Information Systems Security

Lecture 6

User Authentication and
Cryptographic Key Infrastructure

Dr. En. Bader Ahmad

Outline

1. Entity Authentication
2. Entity Authentication Functions
   - 2.1. Something you have
   - 2.2. Something you are
   - 2.3. Something you know
     - 2.3.1. Passwords
     - 2.3.2. OTP
     - 2.3.3. Challenge-Response
3. Cryptographic Key Infrastructure
4. Conclusion

1. Entity Authentication

- The aim of this lecture is to present some techniques that allows one party (the *verifier*) to gain assurances that the identity of another (the *claimant*) is as declared, thereby preventing impersonation.
- These techniques are referred to as ***identification***, ***entity authentication***, and ***identity verification***.
- Entity authentication is the process whereby one party is assured of the identity of a second party involved in a protocol, and that the second has actually participated.

Entity authentication: Uses

1. Access control
   - An entity, often human user, must provide assurance of their identity in real time in order to have access to either physical or virtual resources.
2. As part of a more complex cryptographic process:
   - Typically established at the start of a connection: an entity must provide assurance of their identity in real time in order for the extended process to complete satisfactorily.
     - For example, the process of establishing a symmetric key that two users can use to immediately communicate with one another commonly involves mutual entity authentication in order to provide the two users with sufficient assurance that they have agreed a key with the "correct" person.

Entity authentication: Types

- Types of entity authentication:
  - ***Unilateral entity authentication*** is assurance of the identity of one entity to another (and not vice-versa).
    - Examples:
      - Online shopping
      - File downloading
  - ***Mutual entity authentication*** occurs if both communicating entities provide each other with assurance of their identity.
    - Examples:
      - Online Banking
      - E-learning

2. Entity Authentication Functions

- The most common ways of providing entity authentication are by using (a combination of) the following:
  - Something that you have
  - Something that you are
  - Something that you know

2.1. Something you have

- **Dumb tokens:**
  - Any physical device without a memory that can be used as a type of electronic key.
  - Dumb tokens typically operate with a reader that extracts some information from the token and then indicates whether the information authenticates the entity or not.
  - A good example of a dumb token is a plastic card with a magnetic stripe. The security of the card is based entirely on the difficulty of extracting the information from the magnetic stripe.
  - It is common to combine the use of a dumb token with another entity authentication technique, such as one based on something you know.

Something you have

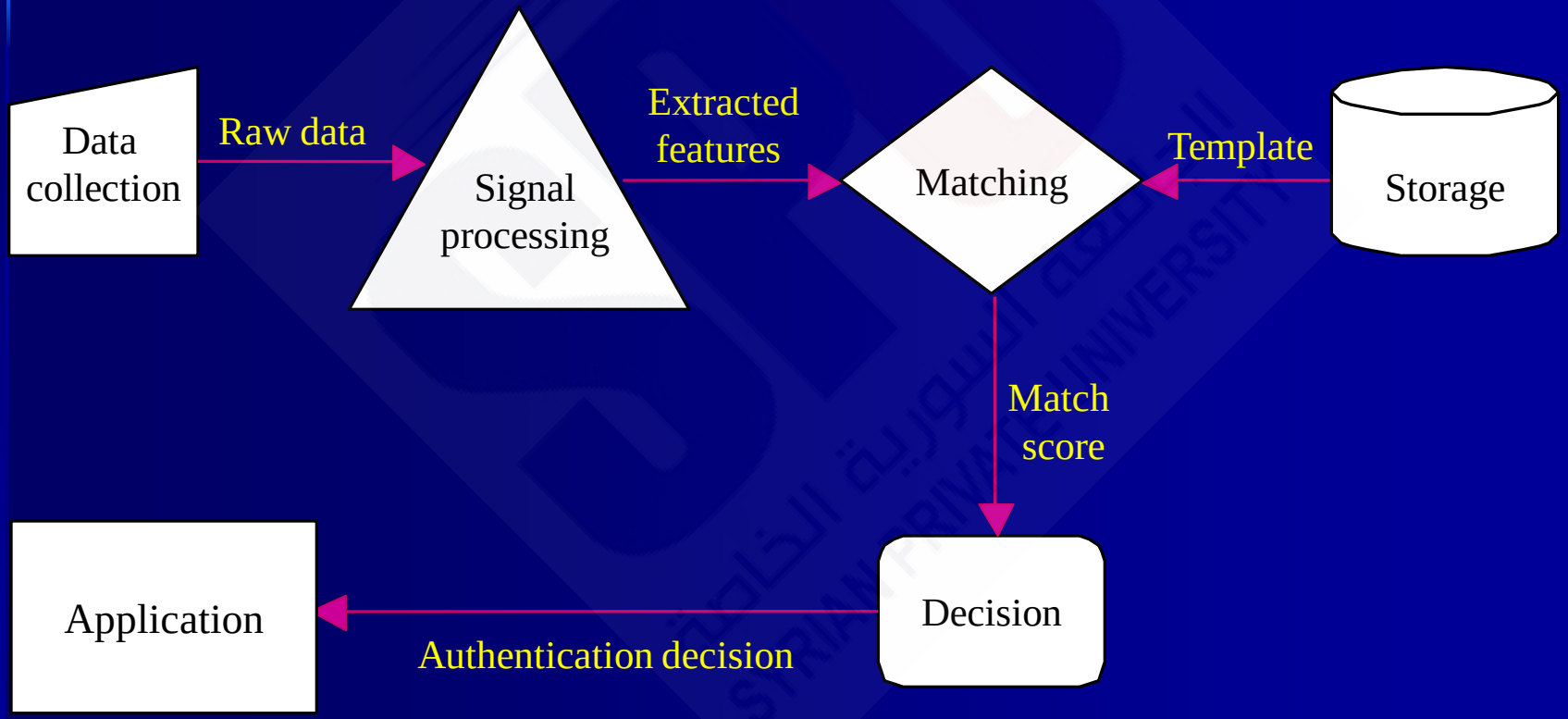
- Smart cards:
  - A plastic card that contains a **chip**, which gives the card a limited amount of memory and processing power.
  - A smart card can **store** secret data more securely, and can also engage in **cryptographic processes** that require some computations to be performed .
  - Smart cards have limited memory and processing power, thus restricting the types of operation that they can comfortably perform.
  - Smart cards are widely used in most countries for banking operations, electronic marketing applications, etc.

2.2. Something you are

- **Biometrics**
  - Techniques for human user authentication that are based on physical characteristics of the human body.
  - A biometric control typically converts a physical characteristic into a digital template that is stored on a database.
    - When the user physically presents themselves for entity authentication, the physical characteristic is measured by a reader, digitally encoded, and then compared with the template.

Something you are

- Biometric system model



Something you are

- **Biometrics**
  - **Static** (unchanging) measurements include fingerprints, hand geometry, face recognition, retina scan.
  - **Dynamic** (changing) measurements include handwriting measurements and voice recognition.
  - There are many implementation issues and as yet entity authentication is not widely provided using these techniques.

Something you are

Fingerprints

Optical fingerprint sensor
[Fingerprint Identification Unit
FIU-001/500 by Sony]

2.3. Something you know

- Passwords:
  - **Passwords** are probably the most popular technique for providing entity authentication, despite concerns about how secure they actually are.
- A password may be a sequence of characters
  - Examples: 10 digits, a string of letters, *etc*.
  - Generated randomly, by user, by computer with user input
- A password may be a sequence of words
  - Examples: pass-phrases
  - A *pass-phrase* is a sequence of characters that is too long to be a password and it is thus turned into a shorter virtual password by the password system
- Algorithms
  - Examples: one-time passwords, challenge-response.

2.3.1. Passwords

- Passwords stored in plaintext files
  - If password file compromised, all passwords are revealed
  - Usually password files are read- and write- protected
- Passwords stored in encrypted file
  - Encrypted/hashed versions of passwords are stored in a password file
- Examples:
  - Windows password and Unix password (next slides)

Windows Passwords

- Older versions:
  - The user's password is converted to uppercase.
  - This password is either null-padded or truncated to 14 bytes.
  - The "fixed-length" password is split into two 7-byte halves.
  - These values are used to create two DES keys
  - Each of these keys is used to DES-encrypt the constant ASCII string "KGS!@#$%", resulting in two 8-byte ciphertext values.
  - These two ciphertext values are concatenated to form a 16-byte value, which is the **LM hash**.
- Network logging: Hashes are sent over the net as cleartext
- What LM hashes-related security weaknesses can be identified?

Windows Passwords

- Newer versions:
  - NTLM1 hashes
    - Windows NT SP3 or later
    - Based on DES and MD4 (hash function)
    - Network authentication: ***Challenge-response*** mechanism
  - NTLM2 hashes:
    - Windows 2000 domains and Windows NT4 SP4 or later
    - Based on HMAC-MD5
    - Network authentication: ***Challenge-response*** mechanism
  - Kerberos
    - Windows 2000 and 2003 or later
      - Windows Vista is not backward compatible.

Unix Password

- Example: Original Unix
  - A password is up to eight characters
  - The password is truncated to its first 8 ASCII characters, forming the Unix DES key
  - The key is used to encrypt the 64-bit constant 0.
  - The Unix DES is a variation of the standard DES.
    - A 12-bit *salt* is used to modify the expansion function in DES,
    - Then DES is iterated 25 times
  - Thus the UNIX password is referred to as ***salted password***
  - Unix passwords are stored in file */etc/passwd*

Attack on passwords

- Replay of passwords
  - Specially when passwords are transmitted in **cleartext**
- Exhaustive password search
  - Trying all possible passwords
- Dictionary attack
  - Most users select passwords from a small subset of the password space (e.g., short passwords, dictionary words, proper names)
  - **Dictionary attack**: the attacker tries all possible words, found in an available or on-line list

Password selection

- Problem: people pick easy to guess passwords
  - Based on account names, user names, computer names, place names
  - Too short, digits only, letters only
  - License plates, acronyms, social security numbers
  - Personal characteristics (nicknames, job characteristics, *etc*.)
- Good passwords can be constructed in several ways
  - Contain both upper and lower case characters (e.g., a-z, A-Z)
  - Have digits and punctuation characters as well as letters e.g., 0-9, @#$%^&*()_+|~- =\`{}[]:";'<>?,./)
  - Are at least fifteen alphanumeric characters long and is a passphrase (Ohmy1stubbedmyt0e).

Password selection

- Are not a word in any language, slang, dialect, jargon, etc.
- Are not based on personal information, names of family, etc.
- Passwords should never be written down or stored on-line.
  - Try to create passwords that can be easily remembered.
  - One way to do this is create a password based on a song title, affirmation, or other phrase.
  - For example, the phrase might be: "This May Be One Way To Remember" and the password could be: "TmB1w2R!" or "Tmb1W>r~" or some other variation.

2.3.2. One-Time Passwords

- Problem with fixed passwords:
  - If an attacker sees a password, he/she can later *replay* the password
- A partial solution: one-time passwords
  - Password that can be used exactly *once*
  - After use, it is immediately invalidated
- Problems
  - Synchronization of user and system
  - Generation of good random passwords
  - Password distribution problem

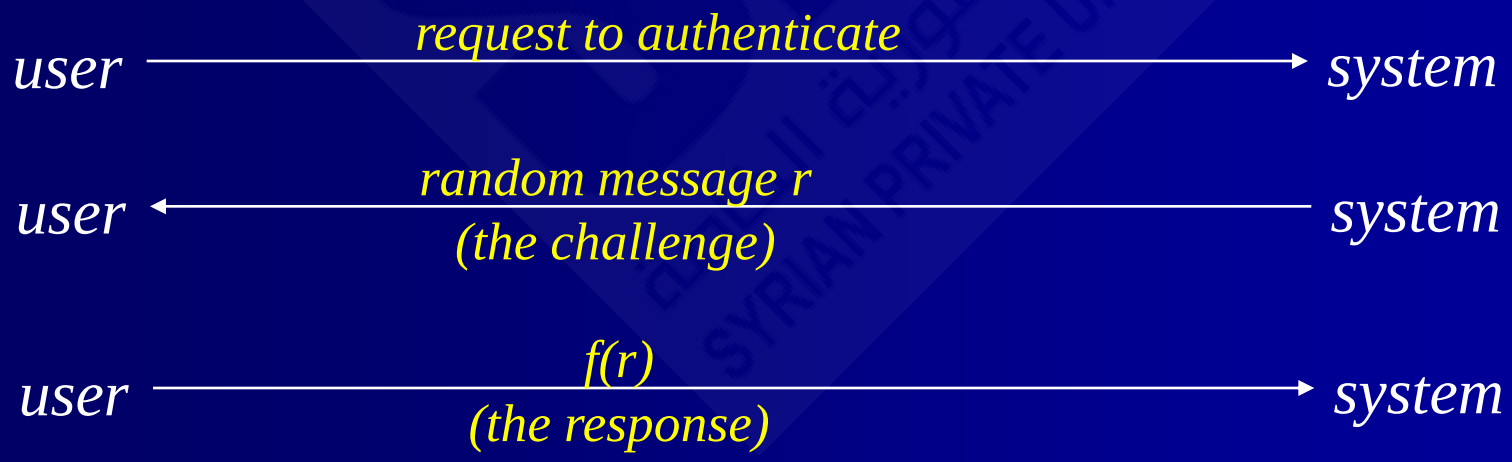
2.3.3. Challenge-Response
(Strong authentication)

- Another alternative is to authenticate in such a way that the transmitted password changes each time
- Let a user *u* wishing to authenticate himself to a system *S*. Let *u* and *S* have an agreed-on secret function *f*.
  - A ***challenge-response*** authentication system is one in which *S* sends a random message *m* (the *challenge*) to *u*, and *u* replies with the transformation $r = f(m)$ (the *response*). *S* then validates *r* by computing it separately.
- The *challenge* may be a ***nonce***, ***timestamp***, ***sequence number***, or any combination.

Challenge-Response

(by symmetric-key techniques)

- The user and system share a secret function *f* (in practice, *f* can be a known function with unknown parameters, such as a cryptographic key).
- This called challenge-response by symmetric-key techniques.



Challenge-Response
(by public-key techniques)

- *A* identifies *B* by checking whether *B* holds the secret (private) key KR_B that matches the public key KU_B
- *A* chooses a random challenge (nonce) r_A. *B* uses its random nonce r_B. *B* applies its public-key system for authentication
- Message sequence:

  1. $A \rightarrow B: r_A$.
  2. $B \rightarrow A: r_B, E_{KR_B}(r_A, r_B)$

3. Cryptographic Key Infrastructure

- Goal: bind identity to key
- Symmetric Cryptography:
  - Not possible as all keys are shared
- Public key Cryptography:
  - Bind identity to public key
  - Crucial as people will use key to communicate with principal whose identity is bound to key
  - Erroneous binding means no secrecy between principals
  - Assume principal identified by an acceptable name

Certificates

- A certificate is a token (message) containing
  - Identity of principal (e.g., Alice)
  - Corresponding public key
  - Timestamp (when issued)
  - Other information (perhaps identity of signer)
  - Signature of a trusted authority (e.g., Cathy)

$$C_A = D_{kv}(K_{u\_a} \parallel \text{Alice} \parallel T)$$

Kv Cathy's private key

C_A *is A's certificate*

Certificate Use

- Bob gets Alice's certificate
  - If he knows Cathy's public key, he can validate the certificate
    - When was certificate issued?
    - Is the principal Alice?
  - Now Bob has Alice's public key

- Problem:
  - Bob needs Cathy's public key to validate *Alice*'s certificate
  - Many solutions:
    - Public Key Infrastructure (PKI),
    - Trust-based certificates (PGP)

X.509 certificate

- Key certificate fields in X.509v3:
  - Version
  - Serial number (unique)
  - Signature algorithm identifier: hash algorithm
  - Issuer's name; uniquely identifies issuer
  - Interval of validity
  - Subject's name; uniquely identifies subject
  - Subject's public key
  - Signature:
    - Identifies algorithm used to sign the certificate
    - Signature (enciphered hash)
- Issuer is called ***Certificate Authority*** (CA)

PKI

- ***A Public Key Infrastructure*** (PKI) is a set of services and policies that lays the framework for binding a public key to an identity and distributing that binding
  - Or in other words, a PKI is a system that provides authentic public keys to applications
- A PKI has three basic processes: Certification, Validation, and Certificate revocation.

1. Certification
   - Binding identities (or attributes) to a public key by a trusted third party to form a certificate.
     - A trusted third part in a PKI is called Certificate Authority (CA),
     - A CA digitally signs a certificate means it vouches for the correctness of the certificate's contents.

PKI

2. Validation
   - The process of verifying the authenticity of the certificate
     - This means that the certificate's contents can be believed.
   - This involves verifying the signature of the CA by using the CA's public key, by checking the validity period contained in the certificate itself, and by checking the certificate against a CRL
   - A CRL (Certificate Revocation List) contains certificate that have been revoked (ie, not valid) by the CA

3. Certificate revocation
   - The process of disavowing a previously issued certificate before its expiration date. This may happen if some of the information contained in the certificate changes.

4. Conclusion

- Entity Authentication
  - Something you have: rarely used alone
  - Something you are: Biometrics
  - Something you know (passwords and secrets)

- Cryptographic Key Infrastructure:
  - authentic Public key
    - entity authentication (based on Challenge-response mechanism)

Questions?