Authentication Mechanisms

Dr. Asem Kitana

Hashing vs. Encryption

- Examples of hashing algorithms (MD5, SHA-256, RIPEMD-160). Encryption algorithms (DES, TDES, AES). What about Diffie-Hellman Algorithm?
- Hashing is Unidirectional (one-way), while encryption is Bidirectional (two-way).
- There is no decoding in hashing.
- Usually hashing for protecting data in storage, while encryption during transport.

Storing Passwords

A password could be stored in a system as:

- Plain password
- Encrypted password
- Hashed password
- Salted password

Often, the hashed passwords are kept in a separate file from the user IDs, referred to as a **shadow password file**.

Improved implementations

- Have stronger, hash/salt variants
- Many systems now use MD5
  - with 48-bit salt
  - password length is unlimited
  - is hashed with 1000 times inner loop
  - produces 128-bit hash

Password choices/concerns

- users may pick short passwords
  - e.g. 3% were 3 chars or less, easily guessed
  - system can reject choices that are too short
- users may pick guessable passwords
  - so crackers use lists of likely passwords
  - e.g. one study of 14000 encrypted passwords guessed nearly 1/4 of them
  - would take about 1 hour on fastest systems to compute all variants, and only need 1 break!

Using Better Passwords

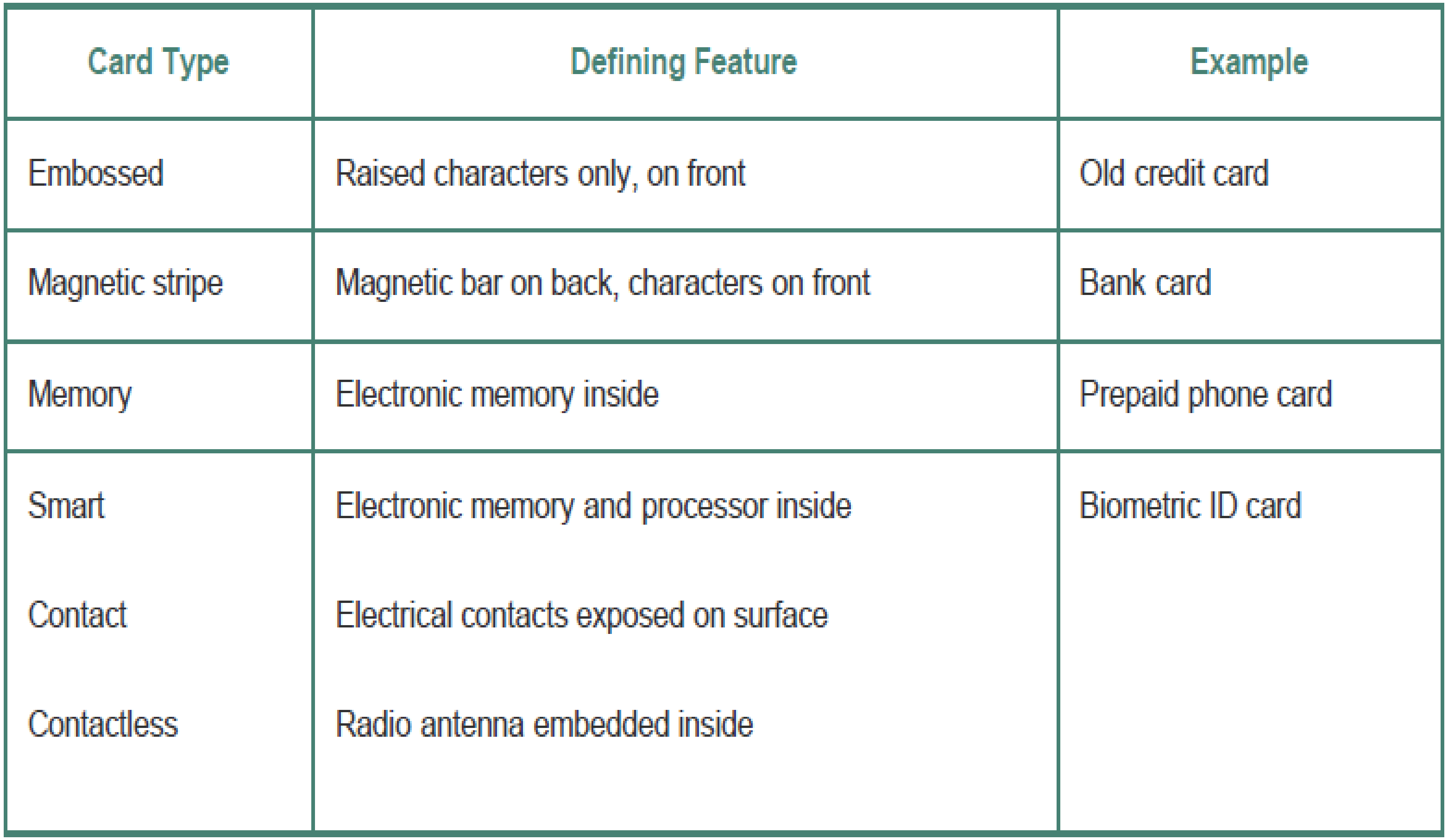
- Clearly have problems with passwords
- Goal to eliminate guessable passwords
  - Still easy for user to remember
- Techniques
  - user education
  - computer-generated passwords
  - reactive password checking (periodic checking)
  - proactive password checking (at the time of selection)

Proactive Password Checking

- Rule enforcement plus user advice, e.g.
  - 8+ chars, upper/lower/numeric/punctuation
  - may not suffice
- Password cracker
  - list of bad passwords
  - time and space issues
- Markov Model
  - generates guessable passwords
  - hence reject any password it might generate
- Bloom Filter
  - use to build table based on dictionary using hashes
  - check desired password against this table

Token-based authentication

- An object a user possesses to authenticate.
- Types of cards used as Tokens:

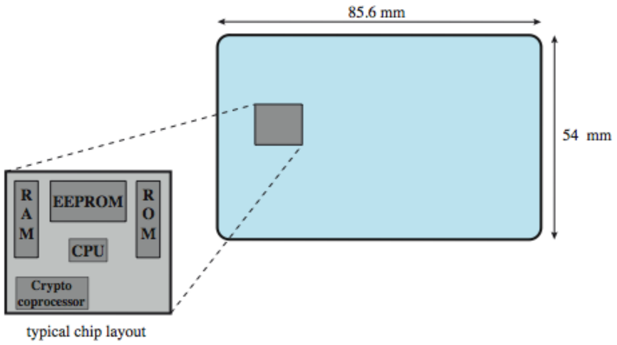


Card Type	Defining Feature	Example
Embossed	Raised characters only, on front	Old credit card
Magnetic stripe	Magnetic bar on back, characters on front	Bank card
Memory	Electronic memory inside	Prepaid phone card
Smart	Electronic memory and processor inside	Biometric ID card
Contact	Electrical contacts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

Memory Card

- store but do not process data
- magnetic stripe card, e.g. bank card
- electronic memory card
- used alone for physical access (e.g., hotel rooms)
- some with password/PIN (e.g., ATMs)
- Drawbacks of memory cards include:
  - need special reader
  - loss of token issues
  - user dissatisfaction (OK for ATM, not OK for computer access)

Smartcard



typical chip layout

- credit-card like
- has own processor, memory, I/O ports
  - ROM, EEPROM, RAM memory
- executes protocol to authenticate with reader/computer

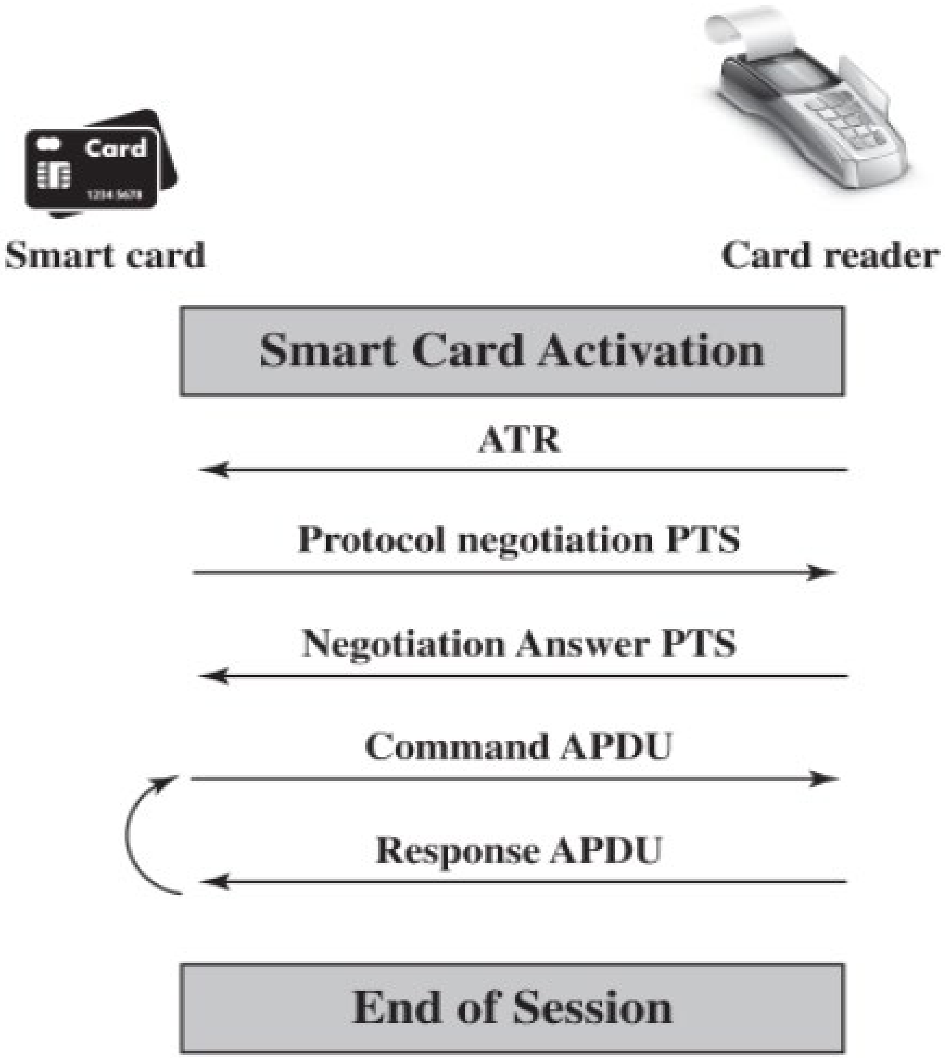
➢ **static:** the user authenticates himself to the token then the token authenticates the user to the computer.

➢ **dynamic:** passwords created every minute; entered manually by user or electronically.

➢ **challenge-response:** computer creates a random number; smart card provides its hash.

- also have USB dongles

Smart card/reader exchange



Smart card

Card reader

Smart Card Activation

ATR

Protocol negotiation PTS

Negotiation Answer PTS

Command APDU

Response APDU

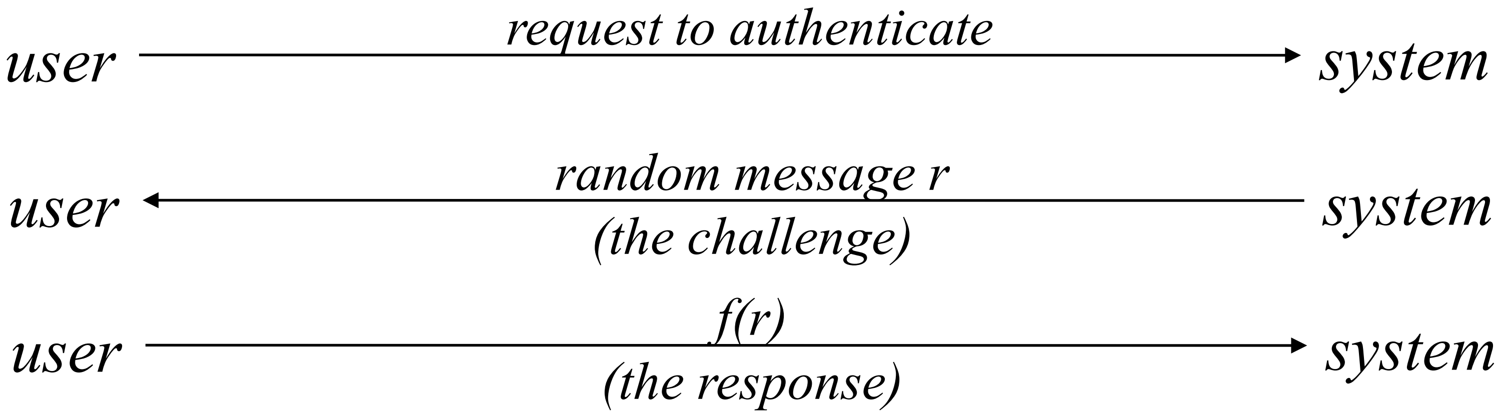
End of Session

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

Remote User Authentication

- Authentication over network more complex
  - Problems of eavesdropping, replay
- Generally use challenge-response
  - user sends identity
  - host responds with random number *r*
  - user computes *f(r,h(P))* and sends back
  - host compares value from user with own computed value, if match user authenticated
- Protects against a number of attacks

Challenge-Response



user → *request to authenticate* → *system*

user ← *random message r* *(the challenge)* ← *system*

user → *f(r)* *(the response)* → *system*

Multi-Factor Authentication (MFA)

- 2FA requires users to provide two types of authentication.
- Typically, this includes a password and a second factor such as a biometric scan, token, or smart card.
- 2FA is more secure than passwords alone but can be more complicated for users.

Single Sign-On (SSO)

- Single sign-on (SSO) is a technology that allows users to authenticate once and access multiple applications or systems.
- SSO eliminates the need for users to enter their credentials for each application or system they need to access.
- SSO can improve security, reduce help desk calls, and improve user productivity.

How SSO Works

- SSO uses a central authentication service to authenticate users.
- Once the user is authenticated, a token is issued that is recognized by other applications or systems.
- The user is then automatically authenticated to any application or system that recognizes the token.

Single Log-Out (SLO)

- Usually the SSO technology combined with SLO feature. Which represents the property whereby a single action of signing out terminates access to multiple software systems.

- Single Log-Out (SLO) also known as Single Sign-Off.

Types of SSO

- Web-Based SSO: Authenticates users for web applications.
- Enterprise SSO: Authenticates users for desktop applications and systems.
- Federated SSO: Authenticates users across multiple organizations or domains.

Benefits of SSO

- Improved Security: SSO reduces the risk of password theft and reuse.
- Increased Productivity: SSO saves time by reducing the number of times users have to enter their credentials.
- Simplified Administration: SSO eliminates the need to manage multiple user accounts and passwords.

Challenges of SSO

- Implementation Complexity: SSO can be complex to implement, especially for legacy applications or systems.
- Integration with Legacy Systems: Legacy systems may not support SSO, requiring additional work to integrate them.
- Security Risks: SSO can create a single point of failure, making it a prime target for attackers.

Common SSO Providers

- Microsoft Azure Active Directory: A cloud-based service that provides SSO for Microsoft applications and other cloud-based applications.

- Okta: A cloud-based service that provides SSO for web applications and other cloud-based applications.

Authentication Security Issues

- eavesdropping
- replay
- trojan horse

Authentication Security Issues

- **Eavesdropping**: attacker attempts to learn passwords by observing the user, finding written passwords, keylogging
  - Countermeasures
    - diligence to keep passwords
    - multifactor authentication
    - admin revoke compromised passwords

Authentication Security Issues

- **Replay**: attacker repeats a previously captured user response
  - Countermeasure
    - Challenge-response
    - 1-time passcodes

Authentication Security Issues

- **Trojan horse**: an application or physical device masquerades as an authentic application or device
  - Countermeasure: authentication of the client within a trusted security environment
- **Denial of service**: attacker attempts to disable a user authentication service (via flooding)
  - Countermeasure: a multifactor authentication with a token

Best Practices for Authentication

- Use strong passwords and enforce password policies.
- Implement MFA whenever possible.
- Keep authentication systems up to date with the latest security patches.
- Monitor and audit authentication logs regularly.

Authentication challenges

- User education and adoption
- Interoperability across systems and applications
- Balancing security with user convenience