

# Network Security Protocols

---

- introduction
- secure web transactions: TLS (https)
- network layer security: IPsec
- WiFi security: WEP, WPA, WPA2
- lessons learnt

(c) Levente Buttyán (buttyan@crysys.hu)

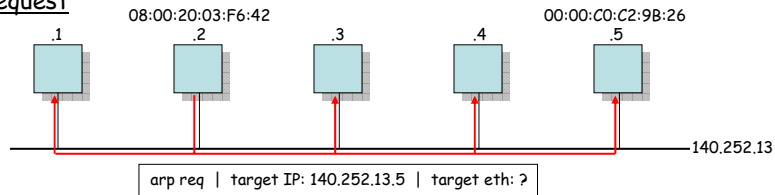
## Introduction

- network security protocols are used to secure communications taking place between two (or more) remote parties over an untrusted network (e.g., the Internet or a wireless network)
- the typical assumption is that the untrusted network is fully under the control of an adversary
  - can eavesdrop messages, alter messages, inject forged messages, delete messages, replay messages, and analyze various traffic properties
- this is far from being an unrealistic assumption
  - e.g., wireless links enjoy no physical protection
  - e.g., compromising a router allows control over all of its traffic
  - e.g., ARP spoofing allows redirection of all packets intended for a given destination to a compromised host on the same LAN

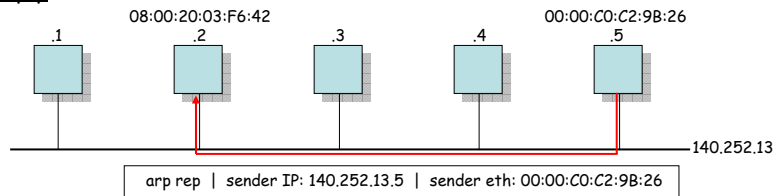
## ARP – Address Resolution Protocol

- mapping from IP addresses to MAC addresses

### Request



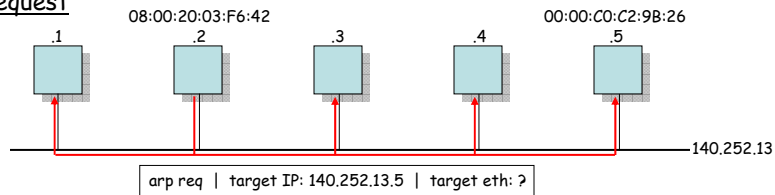
### Reply



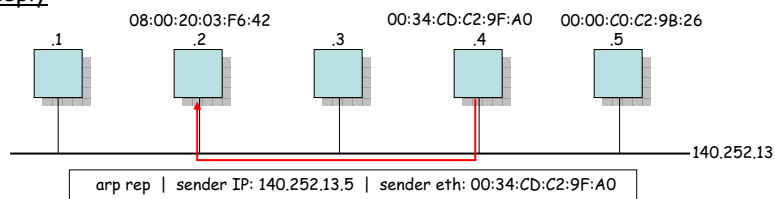
## ARP spoofing

- an ARP request can be responded by another host

### Request



### Reply



## Introduction (cont'd)

- the typical objective of security protocols is to provide the following security services:
  - peer-entity and/or message origin authentication
  - message integrity and replay protection
  - message confidentiality
  - (non-repudiation, privacy, traffic flow confidentiality)
- not surprisingly, network security protocols use cryptographic algorithms as building blocks
  - good examples for how crypto is used in practice today
- designing security protocols is essentially an engineering problem
  - i.e., search for trade-offs between level of protection, efficiency, overhead, usability, ...

## Objectives of this lecture

- explain how some well-known state-of-the-art security protocols work
  - TLS – WWW
  - IPsec – Internet (IP)
  - 802.11i (WPA, WPA2) – WiFi
- demonstrate that communication security functions can be implemented at different layers of the protocol stack
  - TLS – transport layer
  - IPsec – network layer
  - WPA – link layer
- illustrate some security protocol design issues and pitfalls
  - those that each engineer in CS or EE with an MSc degree should be aware of

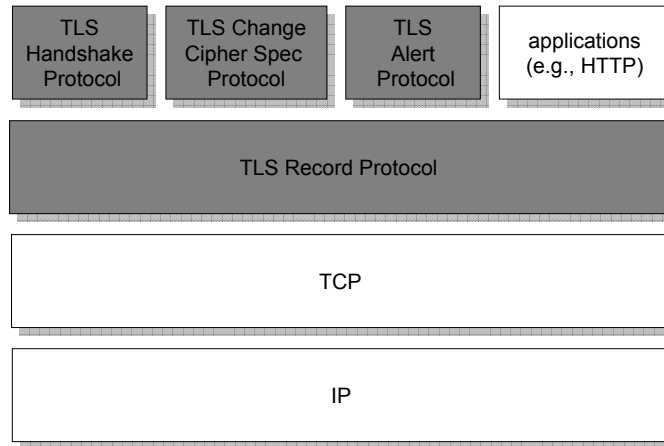
## Lecture outline

- introduction
- **secure web transactions: TLS (https)**
- network layer security: IPsec
- WiFi security: WEP, WPA, WPA2
- lessons learnt

## TLS – Transport Layer Security

- objective
  - to provide a secure transport connection between applications (typically between a web server and a browser → https)
    - mutual authentication of parties
    - encryption, integrity and replay protection of all exchanged messages
- history
  - SSL – Secure Socket Layer
    - developed by Netscape in the mid 90's
    - version 3.0 has been implemented in many web browsers and web servers and become a de-facto standard
  - the IETF adopted it in 1999 under the name TLS
    - TLS v1.0 ~ SSL v3.0 with some design errors corrected
    - further modifications due to some recent attacks → v1.1 → v1.2
    - most recent specification is RFC 5746

## TLS architecture



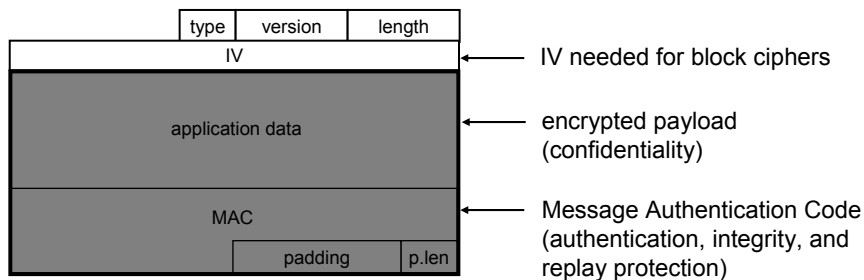
## TLS components

- TLS Handshake Protocol
  - negotiation of security algorithms and parameters
  - key exchange
  - server authentication and optionally client authentication
- TLS Record Protocol
  - fragmentation
  - compression
  - message authentication and integrity protection
  - encryption
- TLS Alert Protocol
  - error messages (fatal alerts and warnings)
- TLS Change Cipher Spec Protocol
  - a single message that indicates the end of the TLS handshake

## Sessions and connections

- a TLS session is a security association between a client and a server
- sessions are stateful; the session state includes:
  - the negotiated security algorithms and parameters
  - certificates (if any)
  - a master secret shared between the client and the server (established during the TLS handshake)
- a session may include multiple secure connections between the same client and server
  - connections of the same session share the session state
  - in addition, each connection has its own connection keys (derived from the master secret) and message sequence numbers
- factoring out the master secret into the session state helps to avoid expensive negotiation of new security parameters for each and every new connection

## TLS Record Protocol



### MAC:

- computed before the encryption
- input to MAC computation:
  - MAC\_write\_key
  - **seq\_num** || type || version || length || payload (compressed fragment)
- supported algorithms: HMAC with MD5, SHA1, SHA256

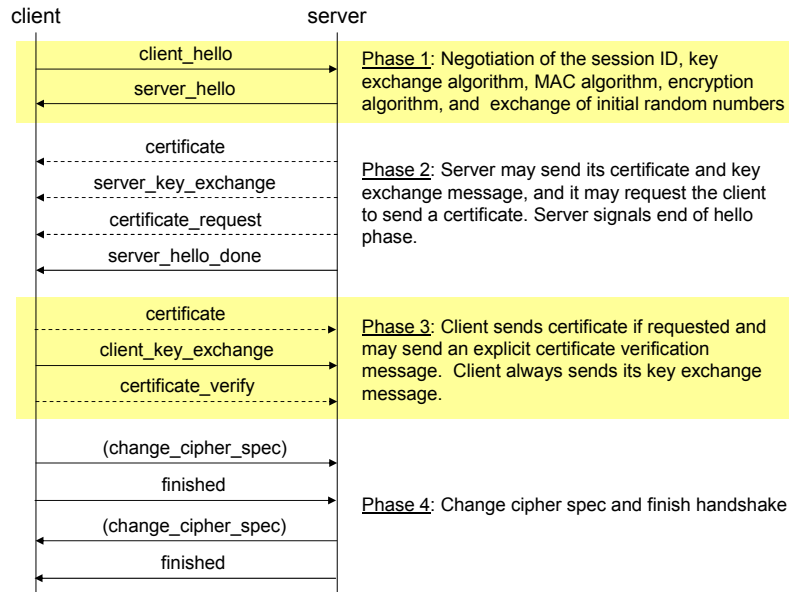
## Encryption in the TLS Record Protocol

- stream ciphers
  - no IV and padding
  - no re-initialization of the cipher for individual messages
  - supported algorithm: RC4 (128)
- block ciphers in CBC mode
  - IV must be a random value
  - padding:
    - last byte is the length  $n$  of the padding (not including the last byte)
    - padding length  $n$  can range from 0 to 255 bytes, but the total length of the encrypted text must be a multiple of the block size of the cipher
    - all padding bytes have value  $n$  → examples for correct padding: x00, x01x01, x02x02x02, ...
    - after decryption, the padding format is verified; if correct, then MAC verification follows
  - supported algorithms: 3DES-EDE, AES (128, 256)
- authenticated encryption modes
  - nonce is carried in the IV field, no padding
  - jointly encrypted and authenticated message is computed from
    - client\_ or server\_write\_key
    - nonce
    - cleartext payload
    - seq\_num || type || version || length
  - supported algorithms: AES-GCM (128, 256), AES-CCM (128, 256)

## TLS Alert Protocol

- each alert message consists of 2 fields (bytes)
- first field (byte): “warning” or “fatal”
- second field (byte):
  - fatal
    - unexpected\_message
    - bad\_record\_MAC
    - decryption\_failure (!)
    - handshake\_failure
    - ...
  - warning
    - close\_notify
    - user\_canceled
    - no\_renegotiation
    - ...
- in case of a fatal alert
  - connection is terminated
  - session ID is invalidated → no new connection can be established within this session

## TLS Handshake Protocol – overview



## Hello messages

- version
  - the highest TLS version supported by the party
- random
  - current time (4 bytes) + pseudo random bytes (28 bytes)
- session\_id
  - if a new session is opened:
    - session\_id of client\_hello is empty
    - session\_id of server\_hello is the new session ID
  - if a new connection is created in an existing session:
    - session\_id of client\_hello is the session ID of the existing session
    - session\_id of server\_hello is the existing session ID if the connection can be created or empty otherwise
- cipher\_suites
  - in client\_hello: list of cipher suites supported by the client ordered by preference
  - in server\_hello: selected cipher suite
  - a cipher suite contains the specification of the key exchange method, the encryption algorithm, and the MAC algorithm
  - exmple: TLS\_RSA\_with\_AES\_128\_CBC\_SHA

## Supported key exchange methods

- RSA based (TLS\_RSA\_with...)
  - the secret key (pre-master secret) is encrypted with the server's public RSA key
  - the server's public key is made available to the client during the exchange
- fixed Diffie-Hellman (TLS\_DH\_RSA\_with... or TLS\_DH\_DSS\_with...)
  - the server has fix DH parameters contained in a certificate signed by a CA
  - the client may have fix DH parameters certified by a CA or it may send an unauthenticated one-time DH public value in the client\_key\_exchange message
- ephemeral Diffie-Hellman (TLS\_DHE\_RSA\_with... or TLS\_DHE\_DSS\_with...)
  - both the server and the client generate one-time DH parameters
  - the server signs its DH parameters with its private RSA or DSS key
  - the client sends an unauthenticated one-time DH public value in the client\_key\_exchange message
  - the client may authenticate itself (if requested by the server) by signing the hash of the handshake messages with its private RSA or DSS key
- anonymous Diffie-Hellman (TLS\_DH\_anon\_with...)
  - both the server and the client generate one-time DH parameters
  - they send their parameters to the peer without authentication

## Server certificate and key exchange messages

- certificate
  - required for every key exchange method except for anonymous DH
  - contains one or a chain of X.509 certificates (up to a known root CA)
  - may contain
    - public RSA key suitable for encryption, or
    - public RSA or DSS key suitable for signing only, or
    - fix DH parameters
- server\_key\_exchange
  - sent only if the certificate does not contain enough information to complete the key exchange (e.g., the certificate contains an RSA signing key only)
  - may contain
    - public RSA key (exponent and modulus), or
    - DH parameters (p, g, public DH value)
  - digitally signed

## Certificate request and server hello done msgs

- **certificate\_request**
  - sent if the client needs to authenticate itself
  - specifies which type of certificate is requested
- **server\_hello\_done**
  - sent to indicate that the server is finished its part of the key exchange
  - after sending this message the server waits for client response
  - the client should verify that the server provided a valid certificate and the server parameters are acceptable

## Client authentication and key exchange messages

- **certificate**
  - sent only if requested by the server
- **client\_key\_exchange**
  - always sent
  - may contain
    - RSA encrypted pre-master secret, or
    - client one-time public DH value
- **certificate\_verify**
  - sent only if the client sent a certificate
  - provides client authentication
  - contains signed hash of all the previous handshake messages from client\_hello up to but not including this message

## Finished messages

- finished
  - sent immediately after the change\_cipher\_spec message
  - used to authenticate all previous handshake messages
  - first message that uses the newly negotiated algorithms and keys
  - computed with a pseudo-random function (see definition later) from the master secret and the hash of all handshake messages

```
PRF( master_secret,  
    "client finished",  
    hash(handshake_messages )
```

```
PRF( master_secret,  
    "server finished",  
    hash(handshake_messages )
```

## The pseudo-random function used in TLS

- $PRF(\text{secret}, \text{label}, \text{seed}) = P\_hash(\text{secret}, \text{label} || \text{seed})$
- $P\_hash(\text{secret}, \text{seed}) = \text{HMAC\_hash}(\text{secret}, A(1) || \text{seed}) ||$   
 $\text{HMAC\_hash}(\text{secret}, A(2) || \text{seed}) ||$   
 $\text{HMAC\_hash}(\text{secret}, A(3) || \text{seed}) ||$

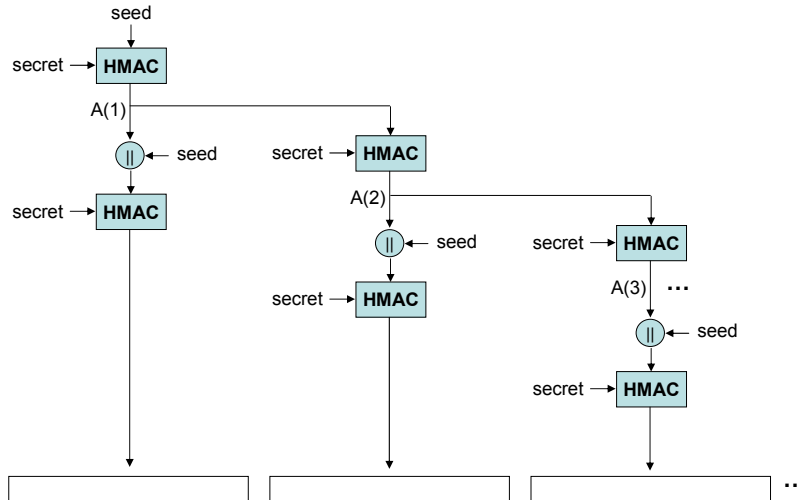
...

where

$A(0) = \text{seed}$

$A(i) = \text{HMAC\_hash}(\text{secret}, A(i-1))$

## P\_hash illustrated



## Key generation

- master secret:  
$$\text{PRF}(\text{pre\_master\_secret}, \text{"master secret"}, \text{client\_random} \parallel \text{server\_random}) \rightarrow 48 \text{ bytes}$$
- connection keys:
  - key block:  
$$\text{PRF}(\text{master\_secret}, \text{"key expansion"}, \text{server\_random} \parallel \text{client\_random}) \rightarrow \text{as many bytes as needed}$$
  - key block is then partitioned:  
$$\text{client\_write\_MAC\_key} \parallel \text{server\_write\_MAC\_key} \parallel \text{client\_write\_key} \parallel \text{server\_write\_key} \parallel \text{client\_write\_IV} \parallel \text{server\_write\_IV}$$

## Key exchange alternatives (examples)

- RSA / no client authentication
  - server sends its encryption capable RSA public key in server\_certificate
  - server\_key\_exchange is not sent
  - client sends encrypted pre-master secret in client\_key\_exchange
  - client\_certificate and certificate\_verify are not sentor
  - server sends its RSA or DSS public signature key in server\_certificate
  - server sends a temporary RSA public key in server\_key\_exchange
  - client sends encrypted pre-master secret in client\_key\_exchange
  - client\_certificate and certificate\_verify are not sent

## Key exchange alternatives (examples cont'd)

- RSA / client is authenticated
  - server sends its encryption capable RSA public key in server\_certificate
  - server\_key\_exchange is not sent
  - client sends its RSA or DSS public signature key in client\_certificate
  - client sends encrypted pre-master secret in client\_key\_exchange
  - client sends signature on all previous handshake messages in certificate\_verifyor
  - server sends its RSA or DSS public signature key in server\_certificate
  - server sends a one-time RSA public key in server\_key\_exchange
  - client sends its RSA or DSS public signature key in client\_certificate
  - client sends encrypted pre-master secret in client\_key\_exchange
  - client sends signature on all previous handshake messages in certificate\_verify

## Summary on TLS

- overall:
  - TLS is a well designed protocol providing strong security
  - current version evolved by considering many known problems of previous versions
  - extremely important for securing web transactions, and for other applications
- TLS Record Protocol
  - good protection against passive eavesdropping and active attacks
  - some protection against traffic analysis (random length padding)
  - uses strong algorithms (HMAC with SHA, AES)
- TLS Handshake Protocol
  - protection against passive and active attacks
  - protection against some rollback attacks (cipher suite rollback, version rollback)
  - uses strong algorithms (RSA, DH, DSS, well designed PRF based on HMAC)

## Lecture outline

- introduction
- secure web transactions: TLS (https)
- **network layer security: IPsec**
- WiFi security: WEP, WPA, WPA2
- lessons learnt

## IPsec

- IPsec is an Internet standard for network layer security (RFC 4301--4309)
  - provides protection for IP and protocols above (ICMP, TCP, ...)
  - allows selection of the required security services and algorithms
  - puts in place the necessary cryptographic keys
  - can be applied between a pair of hosts, between a pair of security gateways (e.g., firewalls), and between a host and a gateway
- components:
  - an authentication protocol (Authentication Header – AH)
  - a combined encryption and authentication protocol (Encapsulated Security Payload – ESP)
  - Security Association and key establishment protocol (IKEv2, not covered in this lecture)
- possible ways to implement IPsec:
  - integration into the native IP stack implementation
  - bump-in-the-stack (BITS): between IP and the network driver
  - bump-in-the-wire (BITW): a separate HW device (security gateway)

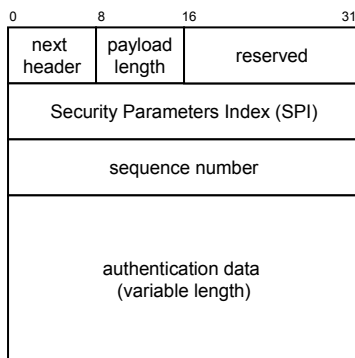
## Modes of operation (both AH and ESP)

- transport mode
  - provides protection primarily for upper layer protocols
  - protection is applied to the payload of the IP packet
    - ESP in transport mode encrypts and optionally authenticates the IP payload but not the IP header
    - AH in transport mode authenticates the IP payload and selected fields of the IP header
  - typically used between end-systems
- tunnel mode
  - provides protection to the entire IP packet
  - the entire IP packet is considered as payload and encapsulated in another IP packet (with potentially different source and destination addresses)
    - ESP in tunnel mode encrypts and optionally authenticates the entire inner IP packet
    - AH in tunnel mode authenticates the entire inner IP packet and selected fields of the outer IP header
  - usually used between two security gateways, or between a host and a security gateway

## Security Associations (SAs)

- the IPsec state shared by two systems (hosts, gateways) is represented by Security Associations (SAs)
- an SA is always one-way, and it is either created for AH or for ESP (or IKE)
- an SA includes the following information:
  - end-point identifiers (e.g., IP addresses)
  - protocol information (algorithms, keys, and related parameters)
  - operational mode (tunnel or transport)
  - message sequence number (for the sender of this SA)
  - anti-replay window (for the receiver of this SA)
    - a counter and a bit-map (or equivalent)
    - used to determine whether an inbound packet is a replay
  - lifetime (a time interval or byte count)
  - ...
- SAs are referenced by an SPI (Security Parameter Index)
  - a bit string carried in AH and ESP headers to allow the receiving party to select the SA which must be used to process the packet

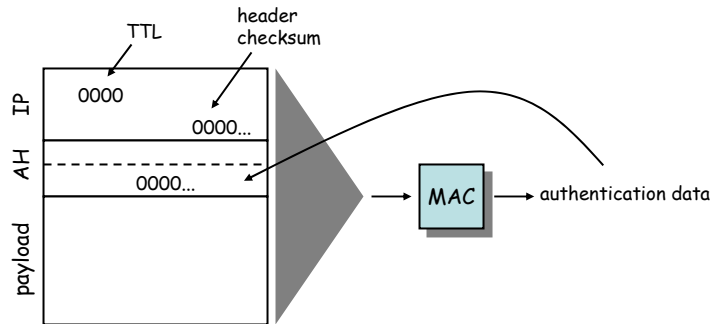
## Authentication Header – AH



- next header
  - type of header immediately following this header (e.g., TCP, IP, etc.)
- payload length
  - length of AH (in 32 bit words)
- Security Parameters Index
  - identifies the SA used to generate this header
- sequence number
  - sequence number of the packet
- authentication data
  - a (truncated) MAC (default length is 3x32 bits)

## MAC

- the MAC is calculated over
  - IP header fields that do not change in transit
  - the AH header fields (authentication data field is set to 0)
  - entire upper layer protocol data unit
- the fields not covered by the MAC are set to 0 for the calculation



## Supported MAC algorithms

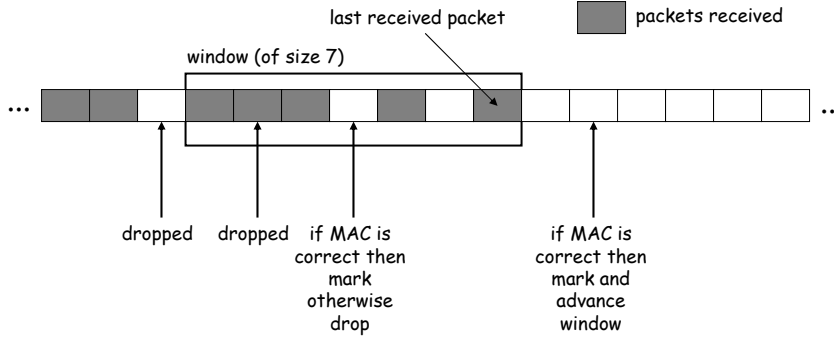
<u>Requirement</u>	<u>Algorithm</u> (notes)
MUST	HMAC-SHA1-96 [RFC2404]
SHOULD+	AES-XCBC-MAC-96 [RFC3566]
MAY	HMAC-MD5-96 [RFC2403] (1)

Note:

(1) Weaknesses have become apparent in MD5; however, these should not affect the use of MD5 with HMAC.

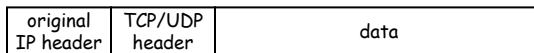
## Replay detection

- replay: the attacker obtains an authenticated packet and later transmits (replays) it to the intended destination
- receiver has an anti-replay window of default size  $W = 64$

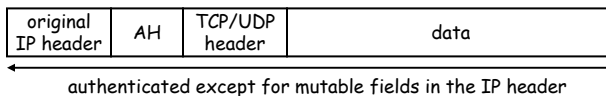


## AH in transport and tunnel mode

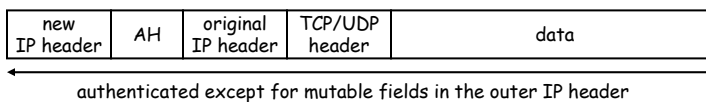
### original IPv4 packet



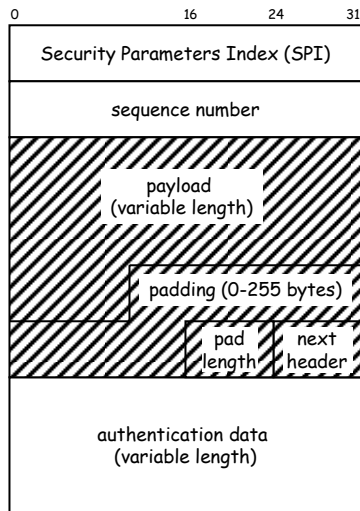
### AH in transport mode



### AH in tunnel mode



## Encapsulating Security Payload – ESP



- Security Parameters Index
  - identifies the SA used to generate this encrypted packet
- sequence number
- payload
  - transport level segment (transfer mode) or encapsulated IP packet (tunnel mode)
- padding
  - variable length padding
- pad length
- next header
  - identifies the type of data contained in the payload
- authentication data
  - a (truncated) MAC computed over the ESP packet (SPI ... next header)

## Encryption and MAC

- encryption
  - applied to the payload, padding, pad length, and next header fields
  - if an IV is needed, then it is explicitly carried at the beginning of the payload data (the IV is not encrypted)
- MAC
  - default length is 3x32 bits
  - MAC is computed over the SPI, sequence number, and encrypted payload, padding, pad length, and next header fields
  - unlike in AH, here the MAC does not cover the preceding IP header

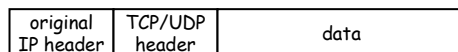
## Supported encryption and MAC algorithms

<u>Requirement</u>	<u>Encryption Algorithm</u>
MUST	NULL
MUST-	TripleDES-CBC [RFC2451]
SHOULD+	AES-CBC with 128-bit keys [RFC3602]
SHOULD	AES-CTR [RFC3686]
SHOULD NOT	DES-CBC [RFC2405]

<u>Requirement</u>	<u>Authentication Algorithm</u>
MUST	HMAC-SHA1-96 [RFC2404]
MUST	NULL
SHOULD+	AES-XCBC-MAC-96 [RFC3566]
MAY	HMAC-MD5-96 [RFC2403]

## ESP in transport and tunnel mode

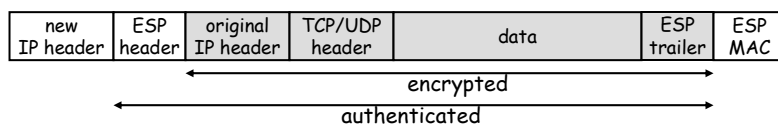
### original IPv4 packet



### ESP in transport mode

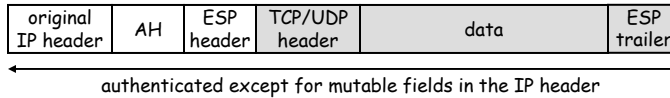


### ESP in tunnel mode



## Combining SAs

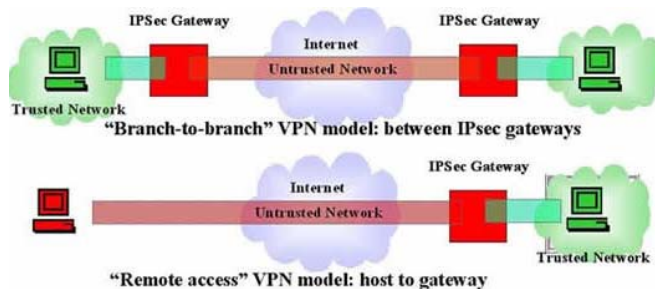
- transport adjacency (basic ESP-AH combination)
  - first apply ESP in transport mode without authentication
  - then apply AH in transport mode



- iterated tunneling (multiple nested tunnels)
  - both end-points of the two tunnels are the same
  - one end-point of the two tunnels is the same
  - neither endpoint of the two tunnels is the same

## Typical application

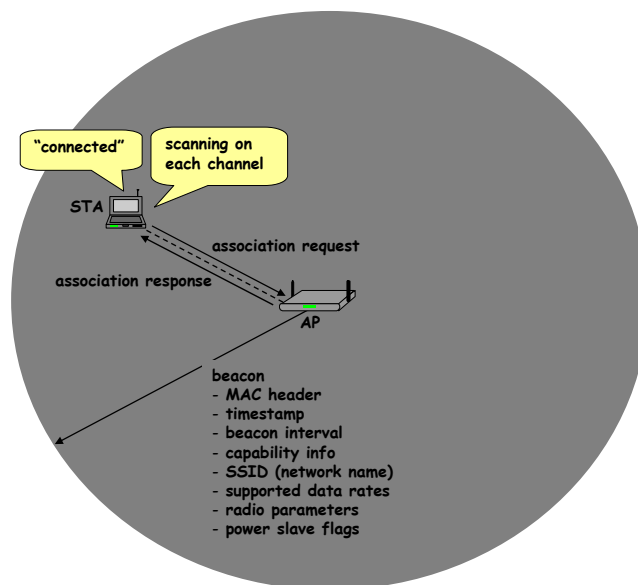
- IPsec based Virtual Private Networks (VPN)



## Lecture outline

- introduction
- secure web transactions: TLS (https)
- network layer security: IPsec
- **WiFi security: WEP, WPA, WPA2**
- lessons learnt

## Brief reminder on the operation of WiFi networks



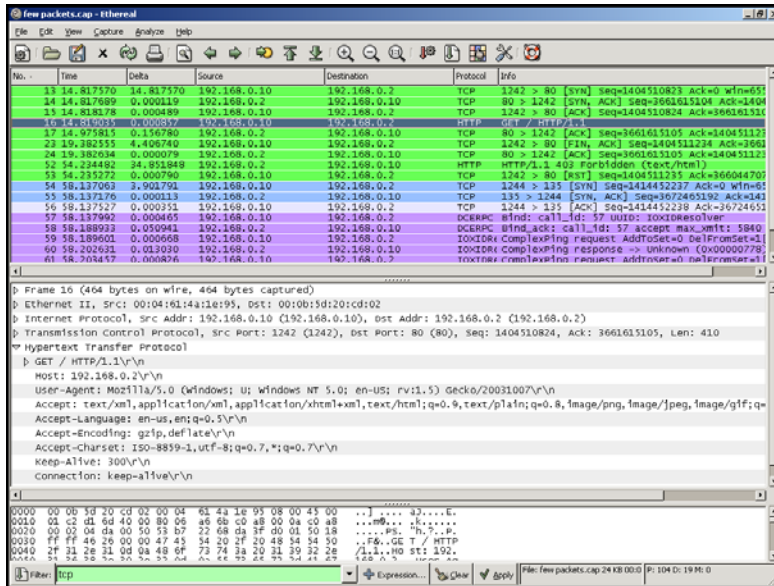
## SSID-based access control

- SSID = Service Set Identifier (network name)
  - a 32-character unique identifier (differentiates one WLAN from another)
  - found in the header of packets and acts as a “password” when a mobile device tries to connect to the WLAN
- unfortunately, the SSID can be sniffed, and hence, this mechanism does not provide really secure access control

## MAC filtering based access control

- MAC address filtering
  - only devices with certain MAC addresses are allowed to associate
  - needs pre-registration of all allowed devices at the AP
- unfortunately, MAC addresses can be sniffed and forged
  - sniffing
    - MAC address is sent in clear in each packet
    - put your WLAN adapter card in promiscuous mode (accepts all packets)
    - eavesdrop the traffic and find out which MAC addresses are accepted
  - forging
    - MAC address of certain WLAN adapter cards can be set by the user
    - example: `# ifconfig ath0 hw ether <mac address of C>`

## Eavesdropping with Wireshark



## Why security is more of a concern in wireless?

- no inherent physical protection
  - physical connections between devices are replaced by logical associations
  - sending and receiving messages do not need physical access to the network infrastructure (cables, hubs, routers, etc.)
- broadcast communications
  - wireless usually means radio, which has a broadcast nature
  - transmissions can be overheard by anyone in range
  - anyone can generate transmissions,
    - which will be received by other devices in range
    - which will interfere with other nearby transmissions and may prevent their correct reception (jamming)
- consequences:
  - eavesdropping is easy
  - injecting bogus messages into the network is easy
  - replaying previously recorded messages is easy
  - illegitimate access to the network and its services is easy
  - denial of service is easily achieved by jamming

## Wireless communication security requirements

- confidentiality
  - messages sent over wireless links must be encrypted
- authenticity
  - origin of messages received over wireless links must be verified
- replay detection
  - freshness of messages received over wireless links must be checked
- integrity
  - modifying messages on-the-fly (during radio transmission) is not so easy, but possible
  - integrity of messages received over wireless links must be verified
- access control
  - access to the network services should be provided only to legitimate entities
  - access control should be permanent
    - it is not enough to check the legitimacy of an entity only when it joins the network and its logical associations are established, because logical associations can be hijacked
- protection against jamming

## WEP – Wired Equivalent Privacy

- part of the IEEE 802.11 specification
- goal
  - make the WiFi network *at least as secure as a wired LAN* (that has no particular protection mechanisms)
  - WEP has never intended to achieve strong security
  - at the end, it achieved no security at all
- services
  - access control to the network
  - message confidentiality
  - message integrity

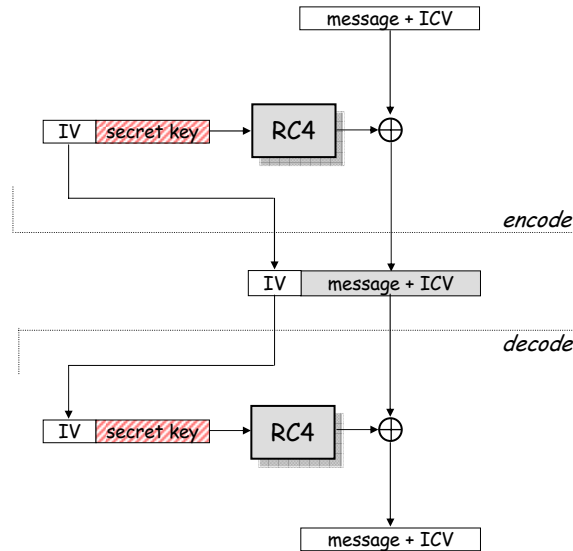
## WEP – Access control

- before association, the STA needs to authenticate itself to the AP
- authentication is based on a simple challenge-response protocol:
  - STA → AP: authenticate request
  - AP → STA: authenticate challenge ( $r$ ) //  $r$  is 128 bits long
  - STA → AP: authenticate response ( $e_K(r)$ )
  - AP → STA: authenticate success/failure
- once authenticated, the STA can send an association request, and the AP will respond with an association response
- if authentication fails, no association is possible

## WEP – Message confidentiality and integrity

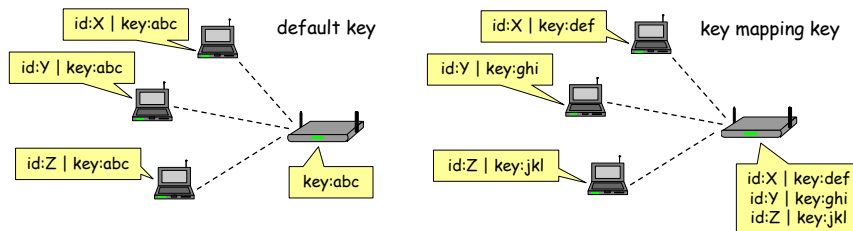
- WEP encryption is based on the RC4 stream cipher
  - it is essential that each message is encrypted with a different key stream
  - the RC4 cipher is initialized with a shared secret key and an IV (initial value)
    - shared secret key (40 or 104 bits) remains unchanged
    - 24-bit IV is changed for every message sent
  - RC4 produces a pseudo-random byte sequence (key stream), which is XORed to the message
  - reception is analogous
- WEP integrity protection is based on an encrypted CRC value
  - ICV (integrity check value) is computed and appended to the message
  - the message and the ICV are encrypted together as described above

## WEP – Message confidentiality and integrity



## WEP – Keys

- two kinds of keys are allowed by the standard
  - default key (also called shared key, group key, multicast key, broadcast key, key)
  - key mapping keys (also called individual key, per-station key, unique key)



- in practice, often only default keys are supported
  - the default key is manually installed in every STA and the AP
  - each STA uses the same shared secret key → in principle, STAs can decrypt each other's messages

## WEP flaws – Authentication and access control

- authentication is one-way only
  - AP is not authenticated to STA
  - STA may associate to a rogue AP which can perform a Man-in-the-Middle attack
- the same shared secret key is used for authentication and encryption
  - weaknesses in any of the two protocol can be used to break the key
  - different keys for different functions are desirable
- no session key is established during authentication
  - access control is not continuous
  - once a STA has authenticated and associated to the AP, an attacker can send messages using the MAC address of STA
  - correctly encrypted messages cannot be produced by the attacker, but replay of STA messages is still possible
- STA can be impersonated
  - ... next slide

## WEP flaws – Authentication and access control

- recall that authentication is based on a challenge-response protocol:
    - ...
    - AP → STA: r
    - STA → AP: IV | r ⊕ K
    - ...
- where K is a 128 bit RC4 output on IV and the shared secret
- an attacker can compute  $r \oplus (r \oplus K) = K$
  - she can use K (and the same IV) to impersonate STA later:
    - ...
    - AP → attacker: r'
    - attacker → AP: IV | r' ⊕ K
    - ...

## WEP flaws – Integrity and replay protection

- there's no replay protection at all
  - IV is not mandated to be incremented after each message
  - receiver is not mandated to check the freshness of received IVs
- attacker can manipulate messages despite the ICV mechanism and encryption
  - CRC is a linear function wrt to XOR:

$$\text{CRC}(X \oplus Y) = \text{CRC}(X) \oplus \text{CRC}(Y)$$

- attacker observes  $(M \parallel \text{CRC}(M)) \oplus K$  where  $K$  is the RC4 output
- for any  $\Delta M$ , the attacker can compute  $\text{CRC}(\Delta M)$
- hence, the attacker can compute:

$$\begin{aligned} & ((M \parallel \text{CRC}(M)) \oplus K) \oplus (\Delta M \parallel \text{CRC}(\Delta M)) = \\ & ((M \oplus \Delta M) \parallel (\text{CRC}(M) \oplus \text{CRC}(\Delta M))) \oplus K = \\ & ((M \oplus \Delta M) \parallel \text{CRC}(M \oplus \Delta M)) \oplus K \end{aligned}$$

## WEP flaws – Confidentiality

- IV reuse
  - IV space is too small
    - IV size is only 24 bits → there are 16,777,216 possible IVs
    - after around 17 million messages, IVs are reused
    - a busy AP at 11 Mbps is capable for transmitting 700 packets per second → IV space is used up in around 7 hours
  - in many implementations IVs are initialized with 0 on startup and then incremented by one after each message
    - if several devices are switched on nearly at the same time, they all use the same sequence of IVs
    - if they all use the same default key (which is the common case), then IV collisions are readily available to an attacker
- exploiting weaknesses in the RC4 cipher
  - tools are readily available from the Internet (e.g., <http://www.aircrack-ng.org>)

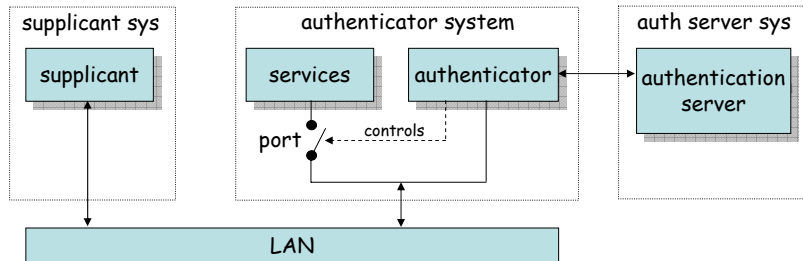
## Overview of 802.11i

- after the collapse of WEP, IEEE started to develop a new security architecture → 802.11i (now integrated in 802.11)
- main novelties in 802.11i wrt to WEP
  - access control model is based on 802.1X
  - flexible authentication framework (based on EAP)
  - authentication can be based on strong protocols (e.g., TLS)
  - authentication process results in a shared session key (which prevents session hijacking)
  - different functions (encryption, integrity) use different keys derived from the session key using a one-way function
  - integrity and replay protection is improved
  - encryption function is improved

## 802.11i vs. WPA and WPA2

- WPA (Wifi Protected Access)
  - industrial name for 802.11i TKIP (Temporal Key Integrity Protocol)
  - integrity protection is based on Michael
  - IV is used as replay counter too
  - encryption is based on RC4, but WEP's problems have been avoided
    - IV length is increased to 48 bits in order to prevent IV reuse
    - per-packet keys are used to prevent known attacks on RC4
  - ugly solution, but runs on old hardware (after software upgrade)
- WPA2
  - industrial name for 802.11i AES-CCMP (Counter mode and CBC-MAC Protocol)
  - integrity protection and encryption is based on AES in CCM mode
    - CBC-MAC
      - computed over the MAC header, CCMP header, and the MPDU (fragmented data)
      - mutable fields are set to zero
      - input is padded with zeros if length is not multiple of 128 (bits)
      - final 128-bit block of CBC encryption is truncated to (upper) 64 bits to get the CBC-MAC value
    - CTR mode encryption
      - MPDU and CBC-MAC value are encrypted, MAC and CCMP headers are not
  - nice solution, but needs new hardware

## 802.1X authentication model



- the supplicant requests access to the services (wants to connect to the network)
- the authenticator controls access to the services (controls the state of a port)
- the authentication server authorizes access to the services
  - the supplicant authenticates itself to the authentication server
  - if the authentication is successful, the authentication server instructs the authenticator to switch the port on
  - the authentication server informs the supplicant that access is allowed

## Mapping the 802.1X model to WiFi

- supplicant → mobile device (STA)
- authenticator → access point (AP)
- authentication server → server application running on the AP or on a dedicated machine
- port → logical state implemented in software in the AP
- one more thing is added to the basic 802.1X model in 802.11i:
  - successful authentication results not only in switching the port on, but also in a session key between the mobile device and the authentication server
  - the session key is sent to the AP in a secure way
    - this assumes a shared key between the AP and the auth server
    - this key is usually set up manually

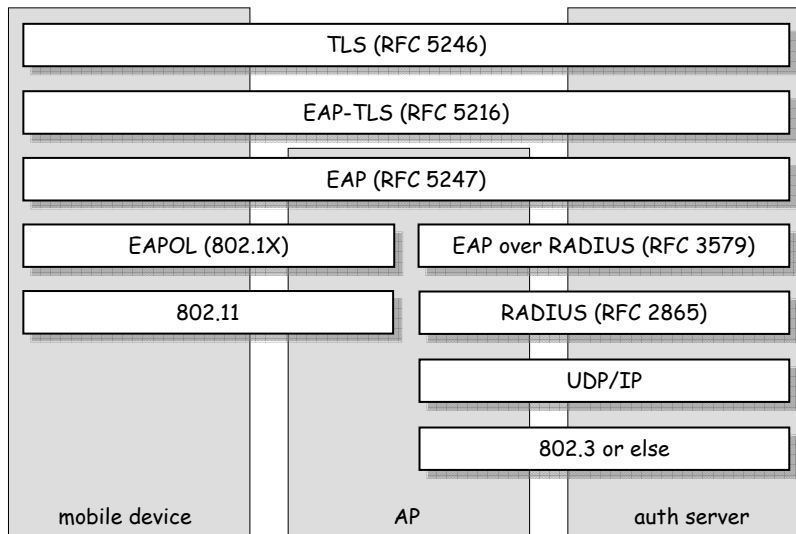
## Protocols – EAP, EAPOL, and RADIUS

- EAP (Extensible Authentication Protocol) [RFC 3748]
  - carrier protocol designed to transport the messages of “real” authentication protocols (e.g., TLS)
  - very simple, four types of messages:
    - EAP request – carries messages from the supplicant to the authentication server
    - EAP response – carries messages from the authentication server to the supplicant
    - EAP success – signals successful authentication
    - EAP failure – signals authentication failure
  - authenticator (AP) doesn't understand what is inside the EAP messages, it recognizes only EAP success and failure
- EAPOL (EAP over LAN) [802.1X]
  - used to encapsulate EAP messages into LAN protocols (e.g., Ethernet)
  - EAPOL is used to carry EAP messages between the STA and the AP
- RADIUS (Remote Access Dial-In User Service) [RFC 2865-2869, RFC 2548]
  - used to carry EAP messages between the AP and the auth server
  - MS-MPPE-Recv-Key attribute is used to transport the session key from the auth server to the AP
  - RADIUS is mandated by WPA and optional for WPA2

## Protocols –EAP-TLS, EAP-TTLS

- EAP-TLS (TLS over EAP) [RFC 5216]
  - only the TLS Handshake Protocol is used
  - server and client authentication, generation of master secret
  - TLS master secret becomes the session key
  - mandated by WPA, optional in WPA2
- EAP-TTLS (Tunneled TLS over EAP) [RFC 5281]
  - phase 1: TLS Handshake possibly without client authentication
  - phase 2: legacy client authentication (e.g., password based) protected by the secure tunnel established in phase 1
    - eavesdropping and man-in-the-middle attacks are prevented
    - privacy is improved (user name is also encrypted)
- EAP-PSK, EAP-FAST, EAP-PEAP, EAP-SIM, EAP-AKA, ...

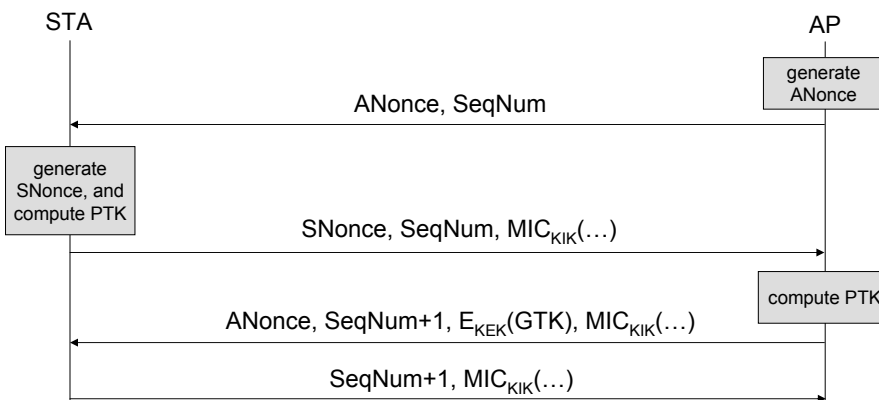
## Summary of the protocol architecture



## Four way handshake (simplified)

objective:

- confirm knowledge of the PMK (Pairwise Master Key resulted from the authentication between STA and Auth Server)
- exchange random values to be used in the generation of session keys



MIC() and E(): HMAC-MD5 and RC4 or HMAC-SHA and AES

## Session keys

- PTK (Pairwise Transient Keys)
  - key encryption key and key integrity key
    - used to protect distribution of group keys (see GTK below)
  - data encryption key and data integrity key
    - used to protect unicast messages exchanged between the AP and STA
  - computation of these keys is based on the PMK, the exchanged nonces, and the MAC addresses
- GTK (Group Transient Keys)
  - group encryption key and group integrity key
    - used to protect broadcast messages sent by the AP to all STAs
  - these keys are generated by the AP

## Summary on WiFi security

- security has always been considered important for WiFi
- early solution was based on WEP
  - seriously flawed !
  - not recommended to use anymore
- the new security standard for WiFi is 802.11i (WPA and WPA2)
  - access control model is based on 802.1X
  - flexible authentication based on EAP and upper layer authentication protocols (e.g., TLS)
  - improved key management
  - WPA (TKIP)
    - uses RC4 → runs on old hardware, but corrects (most of) WEP's flaws
    - WPA has been recently broken ! (paper at ACM WiSec 2009, March 16-18)
  - WPA2 (AES-CCMP)
    - uses AES in CCMP mode (an authenticated encryption mode)
    - needs new hardware that supports AES

## Lessons learnt

- the same security services may be implemented at different layers of the protocol stack
- security protocol design is an engineering problem
  - one needs to find a good trade-off among security, efficiency, and usability
  - problems are often discovered, upgrade is necessary
- main difficulty: security is a non-composable property
  - you may combine otherwise strong building blocks in a wrong way and obtain an insecure system at the end
  - example:
    - stream ciphers alone are OK
    - challenge-response protocols for entity authentication are OK
    - but they shouldn't be combined in a way done in WEP
  - example:
    - encrypting a message digest to obtain a MAC can be acceptable
    - but it doesn't work if the message digest function is linear wrt to the encryption function