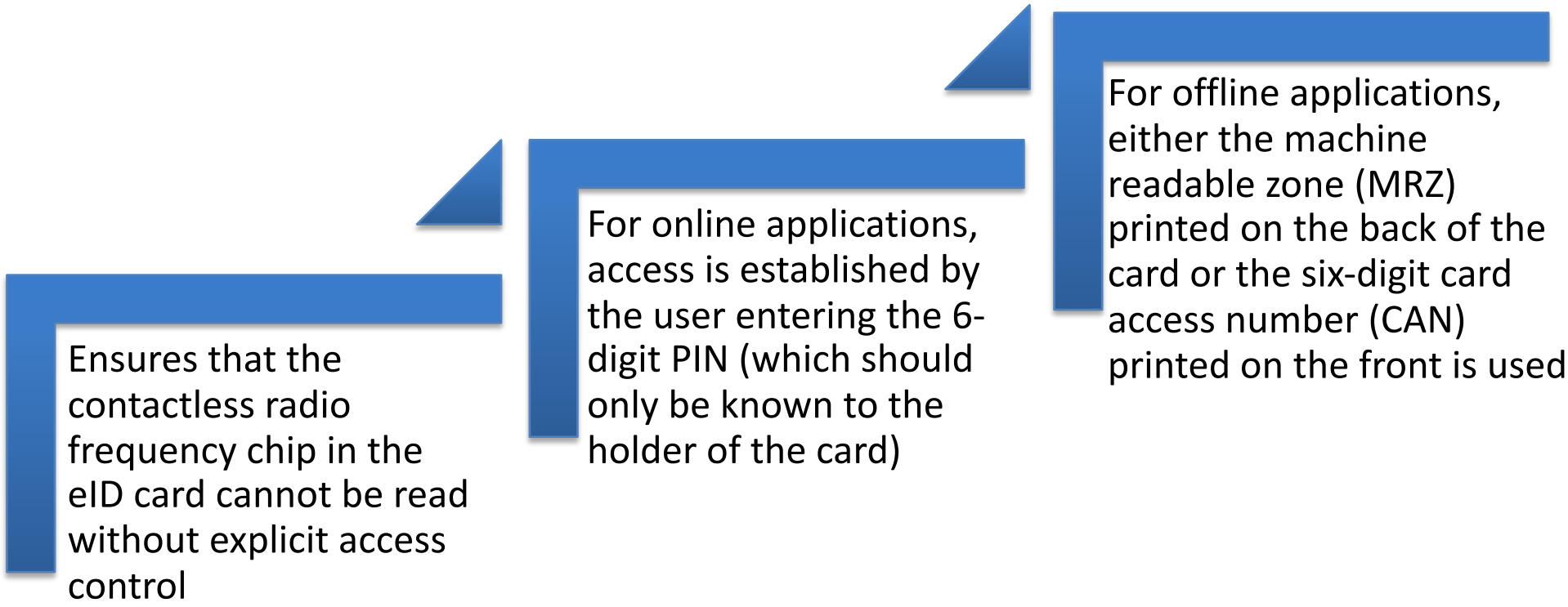
CAN = card access number

MRZ = machine readable zone

PACE = password authenticated connection establishment

PIN = personal identification number

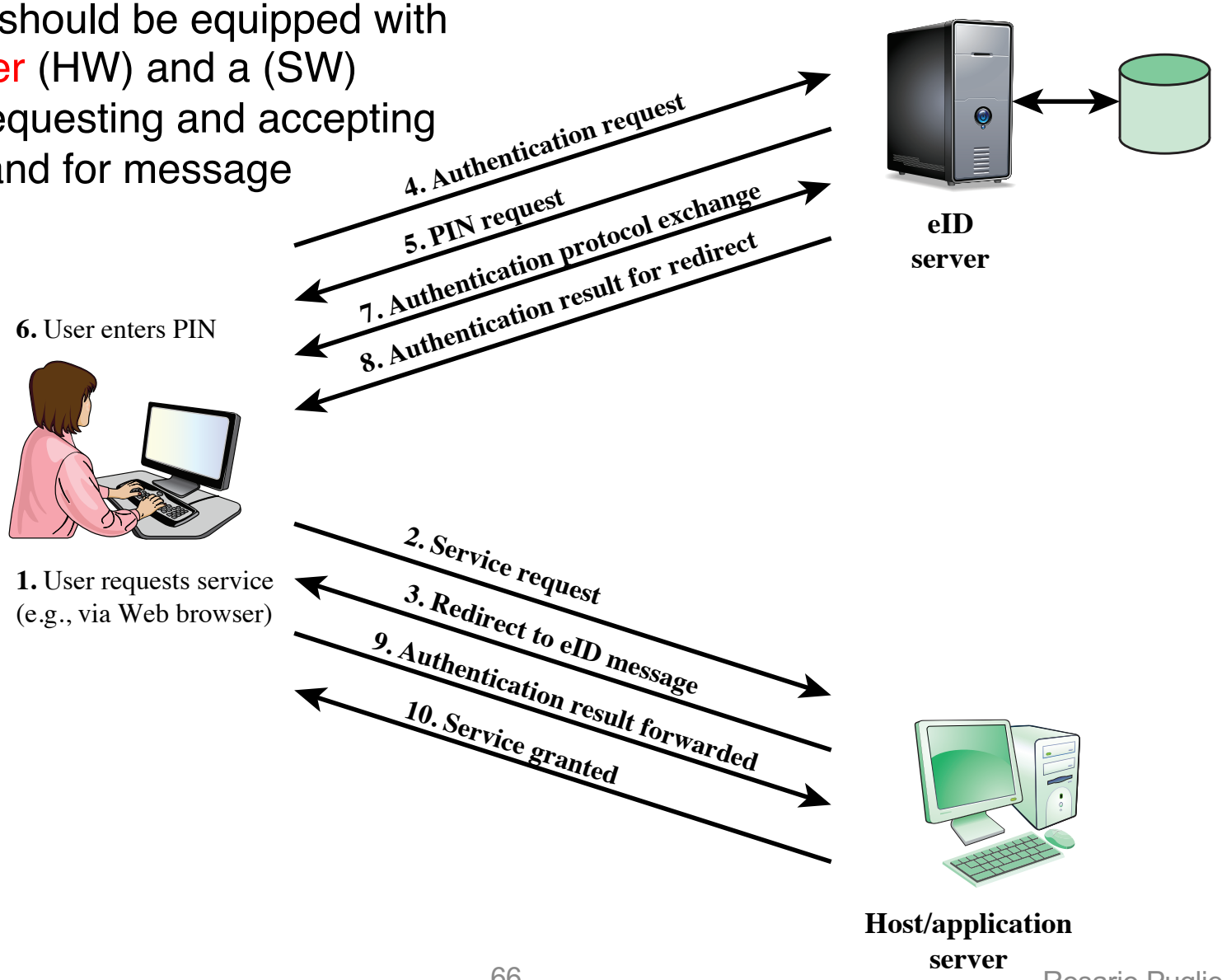
Password Authenticated Connection Establishment (PACE)



User Authentication with eID

A good example of online use of the eID function

The user system should be equipped with an eID card reader (HW) and a (SW) functionality for requesting and accepting the PIN number and for message redirection

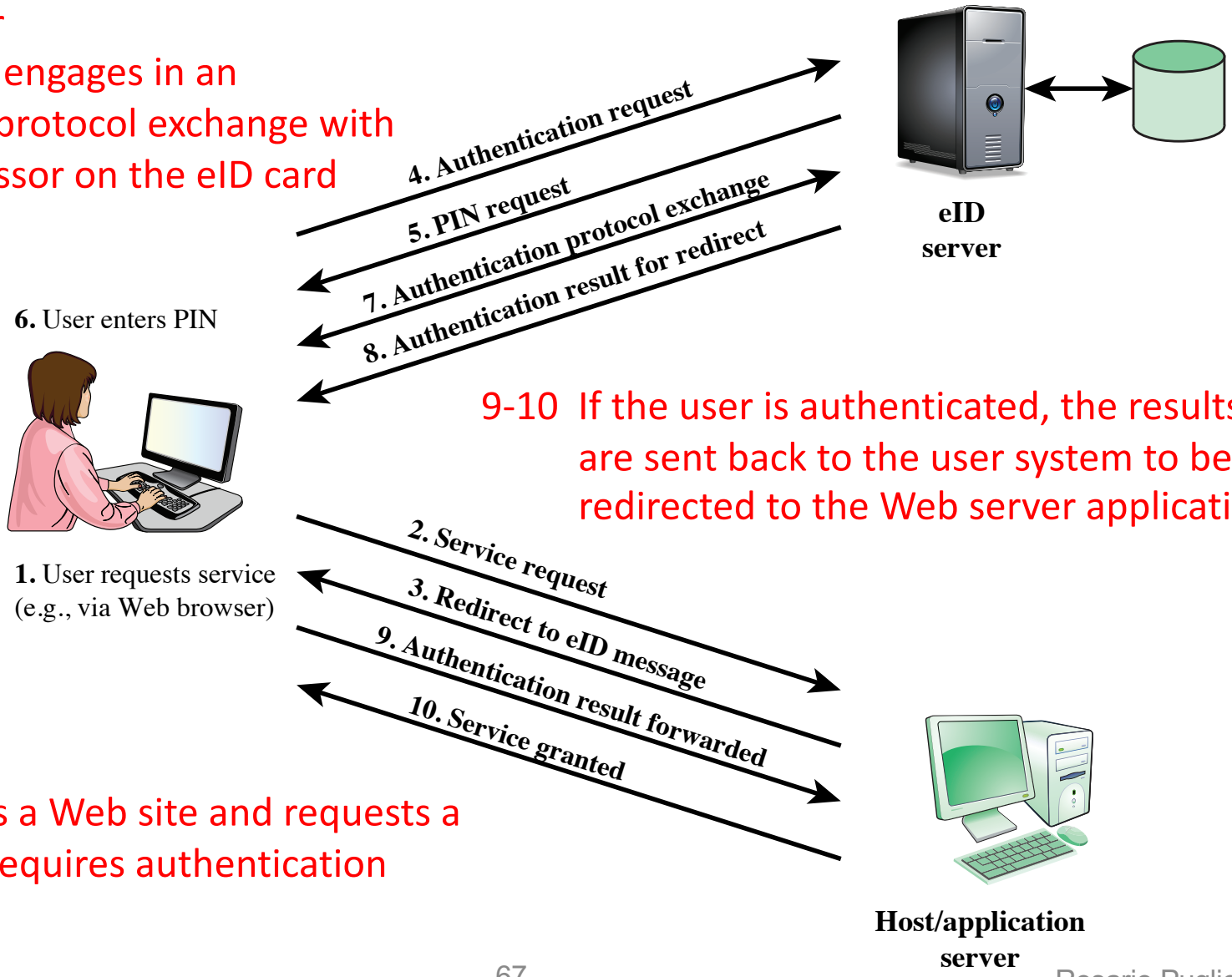


User Authentication with eID

A good example of online use of the eID function

4-6 eID server requests that the user enter the PIN number

7-8 eID server then engages in an authentication protocol exchange with the microprocessor on the eID card



1-3 eID user visits a Web site and requests a service that requires authentication

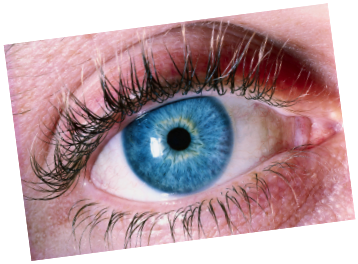
Index

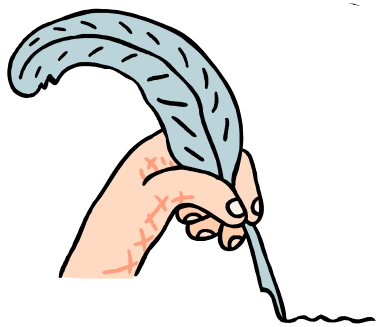
Biometric Authentication

- Attempts to authenticate an individual based on unique physical characteristics
- Based on *pattern recognition*
- Is technically more complex and expensive when compared to passwords and tokens
- Physical characteristics used include:

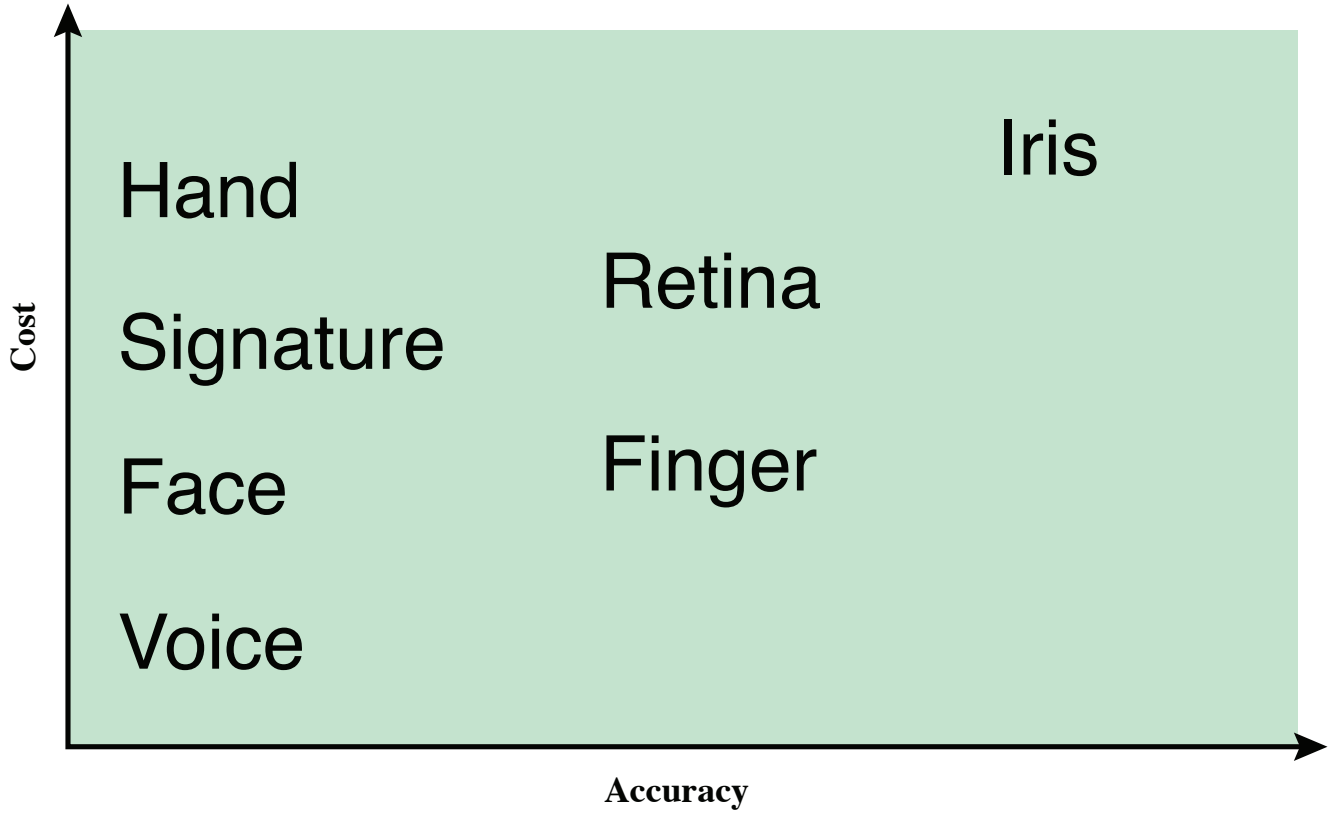
static
- Fingerprints
- Hand geometry
- Facial characteristics
- Retinal & Iris pattern

dynamic
- Signature
- Voiceprint





Cost vs. Accuracy of Various Biometric Characteristics

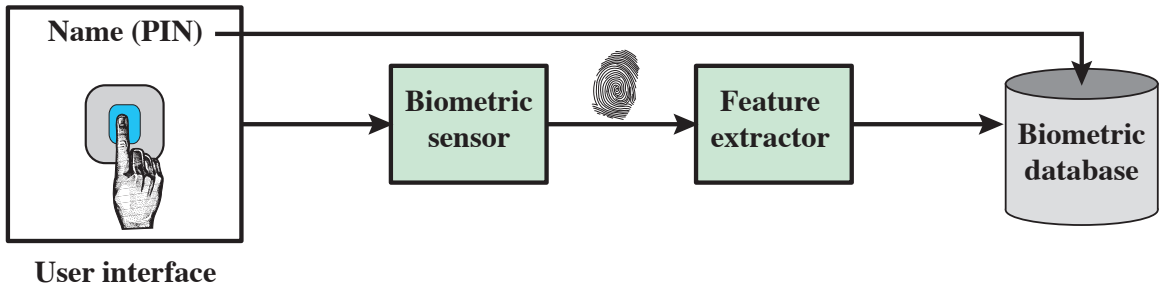


The concept of accuracy does not apply to user authentication schemes using smart cards or passwords

- 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

Operation of a Biometric Authentication System

Enrollment creates an association between a user (name and password/PIN) and the user's biometric characteristics



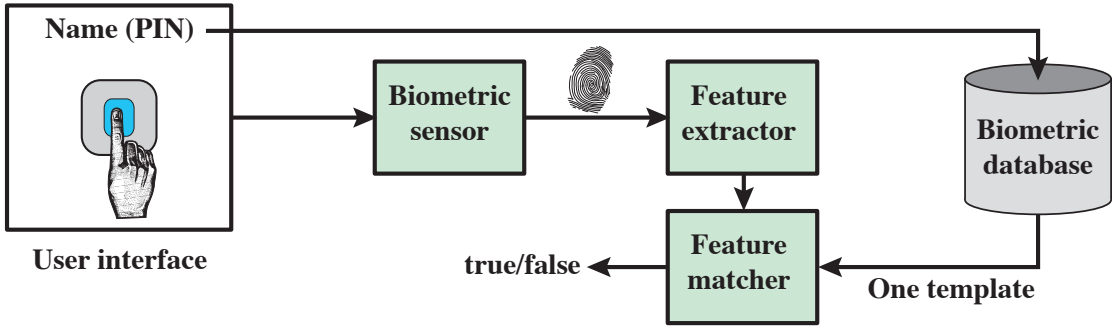
- Analogous to assigning a password to a user
- For a biometric system, the user presents a name and, typically, some type of password or PIN to the system
- At the same time the system senses some biometric characteristic of this user (e.g., fingerprint of right index finger)
- The system digitizes 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

Operation of a Biometric Authentication System

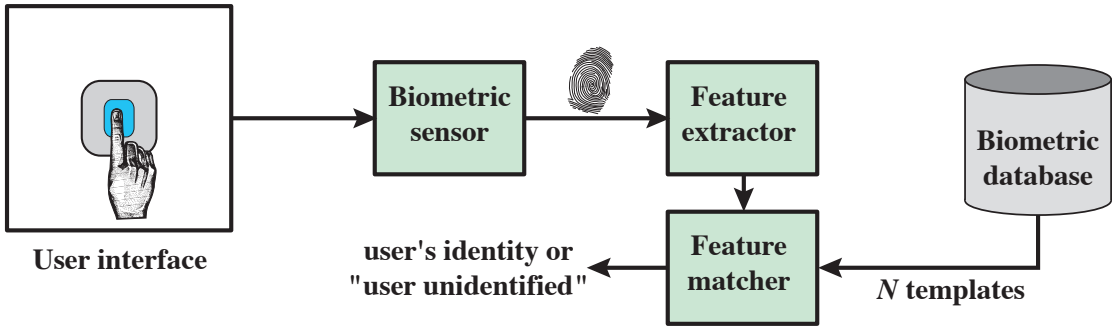
Depending on the application, user *authentication* involves

- **verifying** that a claimed user is the actual user
- **identifying** an unknown user

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

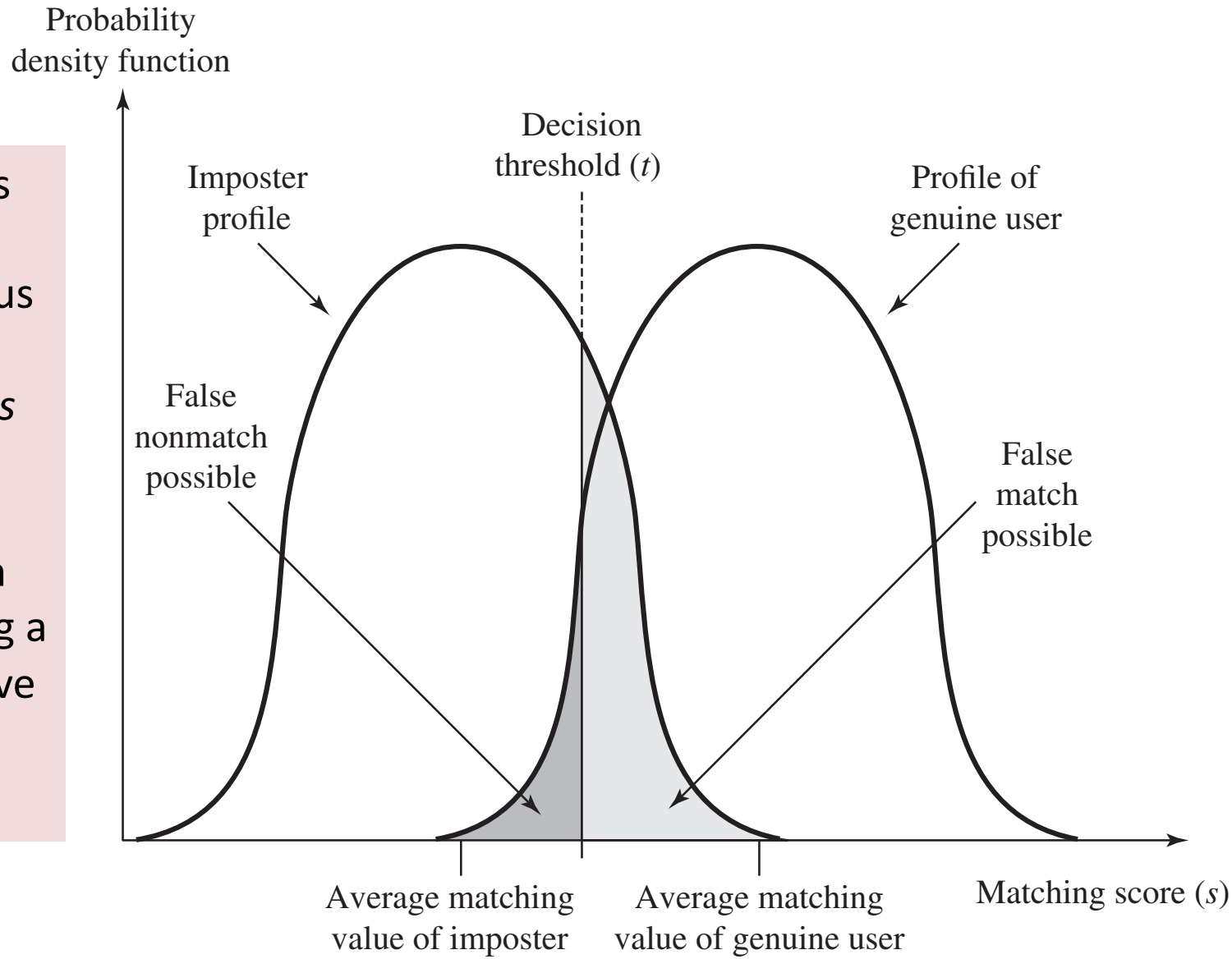


Biometric Accuracy

- In any biometric scheme, 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
- If a single user is tested by the system numerous times, the matching score will vary, with a probability density function typically forming a *bell-shaped curve*

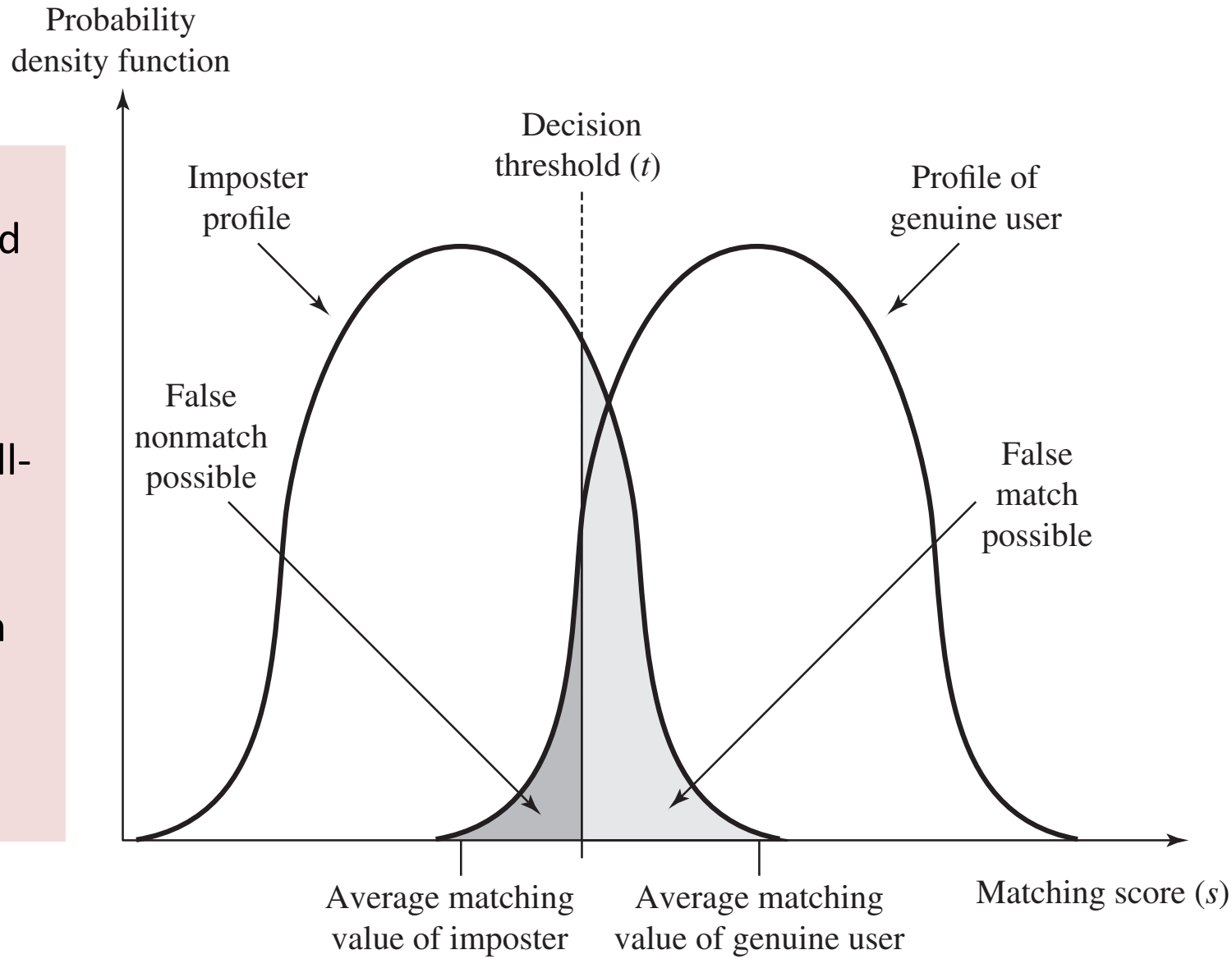
Profiles of a Biometric Characteristic of an Imposter and an Authorized User

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-shaped curve



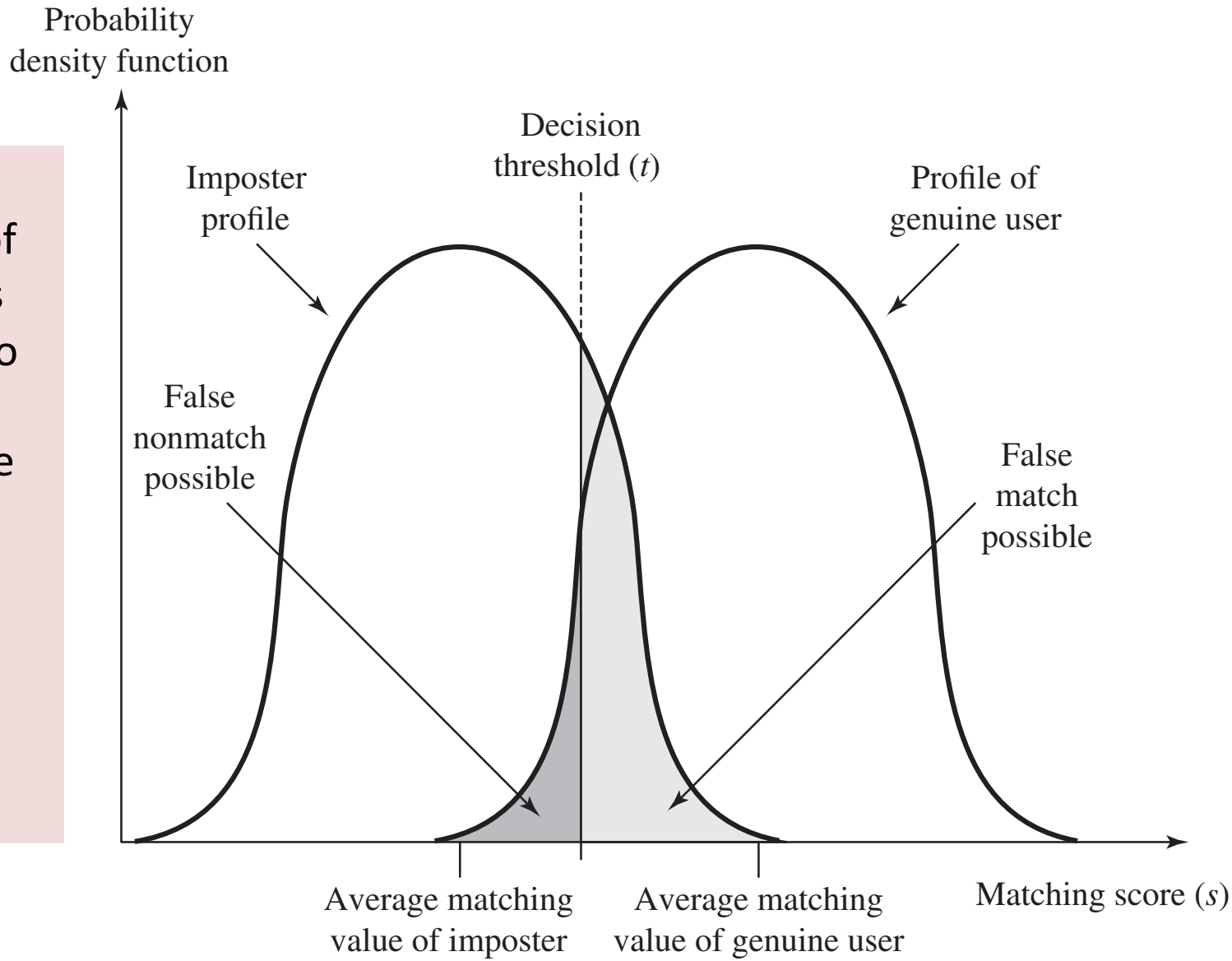
Profiles of a Biometric Characteristic of an Imposter and an Authorized User

Any other individual should have a much lower matching score but again will exhibit a bell-shaped probability density function



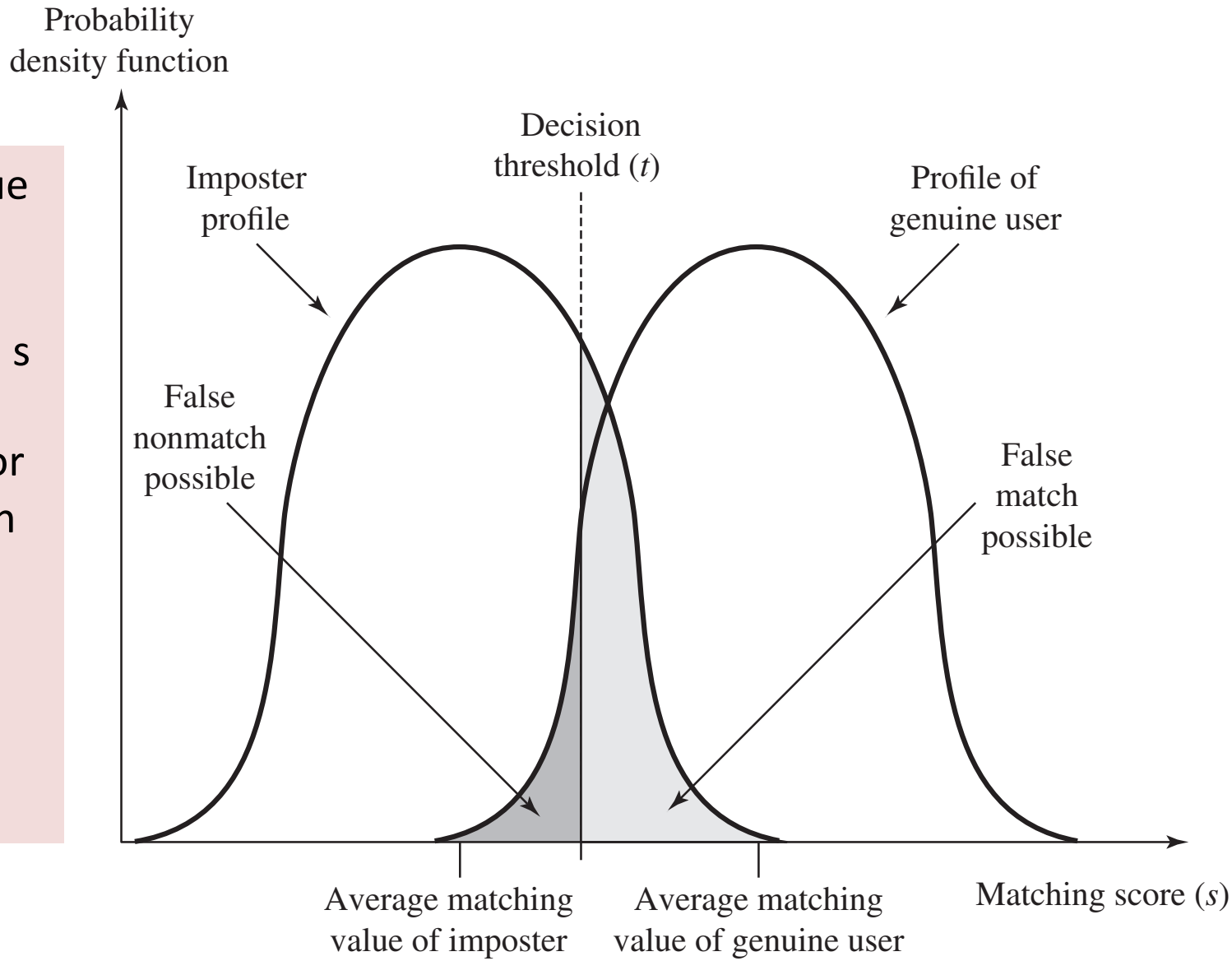
Profiles of a Biometric Characteristic of an Imposter and an Authorized User

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



Profiles of a Biometric Characteristic of an Imposter and an Authorized User

A threshold value t is selected so that if the presented value s ≥ t a match is assumed, and for $s < t$, a mismatch is assumed

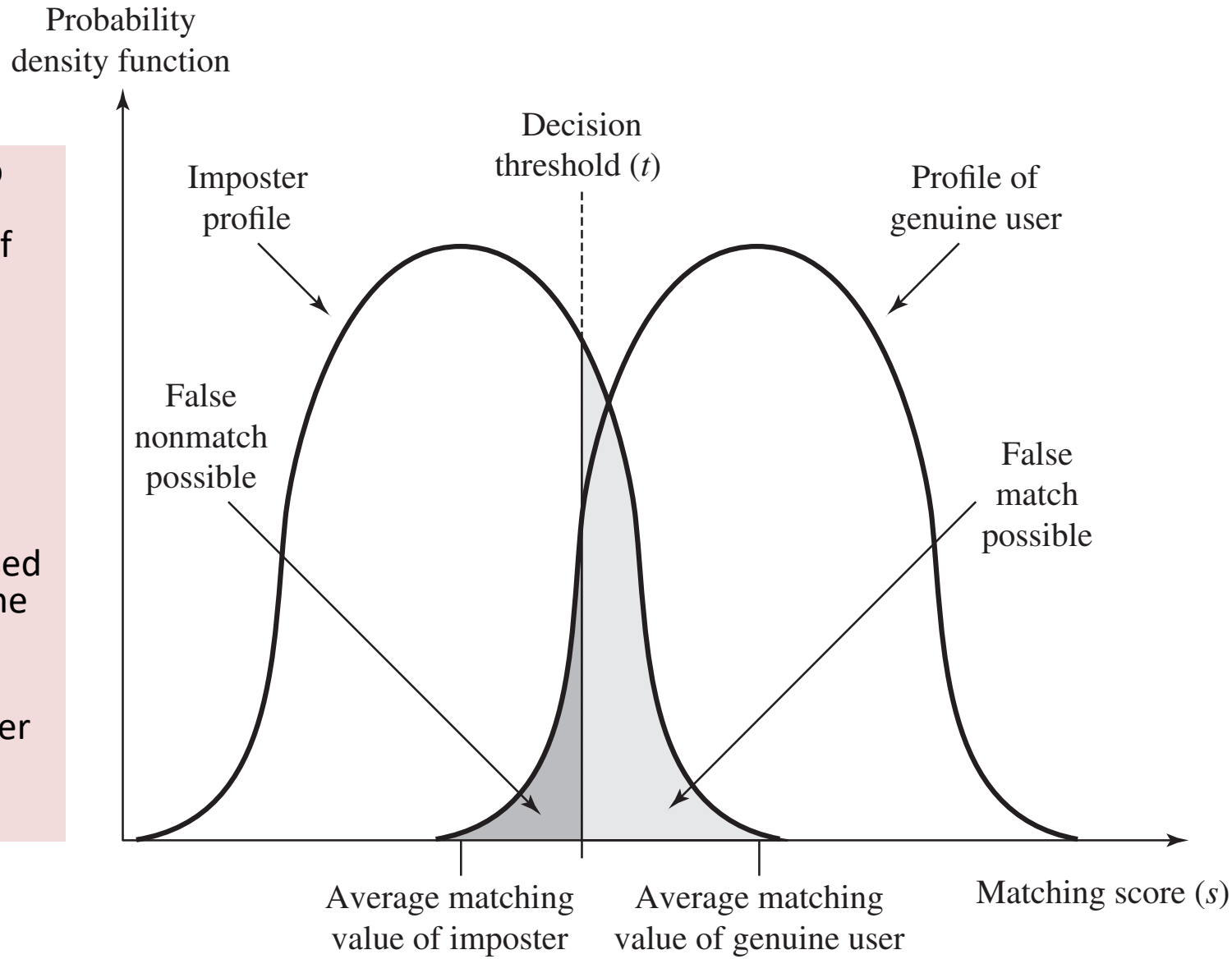


Profiles of a Biometric Characteristic of an Imposter and an Authorized User

The shaded part to the right of *t* indicates a range of values for which a **false match** is possible,

meaning that biometric samples from different sources are erroneously assessed to be from the same source,

resulting in the acceptance of a user who should not be accepted

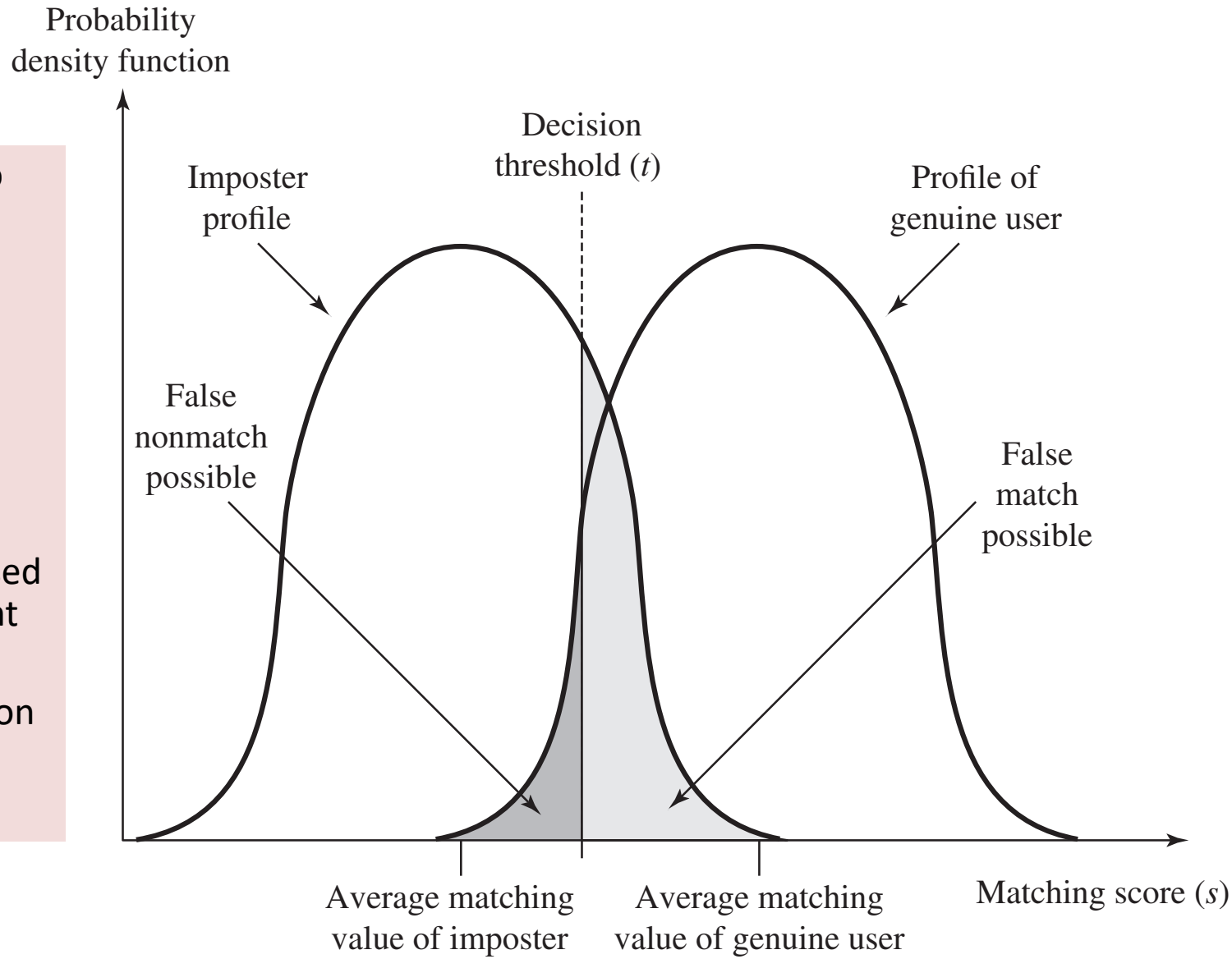


Profiles of a Biometric Characteristic of an Imposter and an Authorized User

The shaded part to the left indicates a range of values for which a **false nonmatch** is possible,

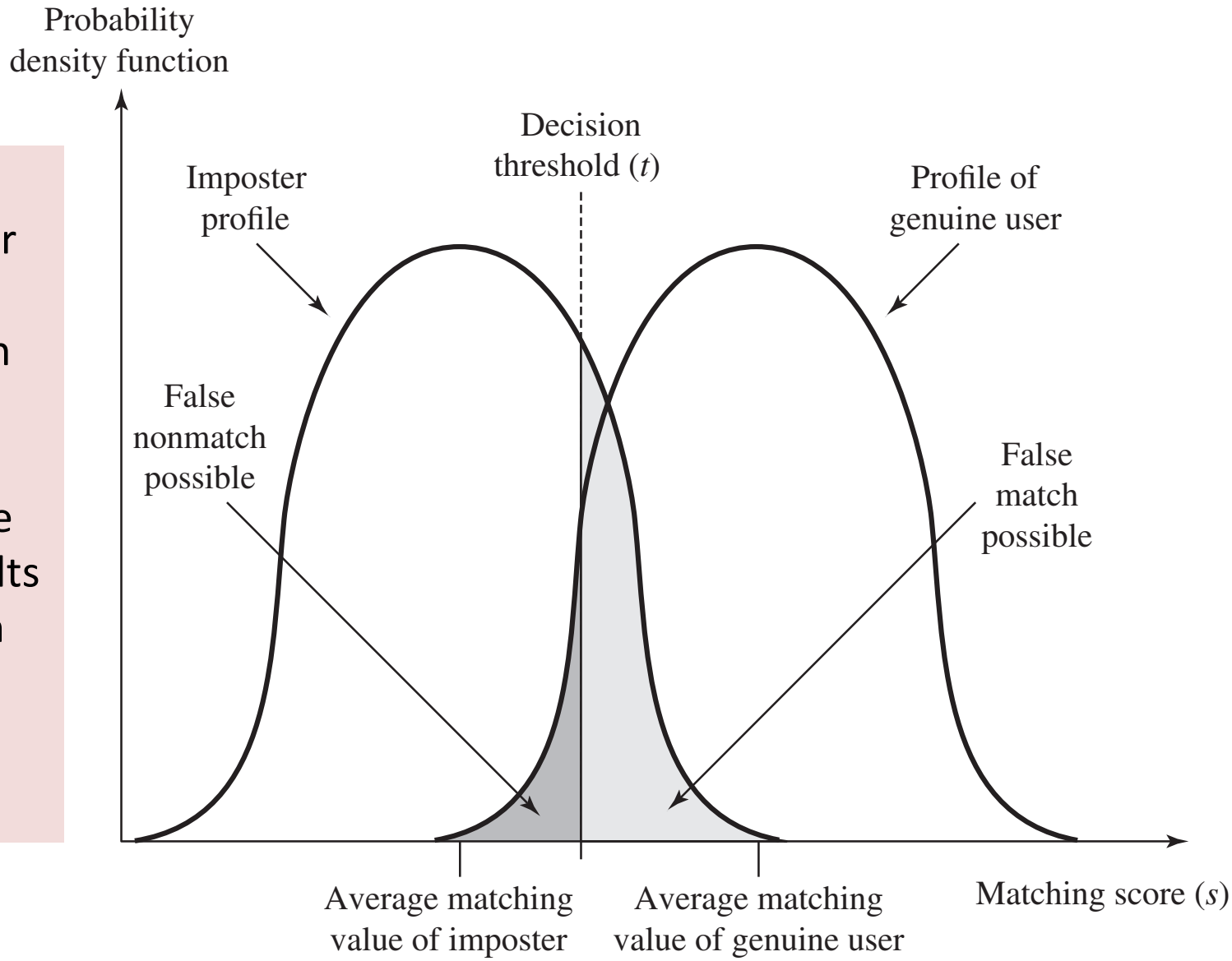
meaning that samples from the same source are erroneously assessed to be from different sources,

causing the rejection of a valid user



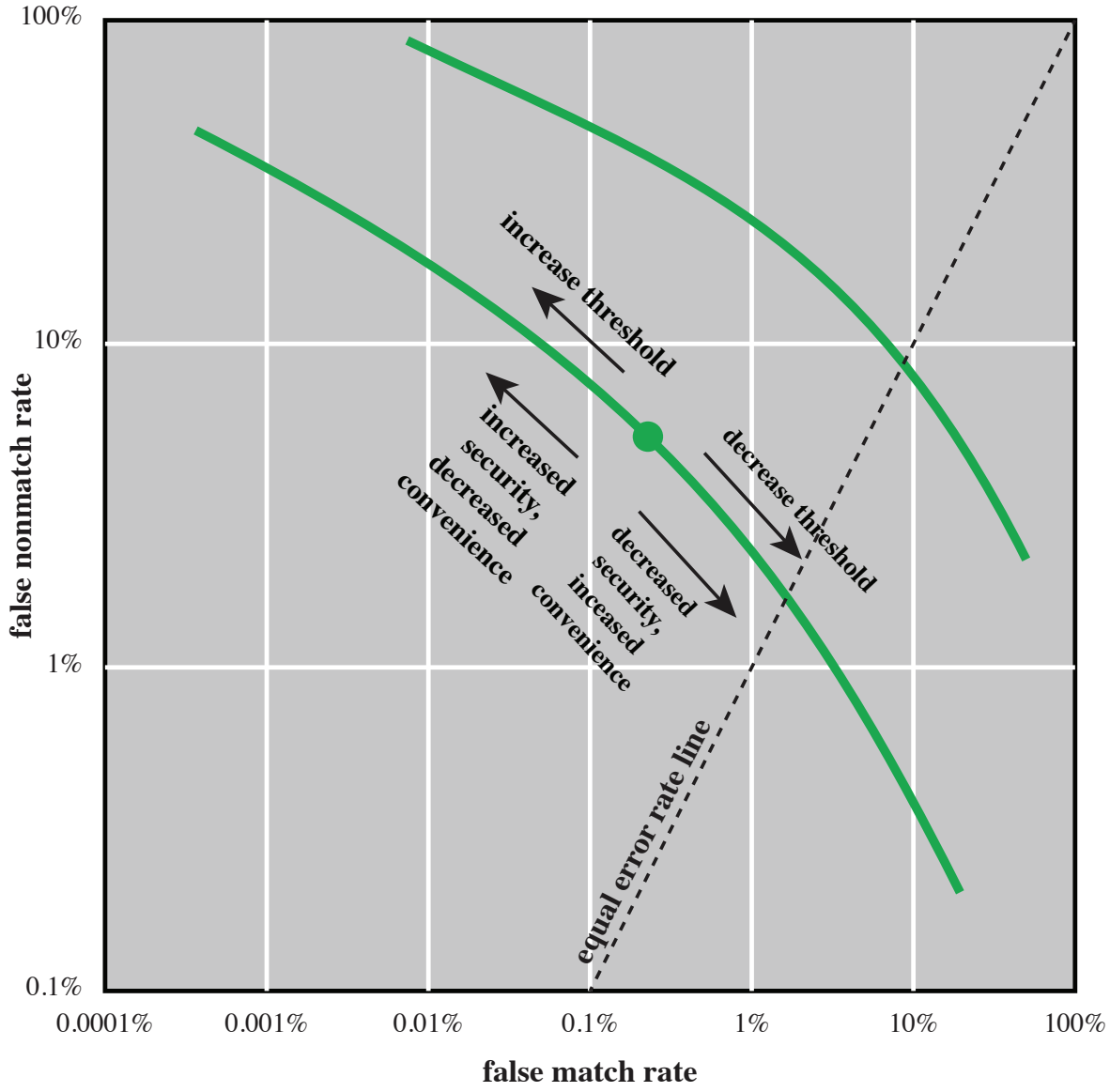
Profiles of a Biometric Characteristic of an Imposter and an Authorized User

By moving the threshold, left or right, the probabilities can be altered, but note that a decrease in false match rate results in an increase in false nonmatch rate, and vice versa

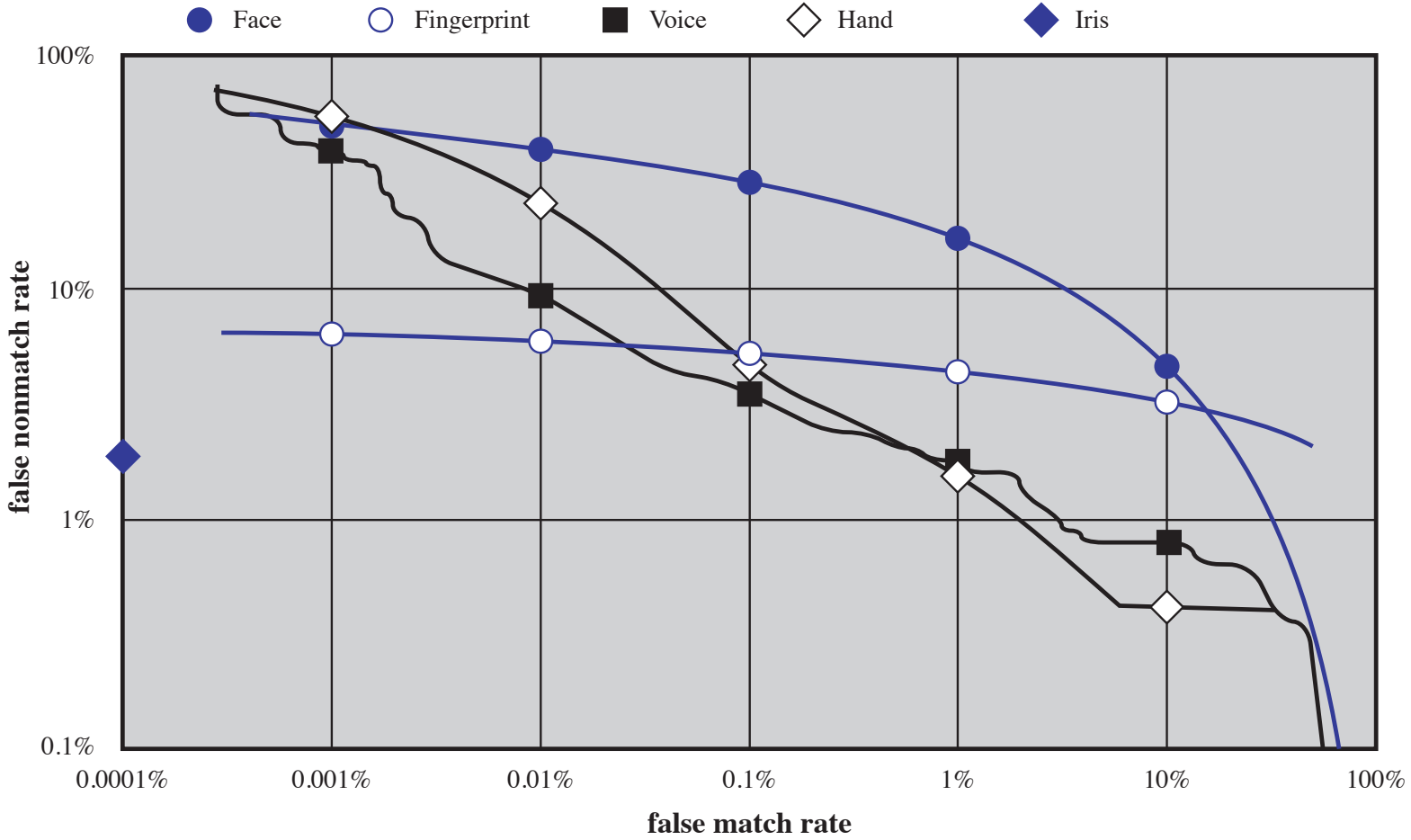


Idealized Biometric Measurement Operating Characteristic Curves

- We can plot the false match versus false nonmatch rate, called the **operating characteristic curve**
- Two idealized curves for two different systems: the curve that is lower performs better
- The **dot** on the curve corresponds to a specific threshold for biometric testing
- Shifting the threshold along the curve up and to the left provides greater security at the cost of decreased convenience
- The inconvenience comes from a valid user being denied access and being required to take further steps



Operating Characteristic Curves from the testing of an actual product



- 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

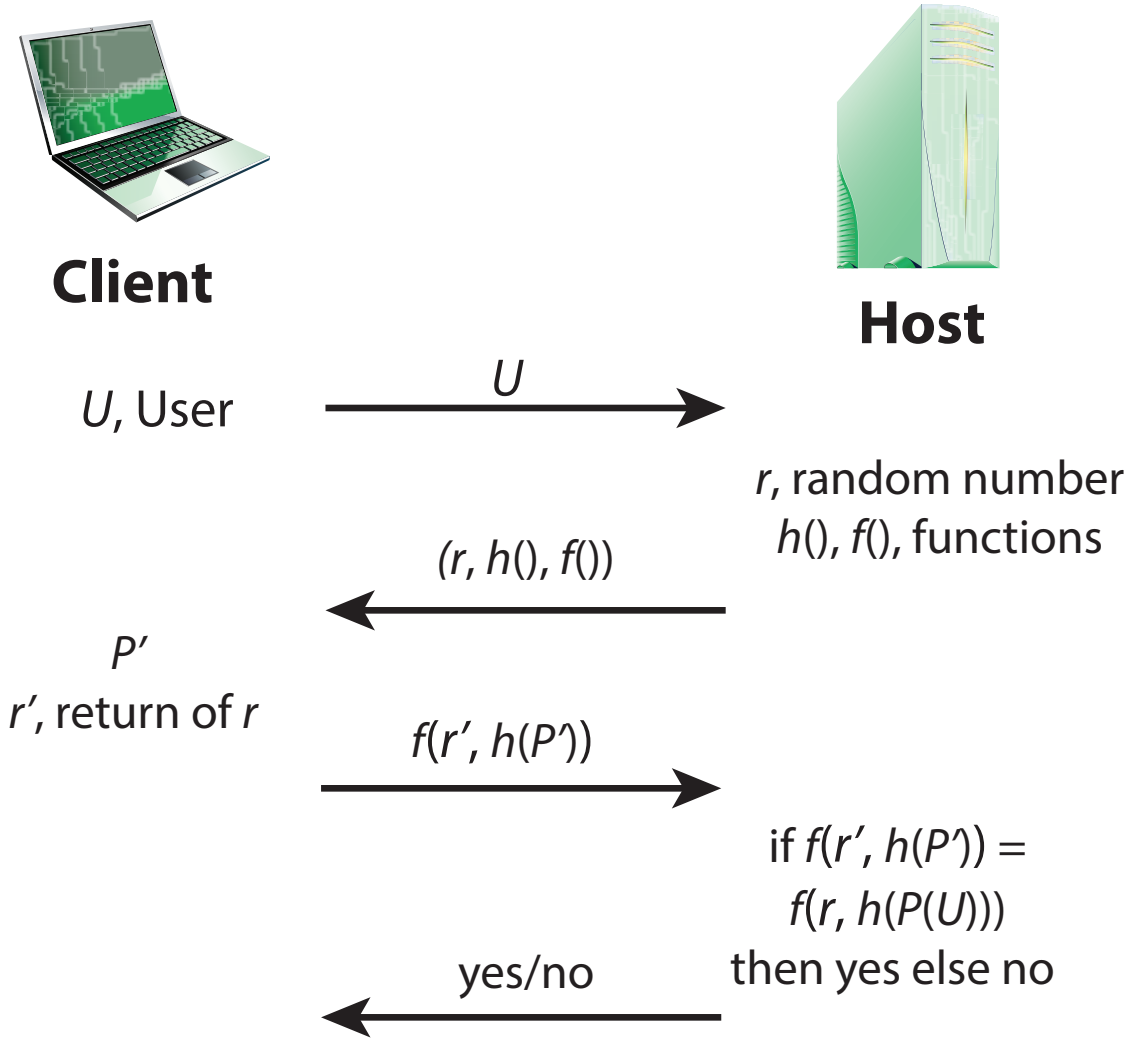
Index

Remote User Authentication

- 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, is the *simplest* form of user authentication
- Authentication over a network, the Internet, or a communication link is *more complex*
  - Additional security threats can be e.g.
    - an eavesdropper being able to capture a password
    - an adversary replaying an authentication sequence that has been observed
- Generally relies on some form of a challenge-response protocol to counter threats

Basic Challenge-Response Password Protocol

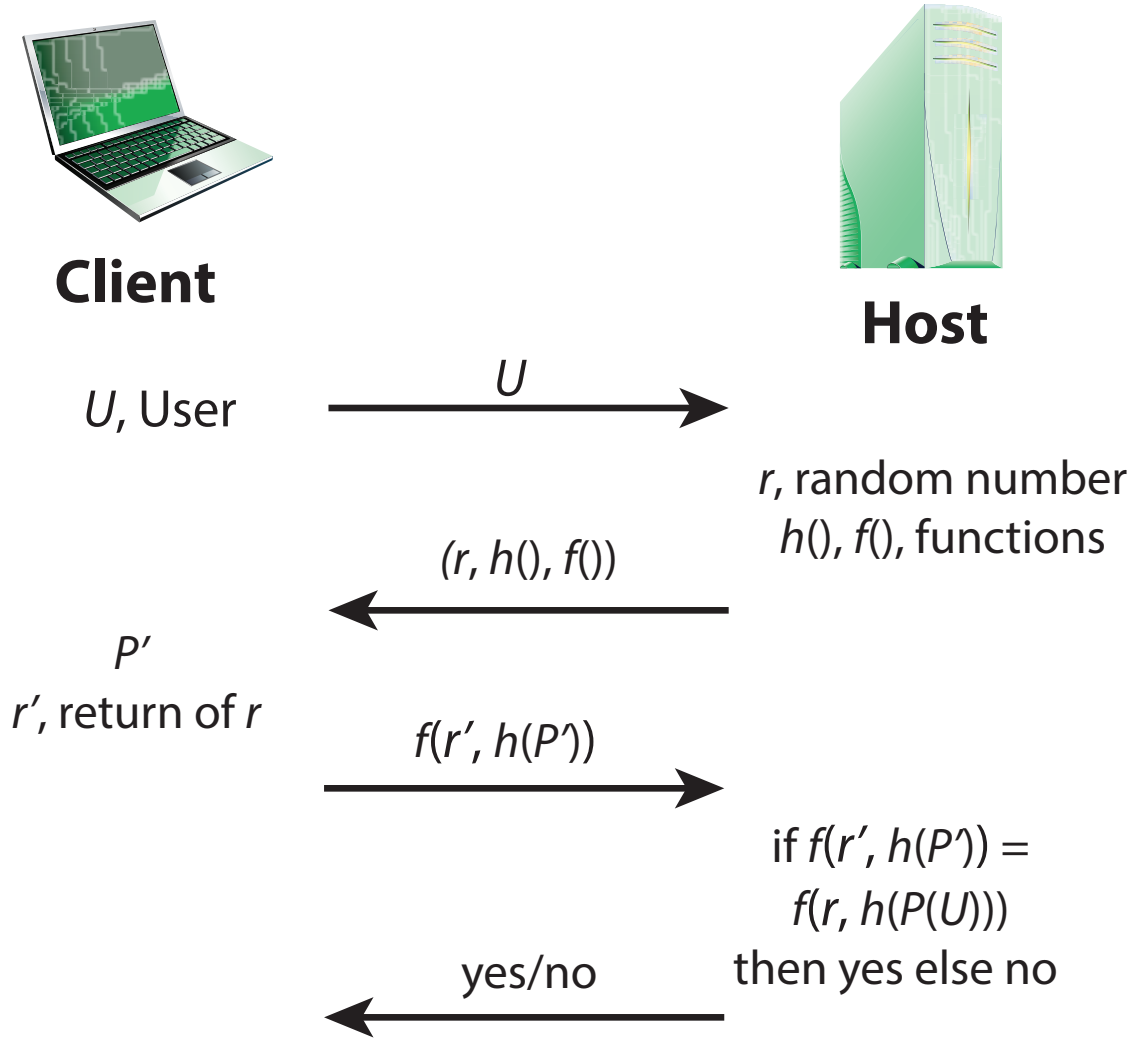
- A user first transmits her identity to the remote host
- The host generates a random number (**nonce**)
- The **challenge** also includes two functions (h() is a HF)
- r' is r, P' is the user's **password,** thus the **response** consists of the hash code of the user's password combined with the random number using the function f
- The host stores the hash code of each registered 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**



Basic Challenge-Response Password Protocol

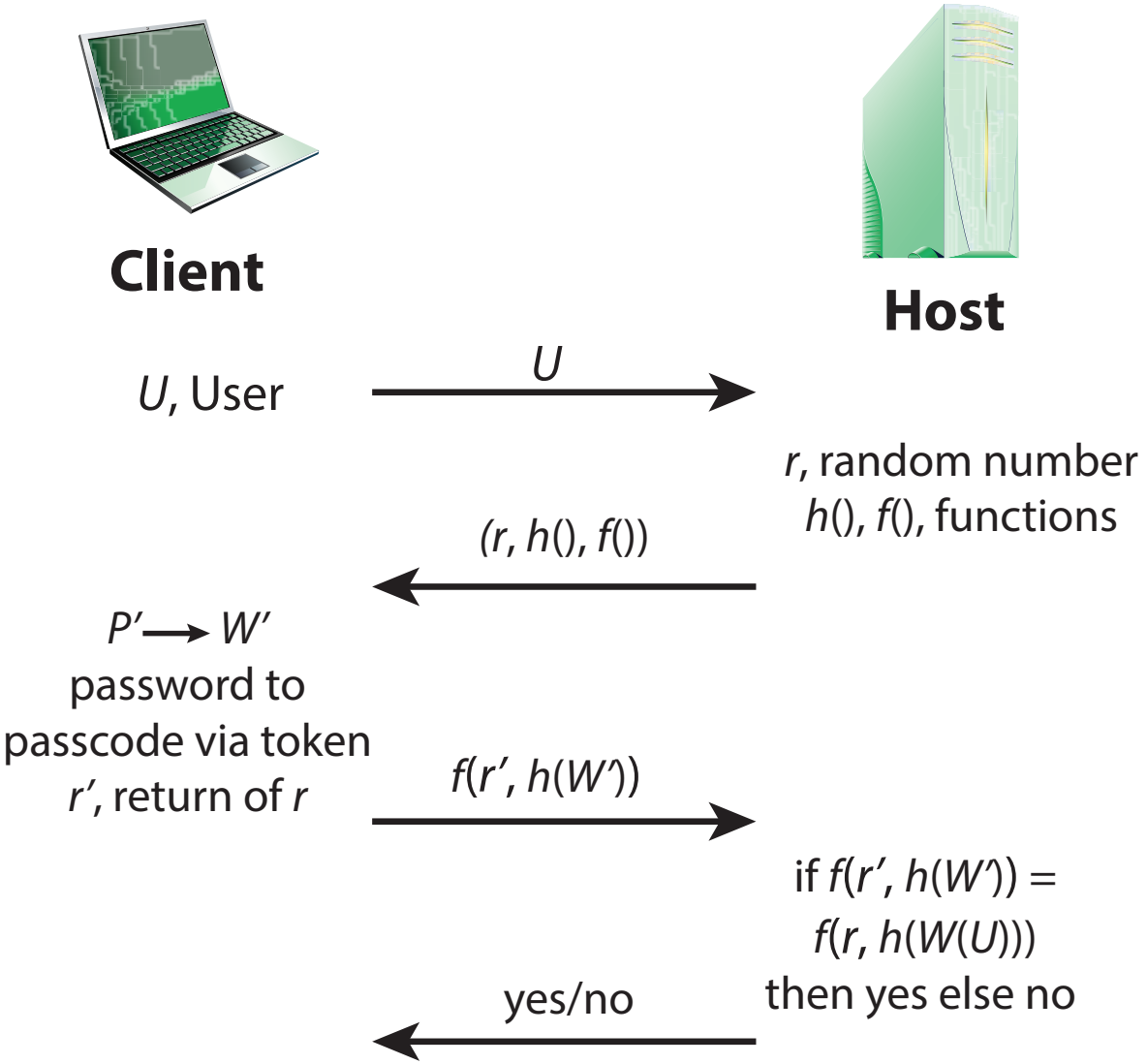
This scheme defends against several forms of attack

- The host stores the hash of each registered user's password, this secures the password from **intruders** into the host system and from **untrustworthy** hosts or unethical administrators
- The hash of the password is not transmitted directly; thus, for a suitable function f, the password hash **cannot be captured** during transmission
- The use of a nonce as one of the arguments of f defends against **replay attacks**, in which an adversary captures the user's transmission and attempts to log on to a system by retransmitting the user's messages



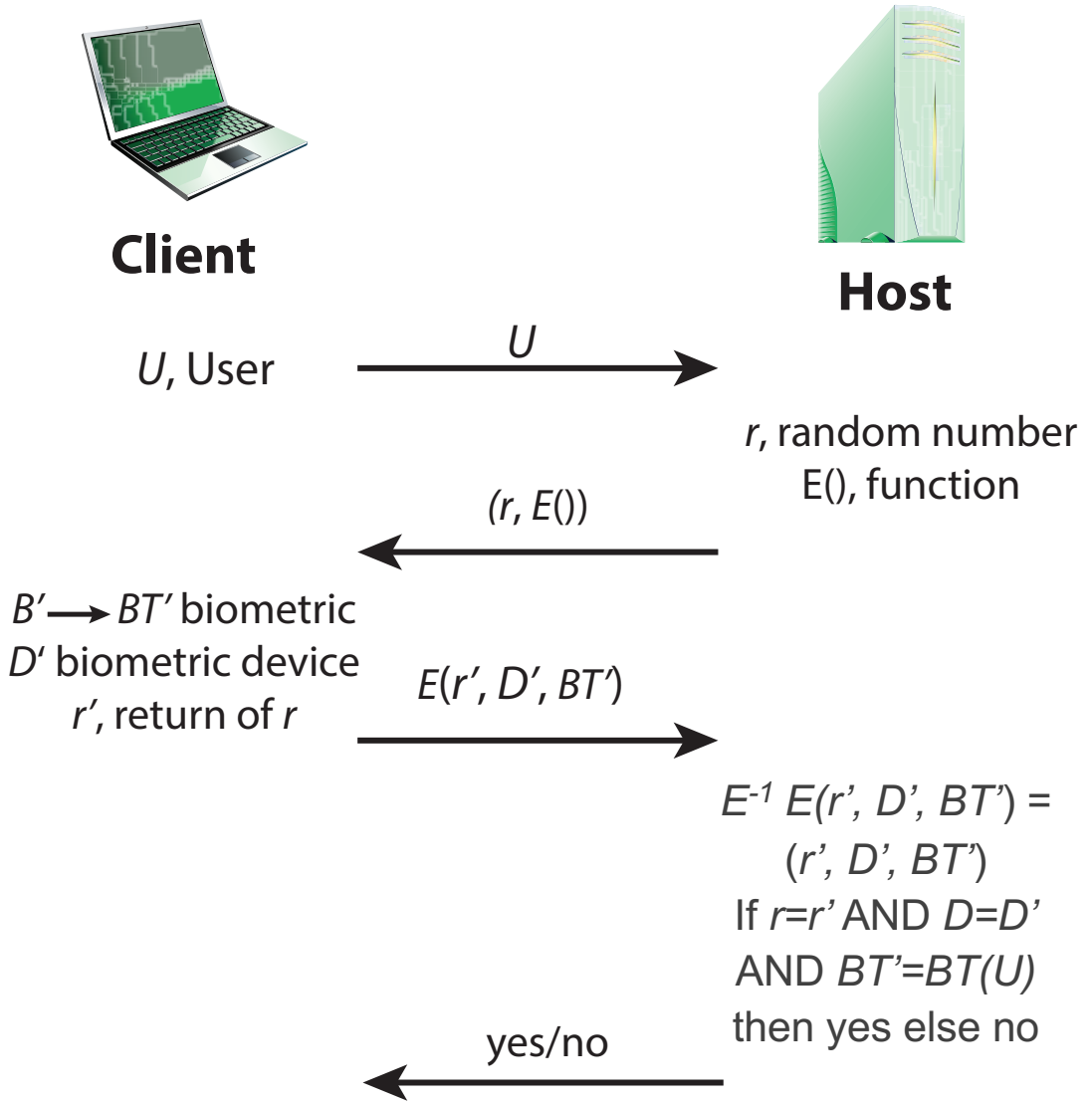
Basic Challenge-Response Token Protocol

- At the user end, the token provides a **passcode** *W'*
- The token either stores a **static** passcode or **dinamically** 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'* shared only between the user and the token
- Instead of a *potentially weak* password, a *long and random* passcode is used
- 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



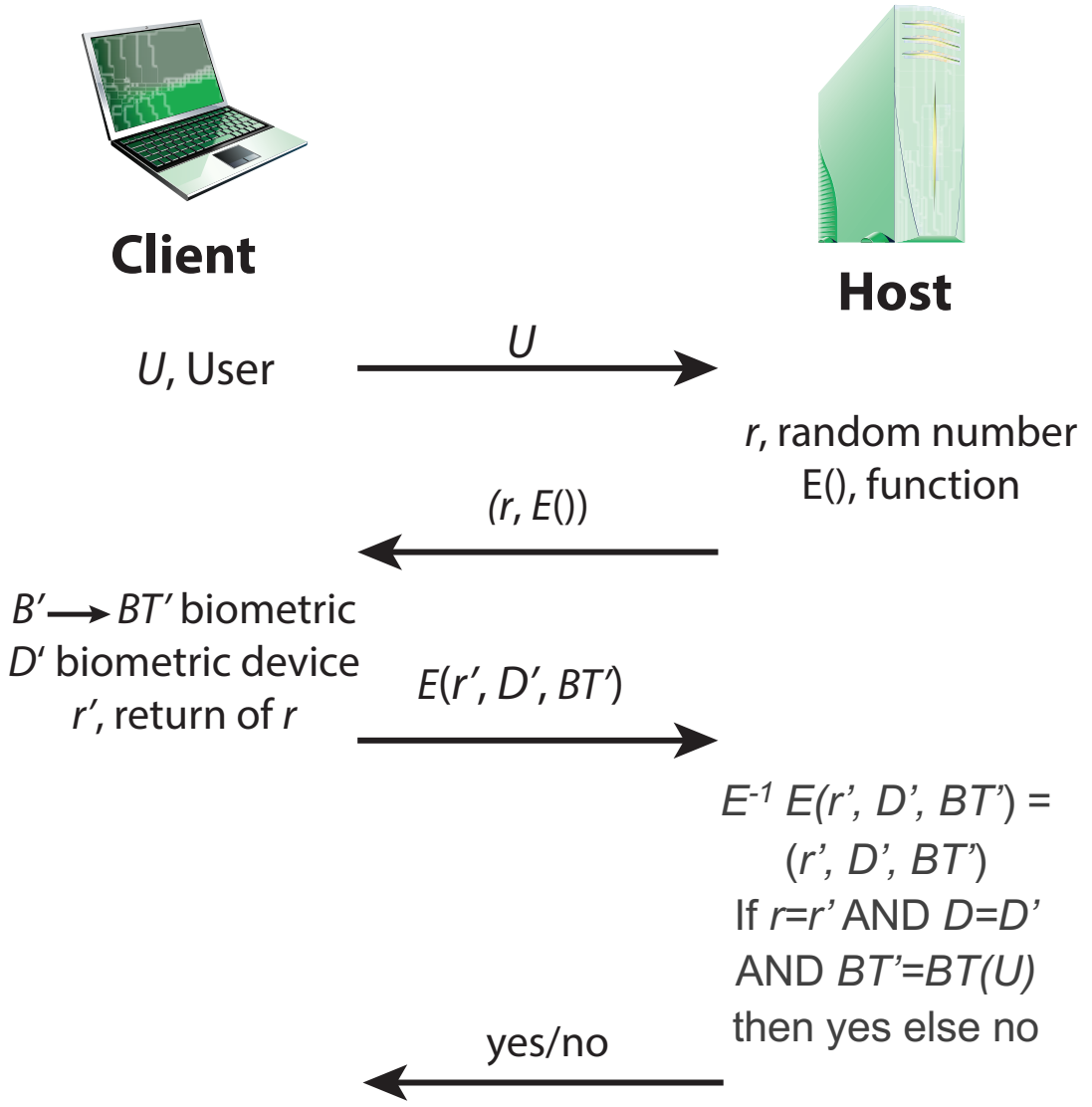
Basic Challenge-Response Static Biometric Protocol

- The host responds with a nonce and the identifier for an **encryption** function E()
- On the user side, a client system controls a biometric device and generates a biometric template BT' from the user's biometric B'
- In the **ciphertext** E(r', D', BT'), D' identifies this particular biometric capture device
- The host decrypts the incoming message to recover the three transmitted parameters and compares these to **locally** stored values



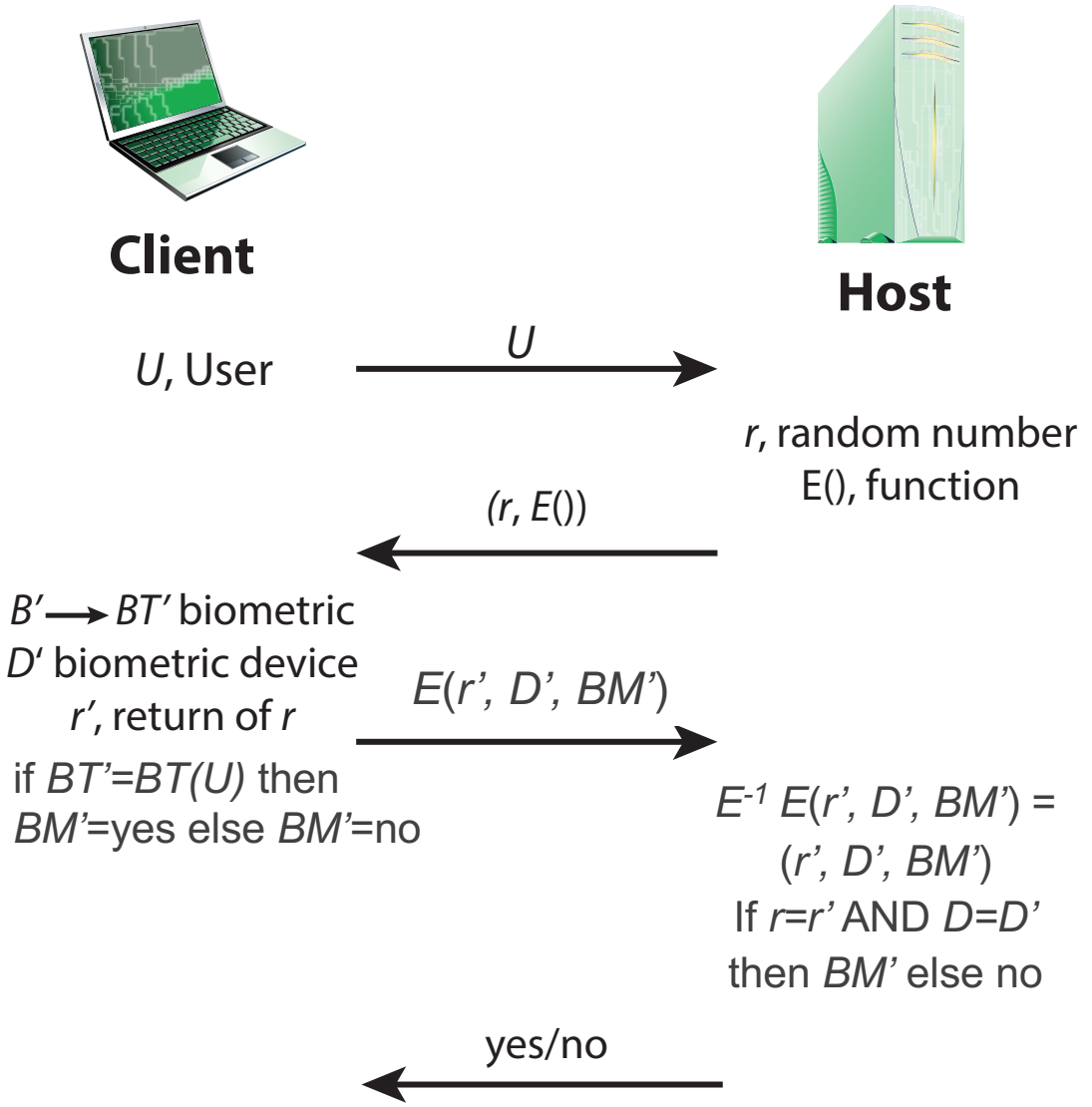
Basic Challenge-Response Static Biometric Protocol

- 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
- Biometrics at the host are protected (for privacy reasons rather than confidentiality) by storing them as an encrypted template (because biometrics are matched not exactly but by 'closeness', and hashed codes do not maintain the property of closeness)
- The **matching score** between BT' and the stored template must exceed a predefined threshold



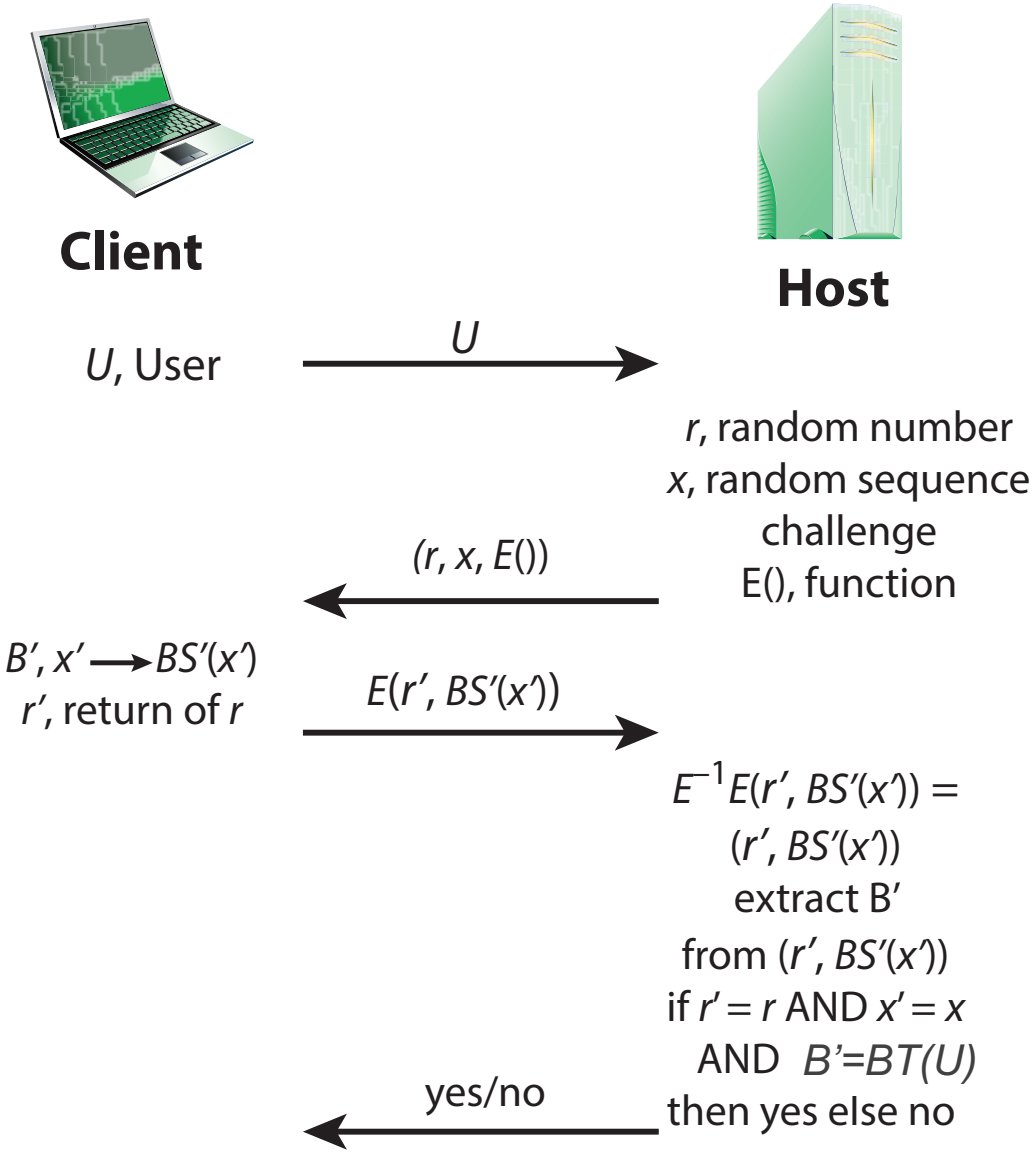
Basic Challenge-Response Static Biometric Protocol

- There is a **variant** of the protocol where the match is done at the client
- The distinction is that a biometric is captured, processed to a template BT', and matched to yield a yes/no match result, BM', all at the client
- The information is transmitted to the host, which determines authentication depending on a correct match and, which is crucial, the **authentication** of the biometric capture device
- The host contains **no biometric** information; instead, the biometric template is stored at the client



Basic Challenge-Response Dynamic Biometric Protocol

- The host provides a **random sequence** as well as a nonce as a challenge
- 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 host generates a comparison based on the incoming biometric signal *BS′(x′)* and the stored **template** *BT(U)* for this user
- If the comparison value exceeds a predefined threshold, the user is **authenticated**

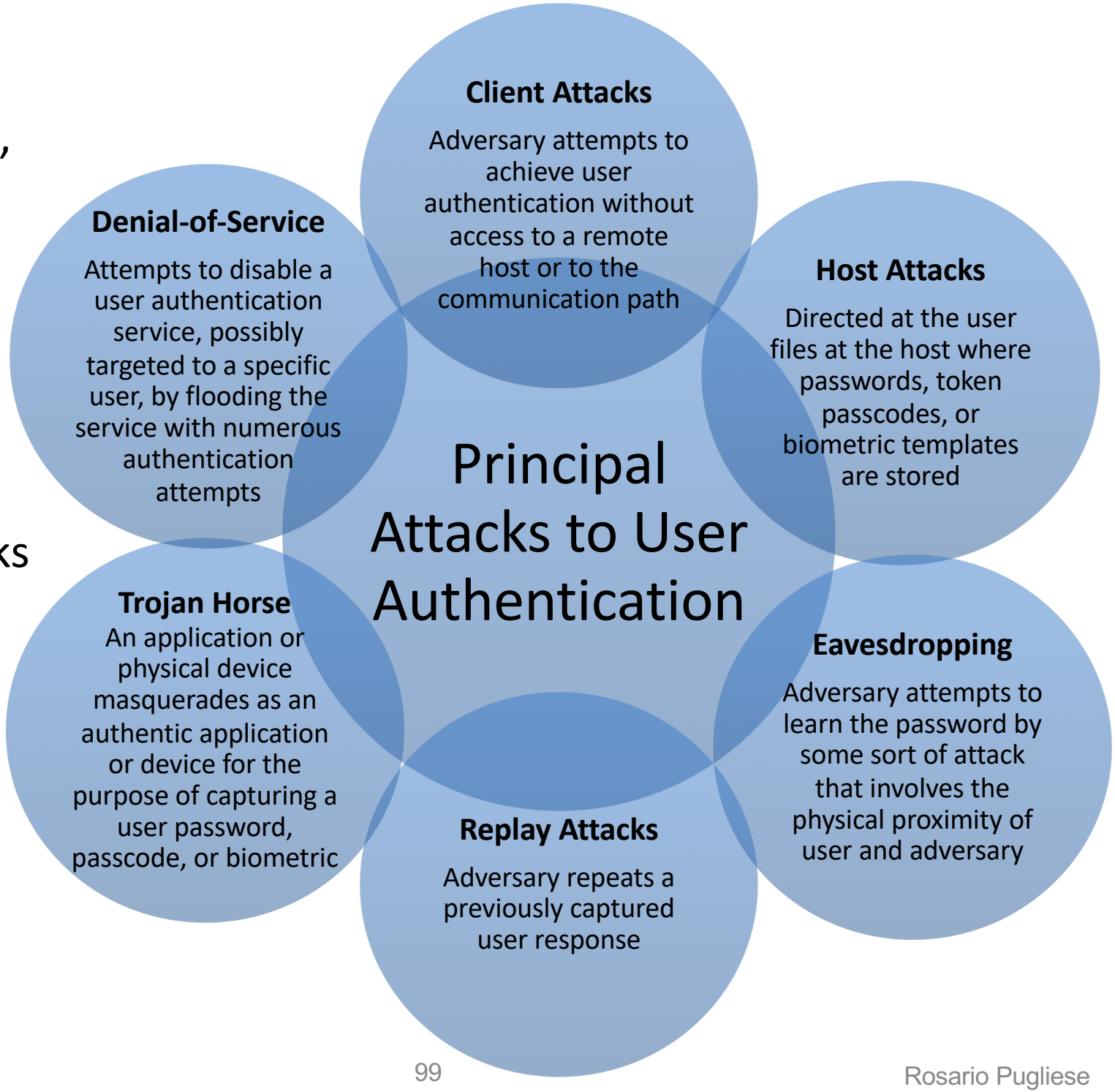


Basic Challenge-Response Dynamic Biometric Protocol

- One **difference** from the static biometric protocol is that the actual biometric is involved in challenge-response
- Another **difference** is that the capture device need not be machine authenticated, since the challenge-response protocol defends against replay and forgery, and matching is performed at the host
- There is also a **variant** where the match is done at the client
  - The result is sent to the host along with a device identifier to verify that it is registered and unmodified
  - As compared with host matching, this protocol saves transmission bandwidth and template storage space at the host, at the cost of a more powerful and trustworthy device at the client

Index

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



Attacks (with examples), Authenticators, and Typical Defenses

Attacks	Authenticators	Examples	Typical defenses
Client attack	Password	Guessing, exhaustive search	Large entropy (use lengthy and unpredictable passwords); limit guessing attempts
	Token	Exhaustive search, but require the physical token	Large entropy (use the token to generate a high-entropy passcode from a low-entropy PIN or password); limit guessing attempts
	Biometric	False match	Large entropy; limit guessing attempts
Host attack	Password	Dictionary/exhaustive search	Hashing; large entropy; protection of password database (through access control)
	Token	Passcode theft	Same as password; 1-time passcode (i.e. no host passcode file)
	Biometric	Template theft	Authentication of the capture device (static biometric); challenge response (dynamic biometric)
Eavesdropping, theft, and copying	Password	Shoulder surfing; finding a written copy of the password	User diligence to keep secret; administrator diligence to quickly revoke compromised passwords; multifactor authentication
	Token	Theft, counterfeit hardware	Multifactor authentication; tamper resistant/evident token
	Biometric	Copying (spoofing) biometric	Copy detection at capture device and capture device authentication
Replay attack	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 forged 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 authentication with token

Index

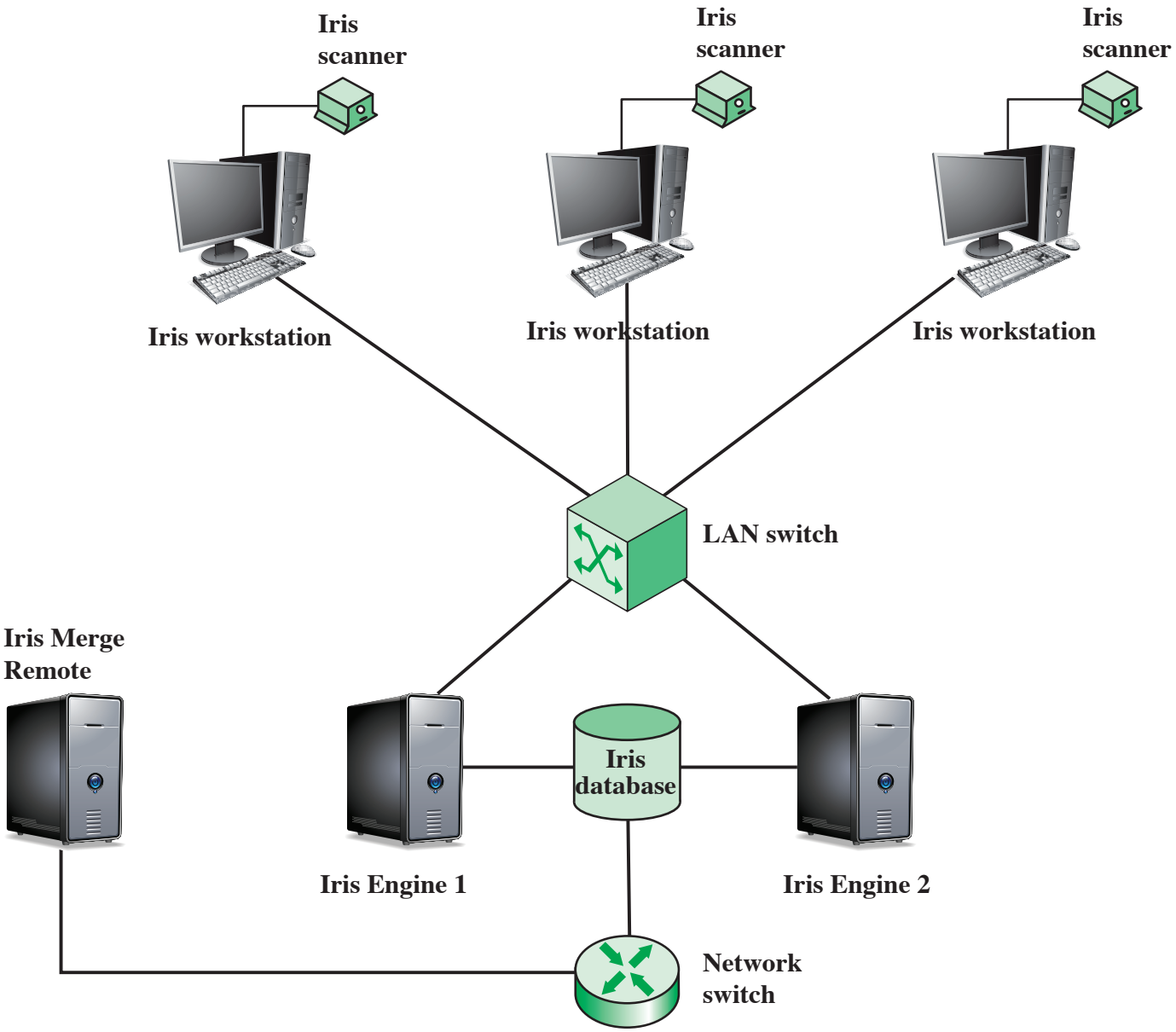
An iris biometric identification system: concerns and requirements

- Developed by the United Arab Emirates (UAE) at border control points
- Concern: to counter attempts by expelled persons to re-enter the country
  - Traditional means of preventing reentry involve identifying individuals by name, date of birth, and other text-based data
  - The risk is that this information can be changed after expulsion
- The UAE identified the following requirements on the biometric identification system
  - Identify a single person from a large population of people
  - Rely on a biometric feature that does not change over time
  - Use biometric features that can be acquired quickly
  - Be easy to use
  - Respond in real-time for mass transit applications
  - Be safe and non-invasive
  - Scale into the billions of comparisons and maintain top performance
  - Be affordable
- Iris recognition has been chosen as the **most efficient** and **foolproof method**
  - No two irises are alike
  - There is no correlation between the iris patterns of even identical twins, or the right and left eye of an individual

An iris biometric identification system: implementation

- System *implementation* involves enrollment and identity checking
- Enrollment**:** all expelled foreigners are subjected to an iris scan at one of the multiple enrollment centers
  - This information is merged into one central database
- Identification: iris scanners are installed at all 17 air, land, and sea ports into the UAE
  - Over a distributed network the iris codes of **all** arriving passengers are compared in real-time exhaustively against the central database of enrollments
  - This is computationally a *more demanding* task than verifying an identity
  - The daily number of iris cross-comparisons is *well over 9 billions*

An iris biometric identification system: architecture



An iris biometric identification system: adversaries

- As with any security system, adversaries are always looking for countermeasures
- **Expatriates** who were banned from the UAE started using eye drops in an effort to fool the government's iris recognition system when they try to re-enter the country
- UAE **officials** had to adopt new security methods to detect if an iris has been dilated with eye drops before scanning
- A new algorithm and computerized step-by-step procedure has been adopted to help officials determine if an iris is in normal condition or an eye-dilating drop has been used

Index

- Digital User Authentication Principles
- Password-Based Authentication
- Token-Based Authentication
- Biometric Authentication
- Remote User Authentication
- Security Issues for User Authentication
- Practical Application: An Iris Biometric System
- Case Study: Security Problems for ATM Systems

A vulnerability of some ATM Systems

- This case study is based on an auditing report released by Redspin (http://www.redspin.com), an IT security company, describing a security vulnerability in ATM (Automated Teller Machine) usage that affects a number of small to mid-size ATM card issuers
- This vulnerability illustrates that *cryptographic functions and services alone do not guarantee security*
  - They must be properly implemented as part of a system

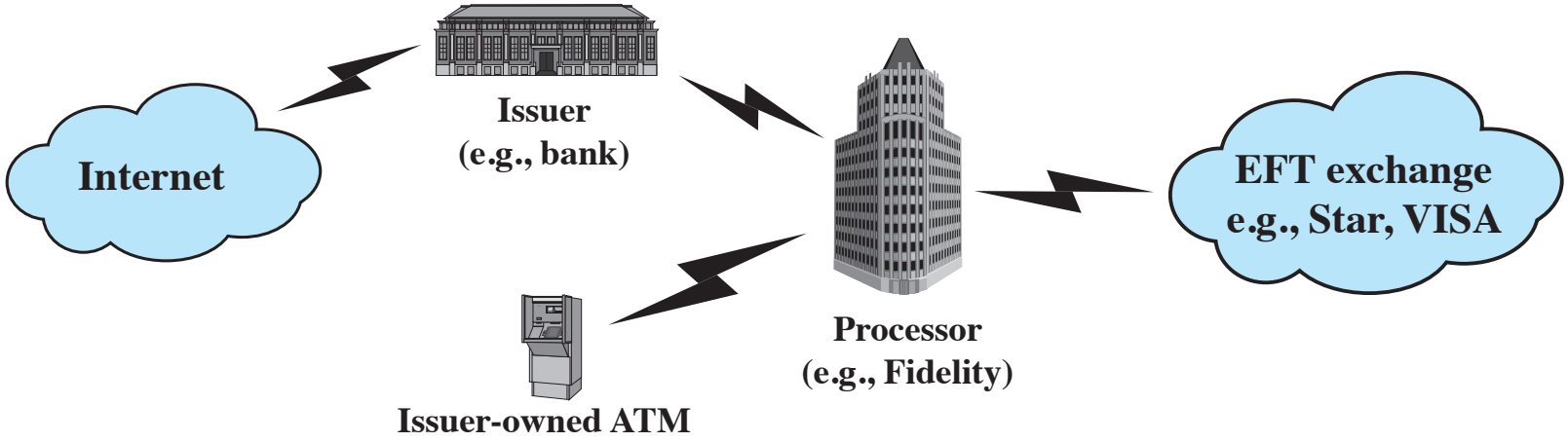
ATM Systems: involved actors

- Cardholder: an individual to whom a debit card is issued; 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 (e.g. a bank) is responsible for the cardholder's account and authorizes all transactions
- Processor: an organization that provides services, such as core data processing (PIN recognition and account updating) and electronic funds transfer (EFT), to issuers
  - EFT allows an issuer to access regional and national networks that connect point of sale (POS) devices and ATMs worldwide

ATM Systems: expected services

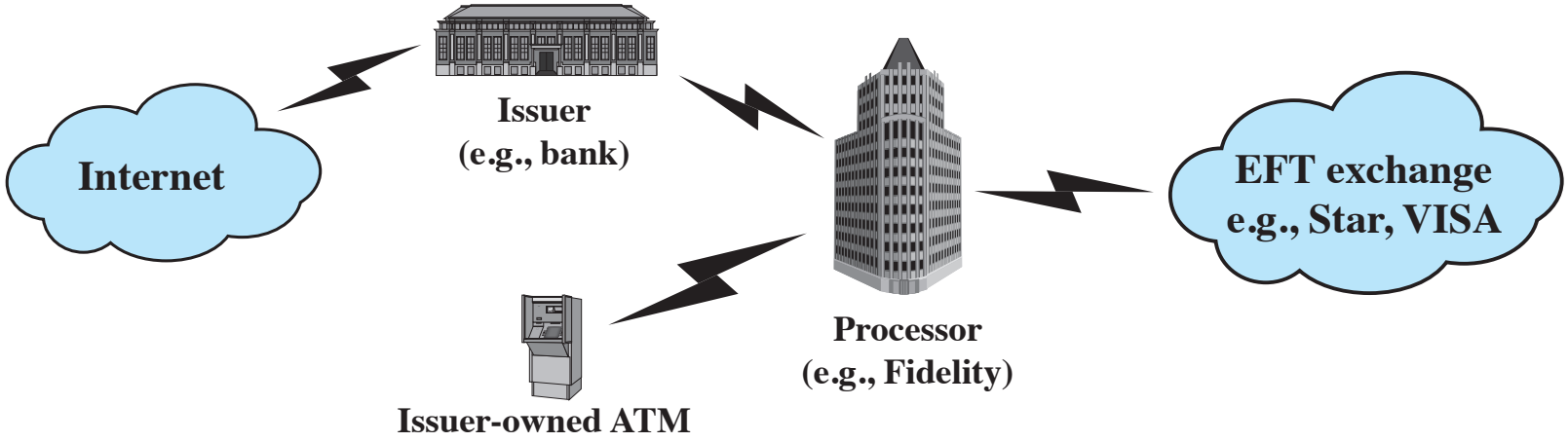
- Customers expect 24/7 service at ATM stations
- For many small to mid-sized issuers, it is more cost-effective for contract processors to provide the required core data processing and EFT services
- Each service typically requires a dedicated data connection between the issuer and the processor, using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

Point-to-point connection to processor



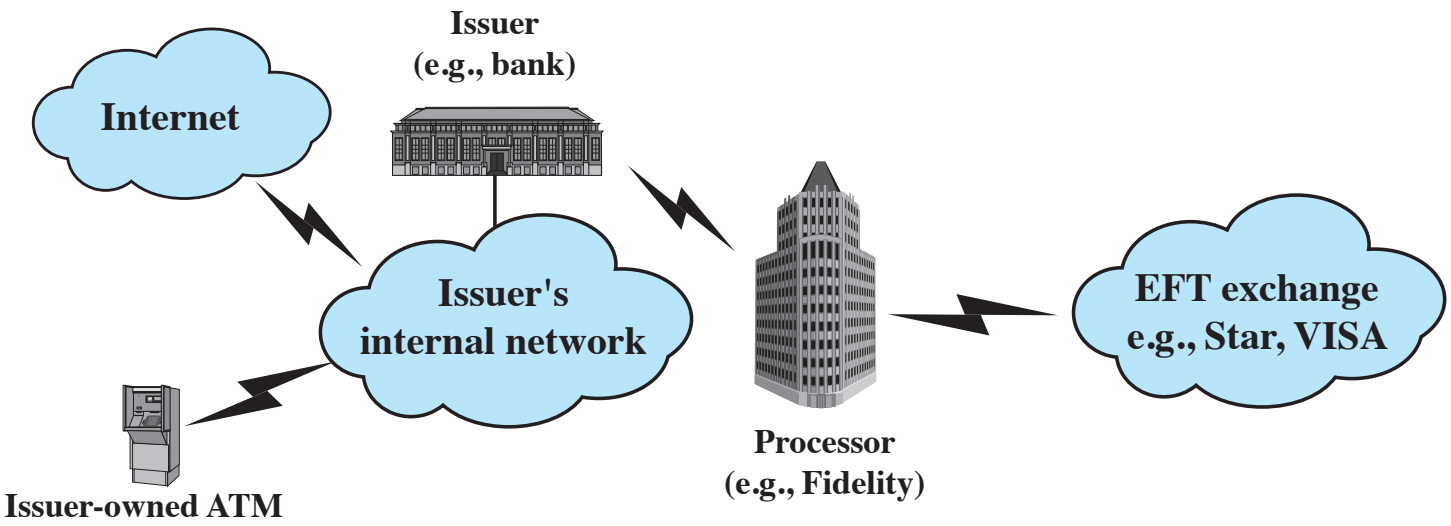
- Prior to about 2003, the ATM units linked directly to the processor rather than to the issuer that owned the ATM
  - The use of a dedicated link made it difficult to maliciously intercept transferred data
- To add to the security, the PIN portion of messages transmitted from ATM to processor was encrypted using DES
- Processors have connections to EFT exchange networks to allow cardholders access to accounts from any ATM
- 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

Point-to-point connection to processor



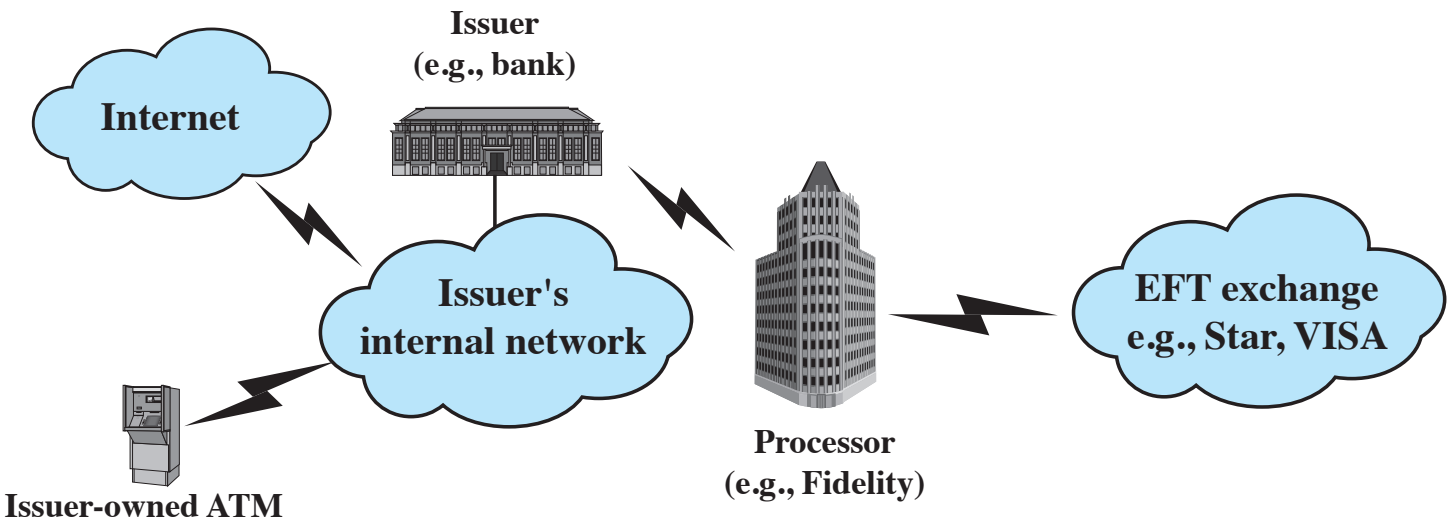
- In the early 2000s, banks worldwide began the process of migrating from an older generation of ATMs using IBM's OS/2 operating system to new systems running Windows
- The mass migration has been spurred by a number of factors, as
  - IBM's decision to stop supporting OS/2 by 2006
  - market pressure from creditors such as MasterCard International and Visa International to introduce stronger cryptography (Triple DES)
  - pressure from U.S. regulators to introduce new features for disabled users

Shared connection to processor



- Many banks included a number of other enhancements at the same time
  - Since 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
  - 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

Shared connection to processor



- 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
  - But for the PIN, all 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

Main security vulnerabilities

The new architecture leads to two main 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
  - Acting *as the ATM*, the adversary can modify account balances or perform money transfers until the ATM encryption key is changed

Redspin recommended Countermeasures

- *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
  - implementing network-level encryption between routers that the ATM traffic traverses
- *Long-term fixes* involve changes in the application-level software
  - Ensuring confidentiality requires encrypting all customer-related information that traverses the network, not only the PIN
  - 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

- Digital User Authentication Principles
  - A Model for Electronic 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 Identify Cards
- 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
- Practical Application: An Iris Biometric System
- Case Study: Security Problems for ATM Systems