CS-3002: Information Security

Lecture # 9: Authentication and Access Control

Department of Software Engineering

FAST-NUCES

Overview

- *Authentication*
- *Passwords*
- *Secure ID*
- *Google 2-step Authentication*
- *Access Control*

Authentication

- **Authentication = binding of an identity to a subject**
- **Confirmation of identity by ...**
  - Knowledge factors = what the entity knows
  - Ownership factors = what the entity has
  - Human factors = what the entity is
  - Location factors = where the entity is

Example

- **Login to a computer**
  - Authentication by *knowledge (password)*
- **Online debit cards**
  - Authentication by *ownership (card) and knowledge (PIN)*
- **Offline debit cards**
  - Authentication by *ownership (card) and human factor (signature)*

Multi-Factor Authentication

- **Authentication using multiple factors**
  - Example: *Scene from the movie "Mission Impossible"*

Ethan Hunt needs to

1. use a stolen chip card (ownership factor)
2. forge a fingerprint (human factor)
3. enter the terminal room (location factor)
4. enter a password (knowledge factor)

Passwords

- **Password =** information confirming the identity of an entity
  - Knowledge of a secret word, phrase or number

- **Often combination with (a)symmetric cryptography**
  - e.g. password is mapped to key of symmetric cipher
  - e.g. password protects private key of public-key algorithm
- ~~Passwords are just great~~. Wait, it’s not that easy

Problems with Passwords

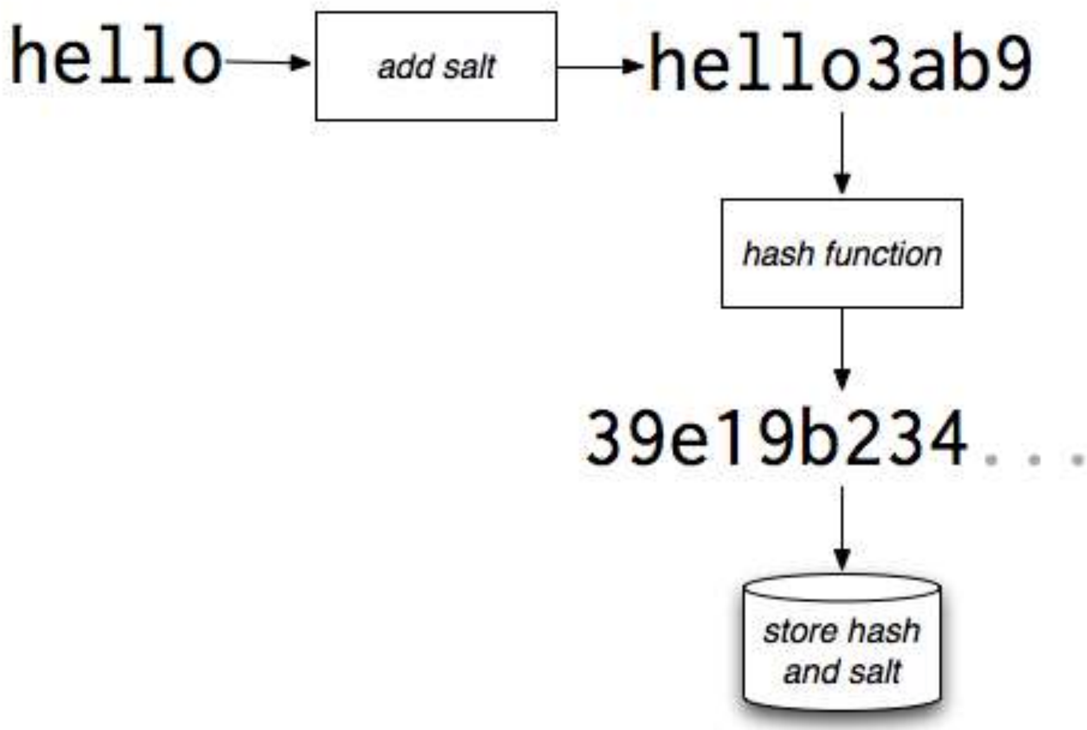
- **Password snooping**
  - Eavesdropping of passwords in network traffic
  - Retrieval of passwords from hosts (e.g. via malware)
- **Password guessing (online) or cracking (offline)**
  - Dictionary attacks = guessing using dictionary of words
  - Brute-force attacks = guessing using all possible strings
- **Human deficiencies**
  - Weak and often re-used passwords

Passwords Storage

- **Passwords should never be stored in clear**
  - Application of cryptographic one-way functions
  - Only encoded (hashed) passwords are stored
    - Sony data breach revealed clear text password.
    - Why twitter auto-reset the passwords recently ?
- **Example: $stored_pw = hash($password);**
  - Simple to validate: hash($input) == $stored_pw?
  - Hard to deduce password from strong hash functions
- **Efficient cracking of stored passwords still possible**
  - Brute-force or dictionary attack using hashed strings

Salted Passwords

- **Encoding of password with random string (salt)**
  - Example: $stored_pw = hash($password+$salt);
  - Salt value stored along with hashed password
- **Cracking of stored passwords more expensive**
  - Same password maps to different hash values
  - Without salt: cracking depends on # words
  - With salt: cracking depends on (# words × # salts)
- **Security depends on quality of password, hash and salt**


hello
add salt
hello3ab9
hash function
39e19b234...
store hash
and salt

Example: Unix Password

- **User credentials stored in two separate databases**
  - /etc/passwd Basic user information (publicly readable)

  - /etc/shadow Salt and hashed passwords (protected)

Good Password?

- **Testing for insecure passwords is *very easy***
  - A normal core i3 laptop can test 21 million MD5 hashs per hour

- **Passwords should be *very hard to guess***
  - No dictionary words, names, dates and patterns
  - Simple transformations (e.g. reversing) not sufficient
  - Minimum length and diversity of passwords

- **Study by Klein from 1989**
  - 21% of 13,797 passwords cracked within one week

Selection of Passwords

- What about these? **Hmh40hcr. and DB:L,I4yF!**

- **Trick: first letters of memorable phrase**
  - "He made him an offer he can't refuse." = **Hmh40hcr.**
  - "Darth Vader: Luke, I am your father!" = **DB:L,I4yF!**

- **Trick: interweave words of memorable phrase**
  - "My kingdom for a horse!" = **KiHor;NgSe**

- **Avoidance of too common phrases**
  - **2bon2b found in 4 out of 30 million passwords**

One-time Passwords

- **Security of passwords "weakens" over time**
  - Password aging = enforced changing of passwords
  - One-time passwords = passwords used exactly once

- **Example: S/Key Algorithm**
  - User chooses initial key K1
  - Recursive hashing: H(K1) = K2, H(K2) = K3,... H(Kn-1) = Kn
  - One-time passwords: P1 = Kn, P2 = Kn-1, ... Pn = K1
  - Hard to deduce next password Pi from previous Pi-1

Example: RSA SecureID

- **Security system using two-factor authentication**
  - Factors: knowledge (password) and ownership (device)
  - Device generates authentication code every 60 seconds
  - Authentication using *password and current code*

- **Code Generation**
  - Device initialized for each user with seed (random number)
  - Code computed from seed and current time (~one-time password)

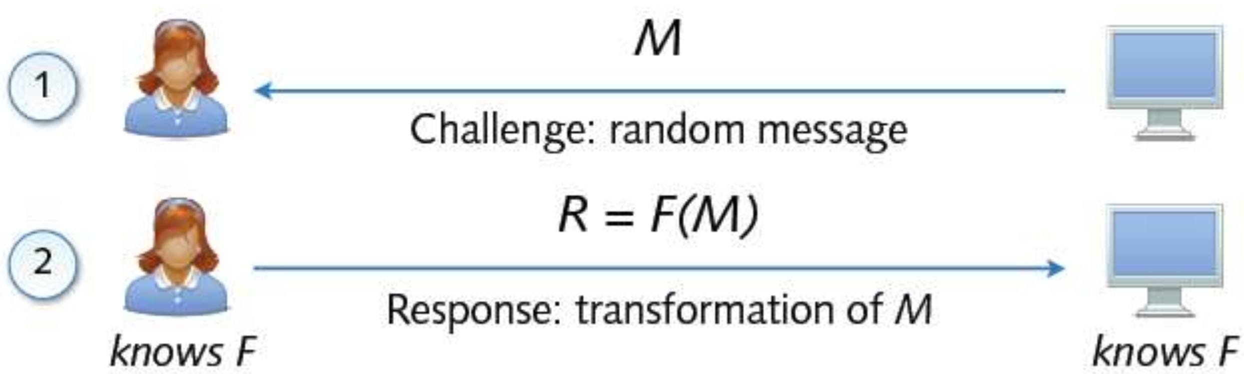
Example: Google 2-Step Verification

- **Security system by Google similar to SecureID**
  - Factors: knowledge (password) and ownership (phone)
  - Authentication code computed on mobile phone
  - Login at Google requires password and current code

https://blog.duosecurity.com/2013/02/bypassing-googles-two-factor-authentication/

Challenge-Reponse

- **Generic protocol scheme for authentication**
  - System and user share a secret function F



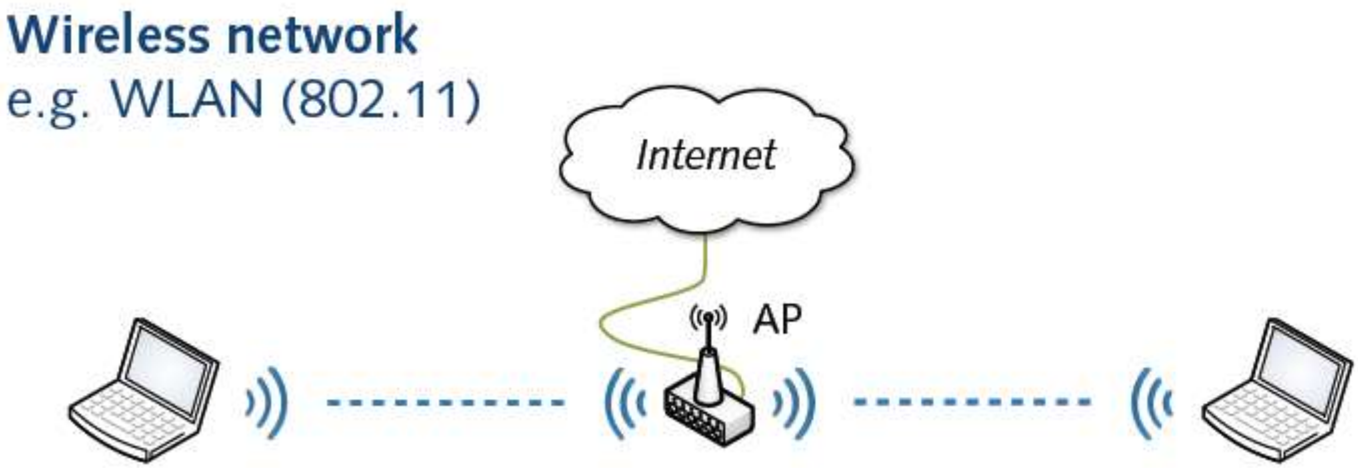
- **Advantages over naive authentication methods**
  - Secret, e.g. password, is never transmitted in cleartext
  - Replay attacks against authentication not possible

Challenge-Response (con’t)

- **Secret function often parameterized by password**
  - F = H(M + P) hash function H and password P
  - F = E_P(M) encryption function E and password P
  - Hard to deduce P if F is cryptographically strong
- **Several methods related to challenge-response scheme**
  - One-time passwords
    - = challenge (index of password); response (password)
  - SecurID / Google 2-step
    - = challenge (current time); response (authentication code)

Example: WPA2 (A Short Excursion)

Wireless Networks



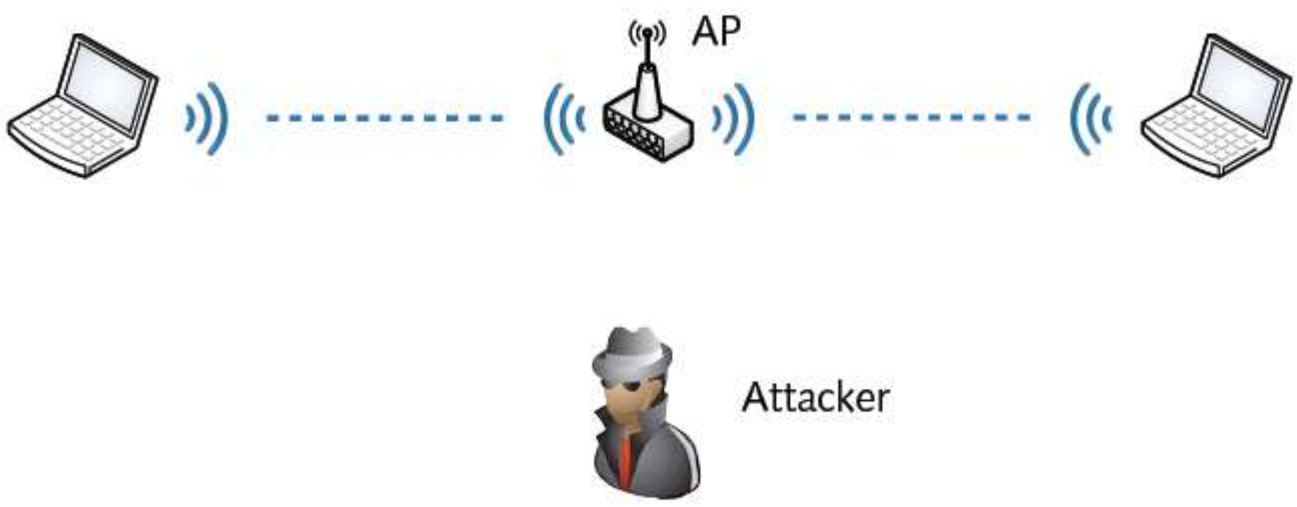
- **Inherent security problems with wireless networks**
  - Communication over shared medium (air)
  - No physical access control and protection
  - Need for additional security measures (WEP, WPA, ...)

A Closer Look at Attacks

- **Common attacks types**
- Masquerading and spoofing
- Eavesdropping of communication
- Tampering of messages

Countermeasures

- Authentication
- Encryption
- Integrity checks

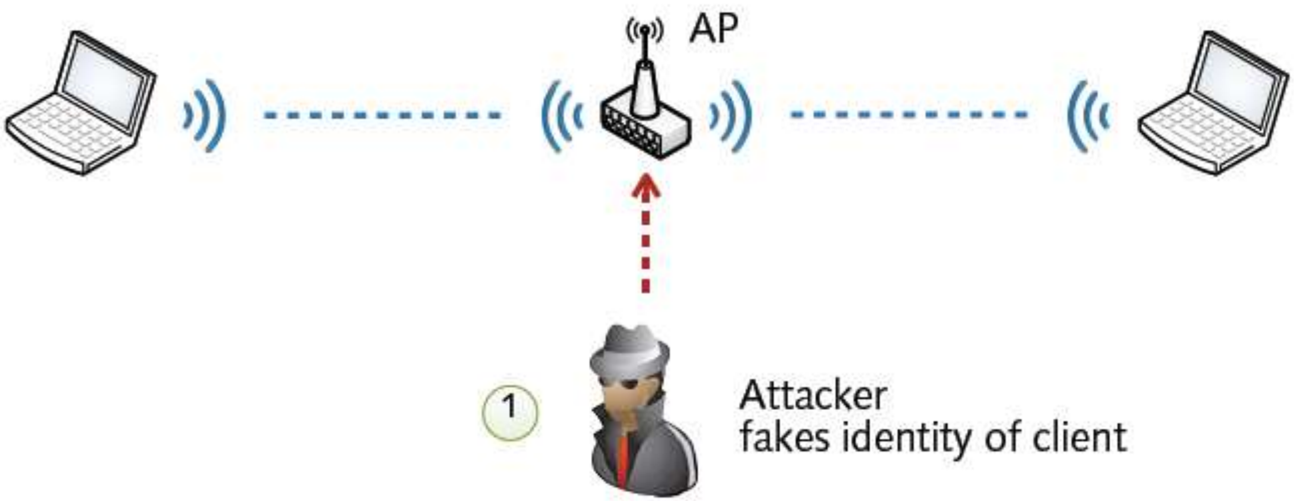


A Closer Look at Attacks

- **Common attacks types**
- Masquerading and spoofing
- Eavesdropping of communication
- Tampering of messages

Countermeasures

- Authentication
- Encryption
- Integrity checks

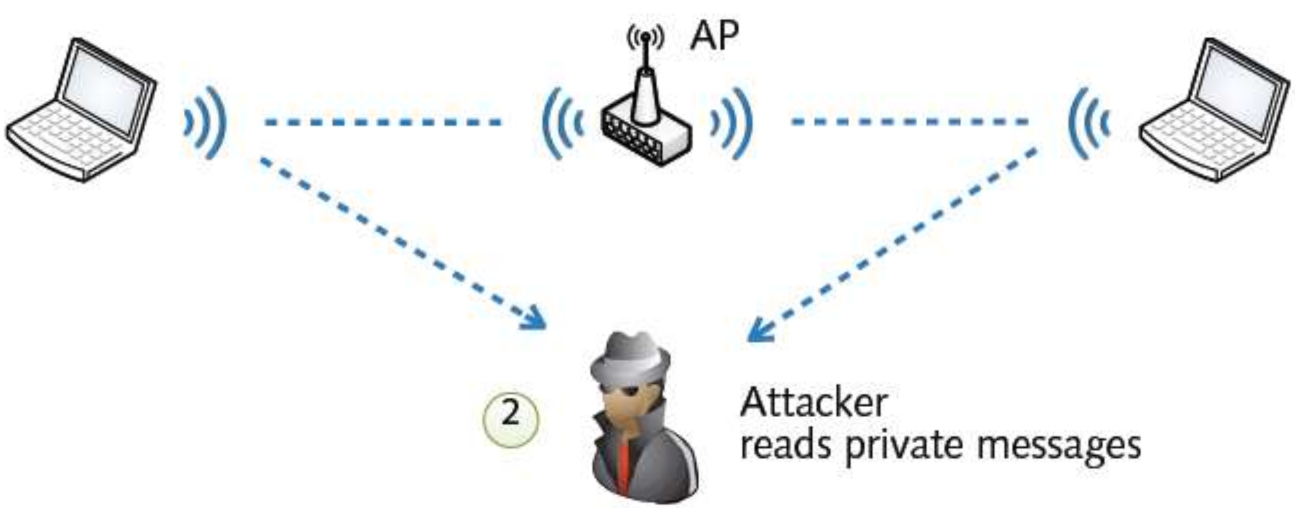


A Closer Look at Attacks

- **Common attacks types**
- Masquerading and spoofing
- Eavesdropping of communication
- Tampering of messages

Countermeasures

- Authentication
- Encryption
- Integrity checks

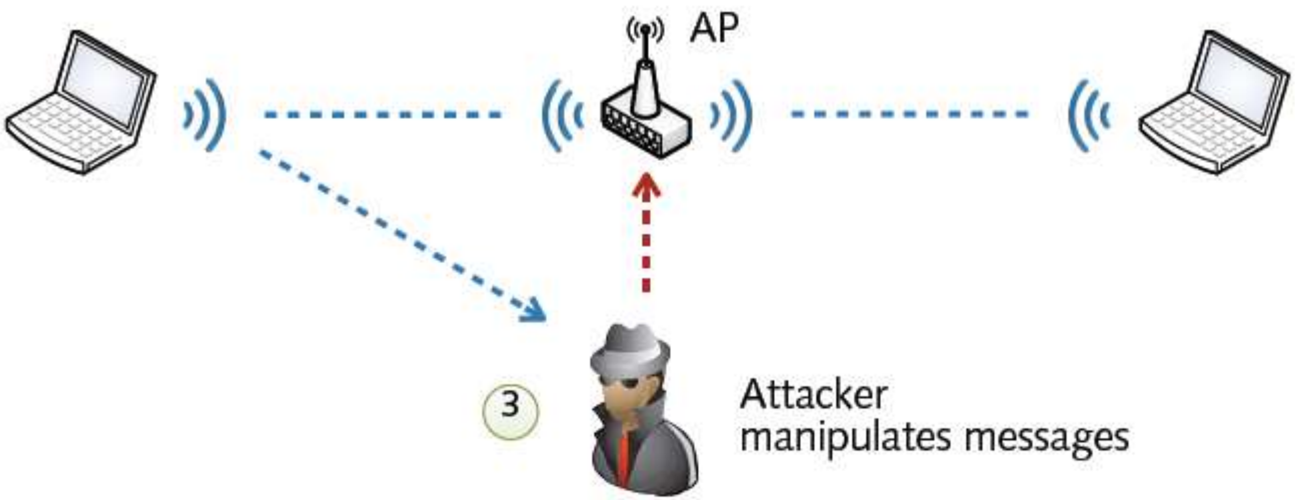


A Closer Look at Attacks

- **Common attacks types**
- Masquerading and spoofing
- Eavesdropping of communication
- Tampering of messages

Countermeasures

- Authentication
- Encryption
- Integrity checks



802.11 and Security

	Authentication	Encryption	Integrity check
1997 WEP (Wired Equivalent Privacy) 802.11a			
	Shared keys	RC4	CRC-32
	medium	weak	weak
2003 WPA (Wi-Fi Protected Access) subset of 802.11i			
	Shared keys / 802.1x	TKIP	Michael
	strong	medium	medium
2004 WPA2 (Wi-Fi Protected Access 2) 802.11i			
	Shared keys / 802.1x	AES-CCMP	AES-CCMP
	strong	strong	strong

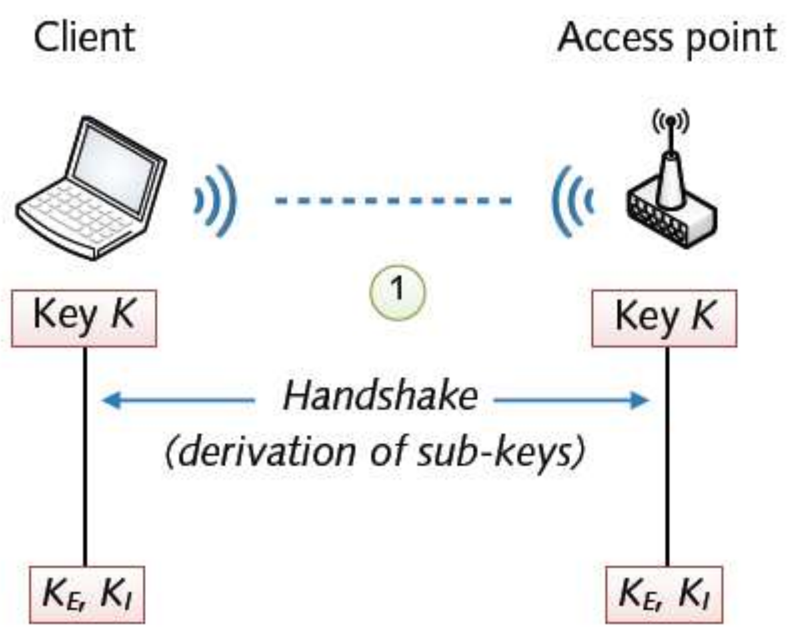
TKIP = Temporal Key Integrity Protocol
AES-CCMP = Counter Cipher Mode with Block Chaining Message Authentication Code Protocol

WPA2 Authentication

- **Two different modes for authentication in WPA2**

1. Personal**: Pre-shared keys (PSK) (aka "passwords")**
2. Enterprise**:** 802.1x with **Extensible Authentication Protocol**

•**Key Exchange**: The client and access point start with a **shared key (Key K)**.
•**Four-way Handshake**: This process generates unique sub-keys for encryption and integrity protection between the client and access point.
•**Sub-keys (Ke, Ki)**: Derived from the original shared key, these sub-keys are used to encrypt and secure the communication between the devices.



Chat gpt content for above slide

- The above slide explains the two different authentication modes in WPA2 and provides an overview of the handshake process involved in establishing a secure connection between a client (like a laptop) and an access point (e.g., a Wi-Fi router).
- **1. WPA2 Authentication Modes:**
- **Personal Mode**:
  - Uses **Pre-shared Key (PSK)**, which is commonly referred to as a "password."
  - This mode is typically used in home or small office networks.
  - The PSK is shared among all devices on the network, and each device must know this password to connect.
- **Enterprise Mode**:
  - Uses **802.1x with Extensible Authentication Protocol (EAP)**.
  - Commonly used in larger organizations, it provides stronger security than PSK.
  - In this mode, each user can have individual login credentials, which are validated through a central authentication server. This enables better control and tracking of users on the network.

WPA2 Authentication

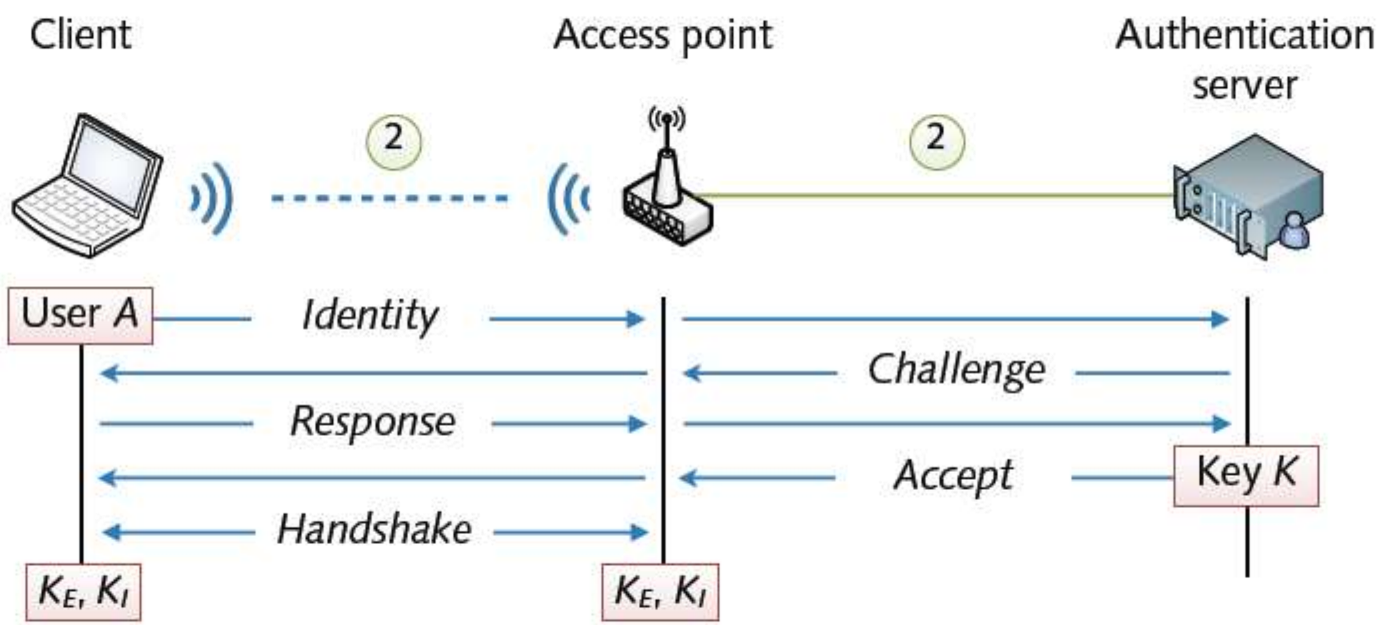
- **Two different modes for authentication in WPA2**

1. Personal**: Pre-shared keys (PSK) (aka "passwords")**
2. Enterprise**:** 802.1x with **Extensible Authentication Protocol**

WPA2 Authentication

- **Two different modes for authentication in WPA2**

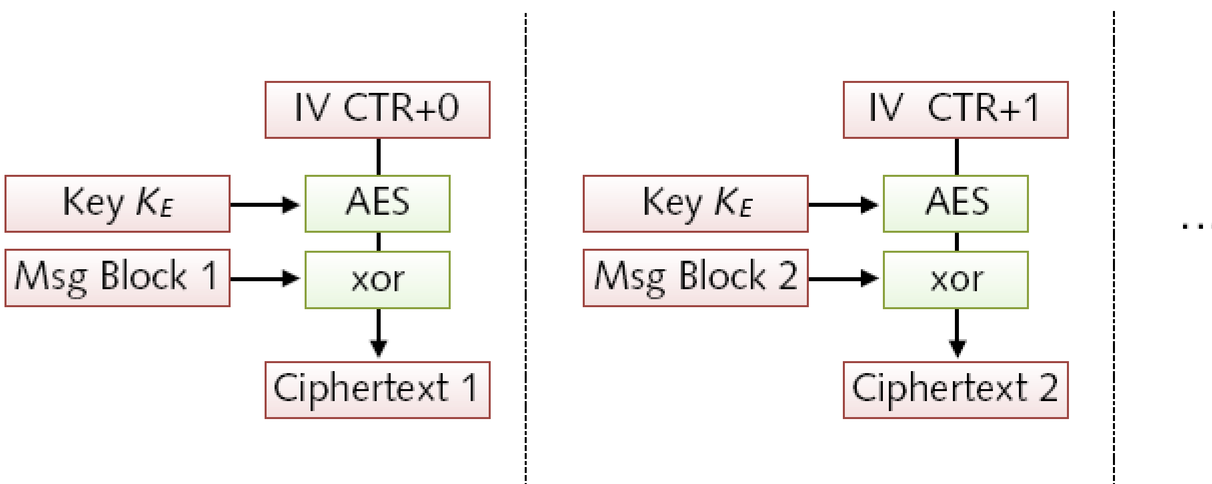
1. Personal**: Pre-shared keys (PSK) (aka "passwords")**
2. Enterprise**:** 802.1x with **Extensible Authentication Protocol**



WPA2 Encryption

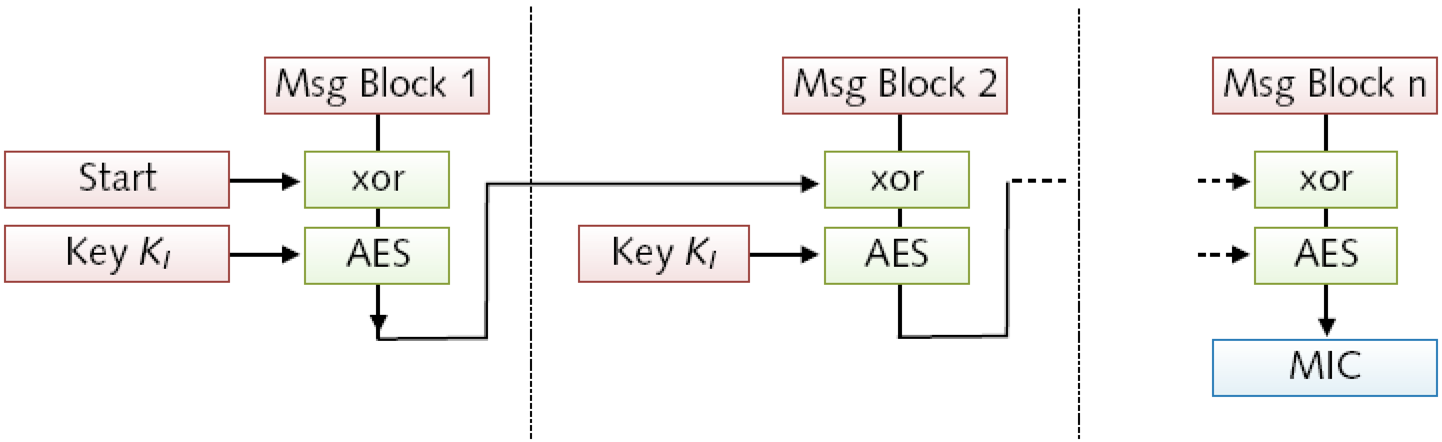
- **Partitioning of each message in blocks**

- **Encryption of each message block in counter mode**
  - Advanced Encryption Standard (AES) using key *KE*

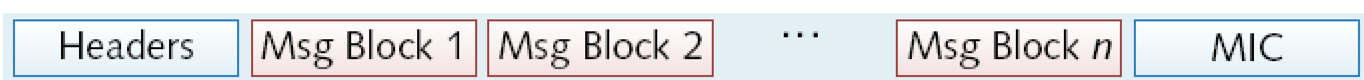


WPA2 Integrity Check

- **Chaining of cipher blocks to a keyed hash value**
  - Message Integrity Code (MIC) using key K_I



- **MIC appended to message prior to encryption**



How secure is WPA2?

- **Attacks against WPA2**
  - (Almost) no attacks against cryptographic protocol
  - Best attack strategy so far: *brute-force attacks*
  - Target for potential attacks: Complexity of protocol

- **WPA2 security in practice**
  - Strength of passphrase in personal mode
  - Strength of authentication protocol in enterprise mode

Access Control

Access Control

- **Authorization and access control**
  - Control of what a subject is allowed to do
  - Management of permissions and capabilities
  - Often tight coupling with authentication

- **Examples**
  - Execution of programs, reading of files, …

Access Control Matrix

Objects

Subjects

	File 1	File 2	Process 1	Process 2
User 1	read		control, send	receive
User 2	write	read	send	
User 3	read, execute	read	control	control

Access control lists

Capabilities

- **Classic and simple representation for access control**
  - Mapping from subjects and objects to permissions

Access Control Models

- **Access control non-trivial in practice**
  - Complex systems ⇝ complex access control models

- **Some characteristics of access control models**
  - Definition of objects and subjects

    E.g. subjects can be users, processes or hosts
  - Representation of permissions

    E.g. columns (access control lists), rows (capabilities)
  - Management of permissions

    E.g. discretionary, mandatory or role-based access control

Representation: Access Control Lists

- **Access control lists (ACL)**
- Attachment of permissions to objects (columns)
- ⊕ Efficient and decentralize organization of permissions
- ⊖ Listing of subject permissions very involved

- ***Example: OpenBSD packet filter***
  - Deny access to the SSH service from any host
  - → *block in quick proto tcp from any to any port ssh*

Representation: Capabilities

- **Capabilities**
- Attachment of permissions to subjects (rows)
- ⊕ Listing and control of subject permissions simple
- ⊖ Fine-grained permissions difficult to implement

- ***Example: Linux capabilities***
  - Restrict permissions to reboot system and load modules
  - → *lcap -z CAP_SYS_BOOT CAP_SYS_MODULE*

Management of Permissions

- **Discretionary Access Control (DAC)**
  - Owner of an object controls access
  - Convenient but insecure if object changes owner

- **Mandatory Access Control (MAC)**
  - System globally enforces access control
  - Very secure but tedious to design and operate

- **Role-based Access Control (RBAC)**
  - System enforces access control using roles
  - In-between DAC and MAC models

Example: UNIX Permissions

- **Discretionary access control of files**
  - Owner manages permissions of his files
- **Fixed-size access control lists: rwx rwx rwx**
  - Three subjects: user, group and other
  - Three permissions: read, write and execute

  - (a) Everybody can read the passwd file; root can write to it
  - (b) Only root and the group shadow can read the shadow file

Example: UNIX Permissions (con't)

- **Simple notation for management of permissions**
  - *<subjects>* +|-|= *<permissions>*
  - Subjects: *u (user)*, *g (group)*, *o (others)*, *a (all)*
  - Permissions: *r (read)*, *w (write)*, *x (execute)*

- **Examples**
  - Make file readable to everyone: *chmod a+r file*
  - Remove write permission from group: *chmod g-w file*
  - Make file readable by user only: *chmod u=r file*

- ***Alternative for UNIX gurus: octal encoding***

Special Permissions

- **Some permissions with special semantics**
  - +x makes directories searchable
  - +t sticky bit (for directories deletion is restricted)
  - +s suid bit (change user id to file owner during execution)

- **A UNIX backdoor from the 1990s**

```
# cp /bin/sh /tmp/.backdoor
# chown root:root /tmp/.backdoor
# chmod u+s /tmp/.backdoor
```

  - If it's bad, why do we need the suid bit?

Acknowledgements

Material in this lecture are taken from the slides prepared by:

- Prof. Dr. Konrad Rieck (Uni-Göttingen)