

TRANSPORT LAYER SECURITY

A Brief Introduction...

TABLE OF CONTENTS

- The CIA Triad
- SSL/TLS
 - TLS Record Layer
 - TLS Handshake
 - TLS Encryption
 - Cipher Suites
 - Extensions
- Example
- Attacks to TLS

THE CIA TRIAD

The first wave of **network protocols**

ARP, IP, TCP, UDP, HTTP, FTP, ...

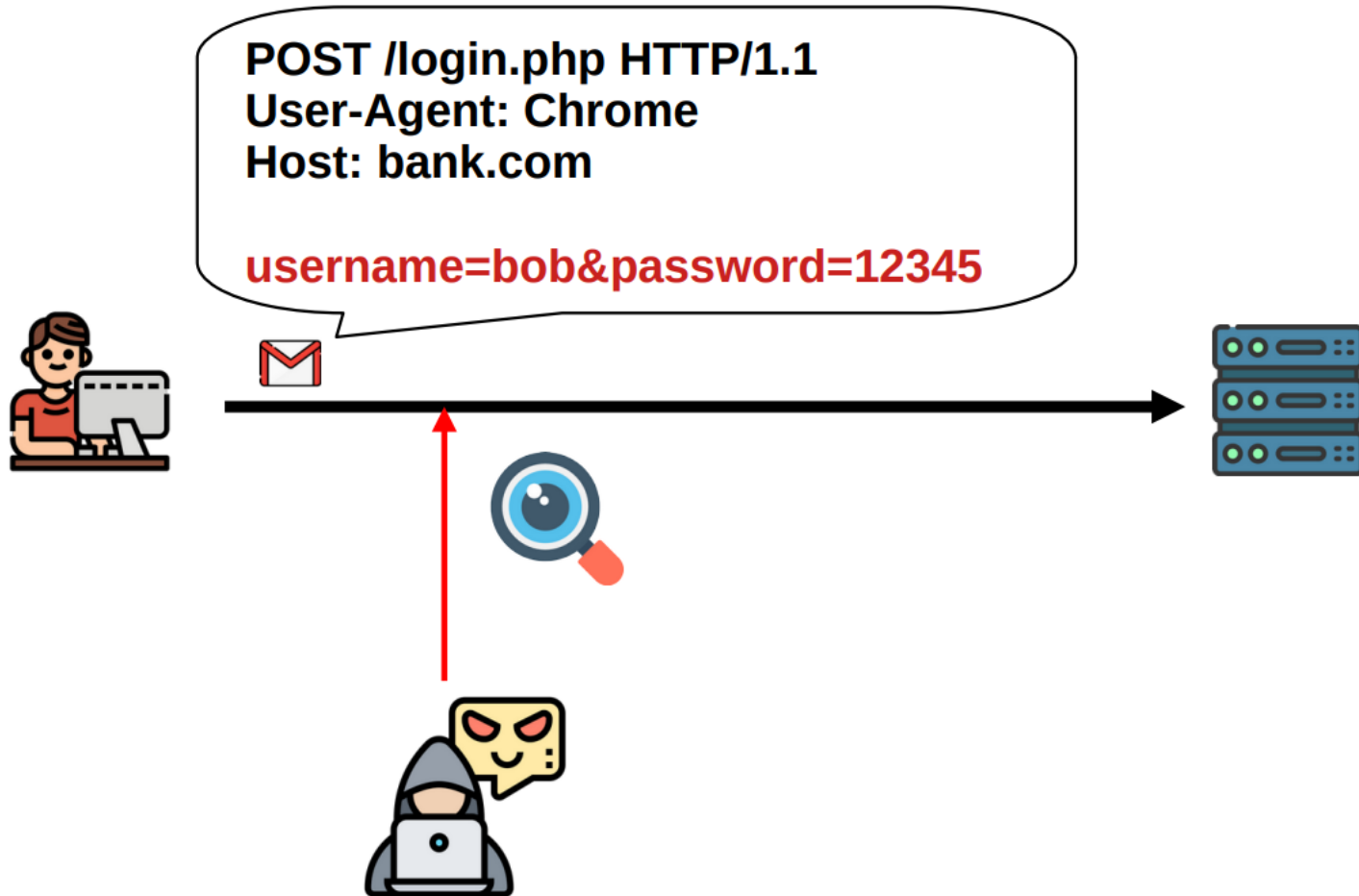
did not offer any **cryptographic services** such as

- **confidentiality**
- **integrity**
- **authentication**

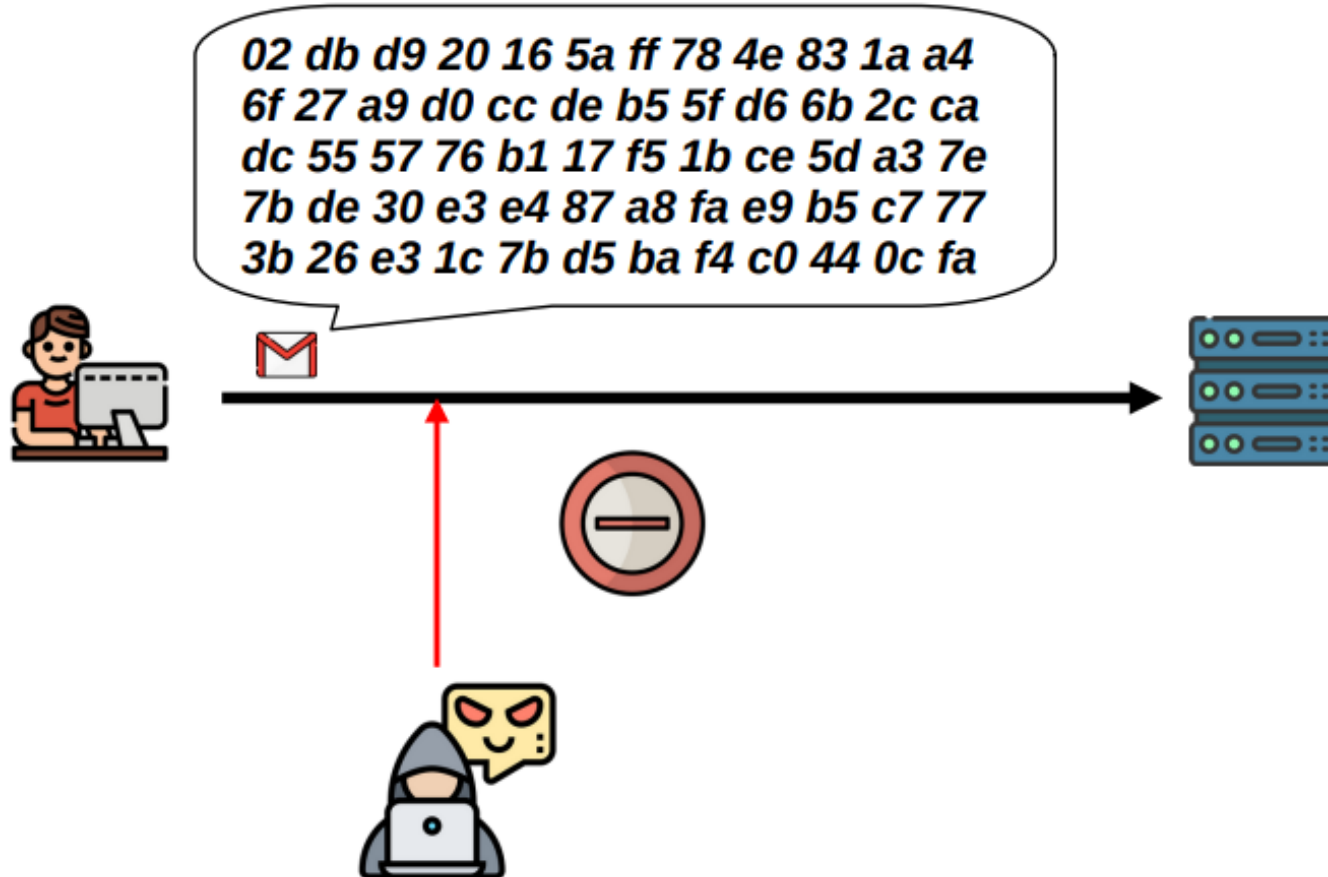
Confidentiality (1/3)

We have **confidentiality** when data, objects and resources are protected from unauthorized viewing and other access.

Confidentiality (2/3)



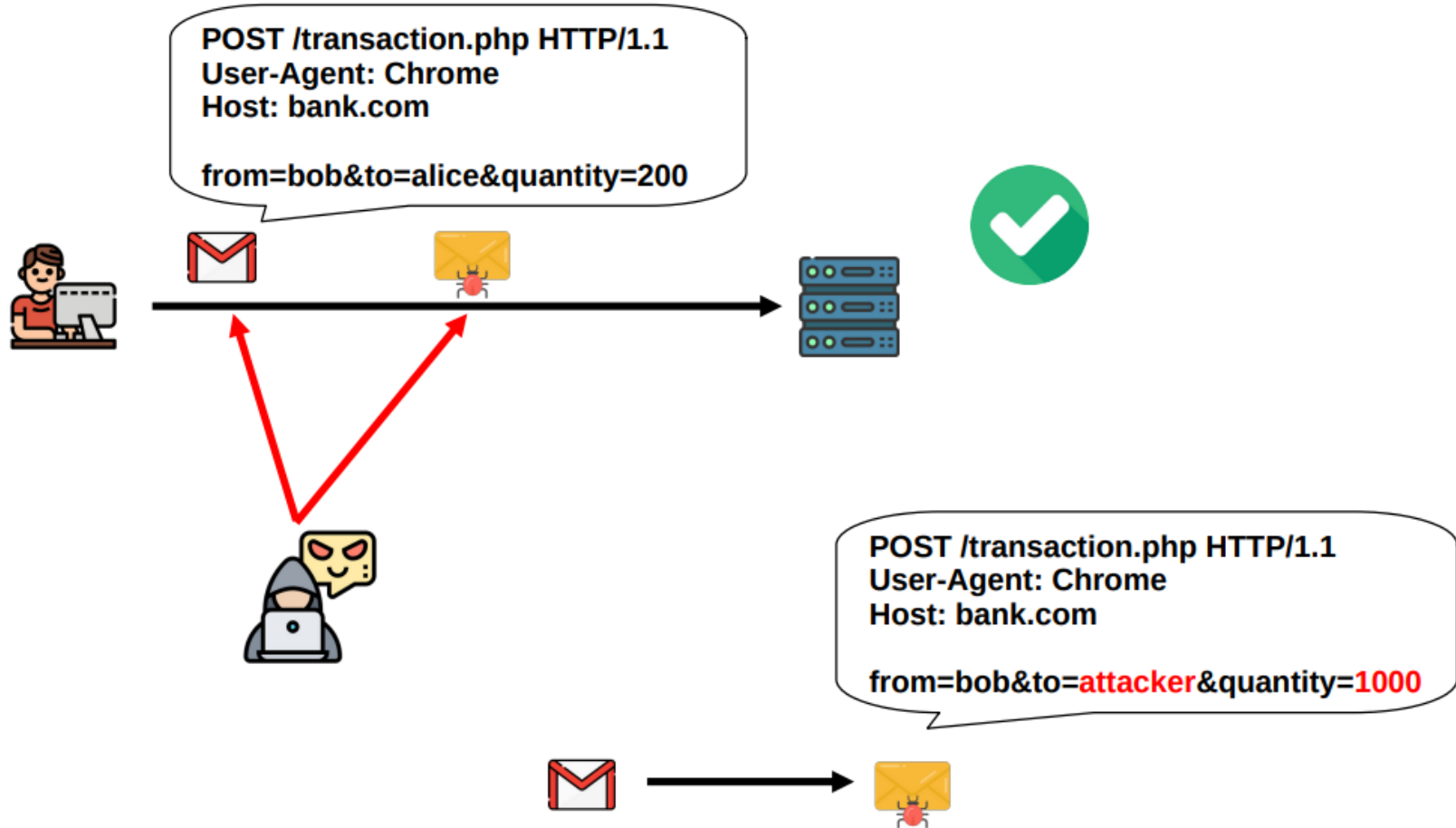
Confidentiality (3/3)



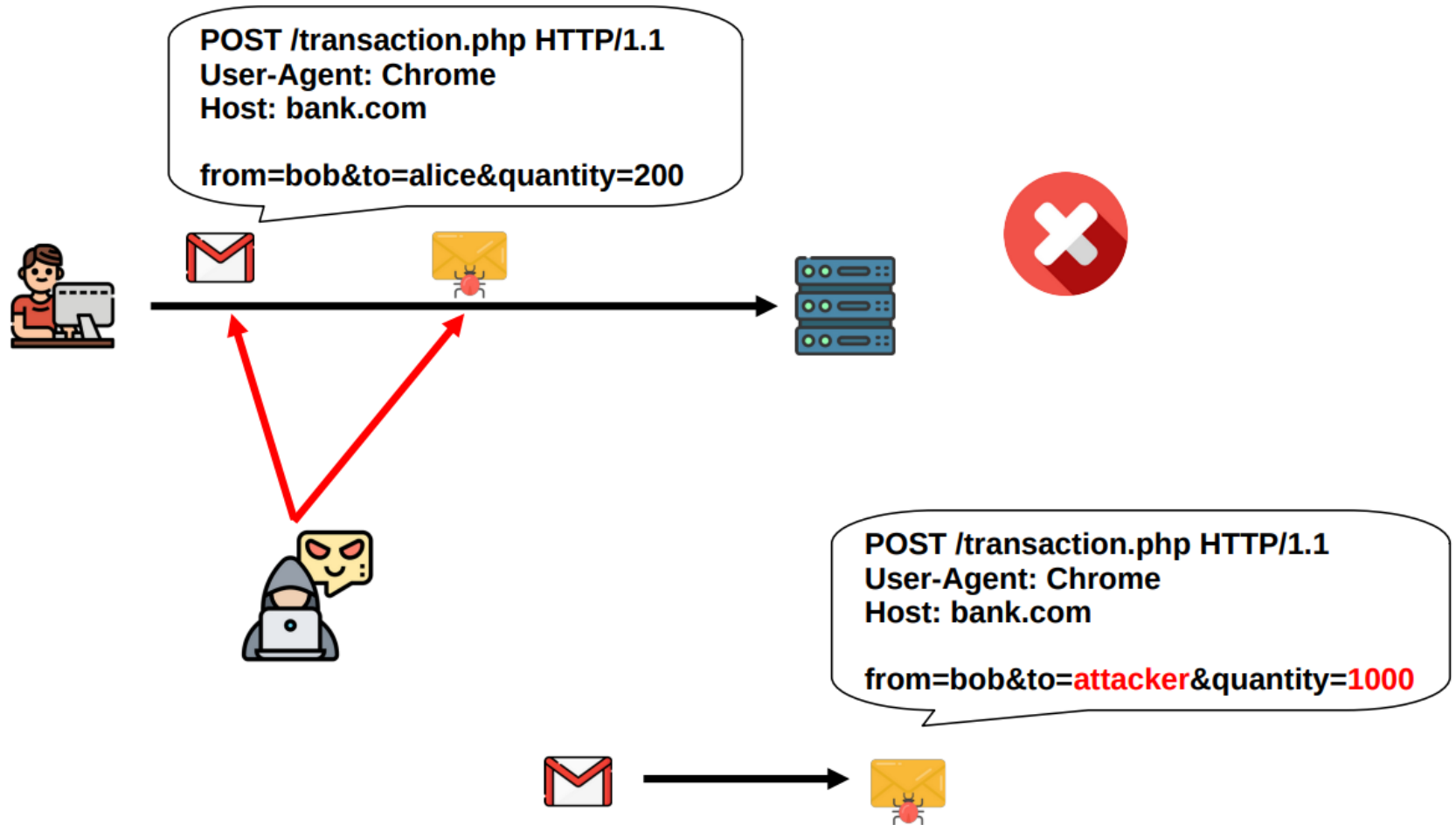
Integrity (1/3)

We have **integrity** when data is protected from unauthorized changes to ensure that it is reliable and correct.

Integrity (2/3)



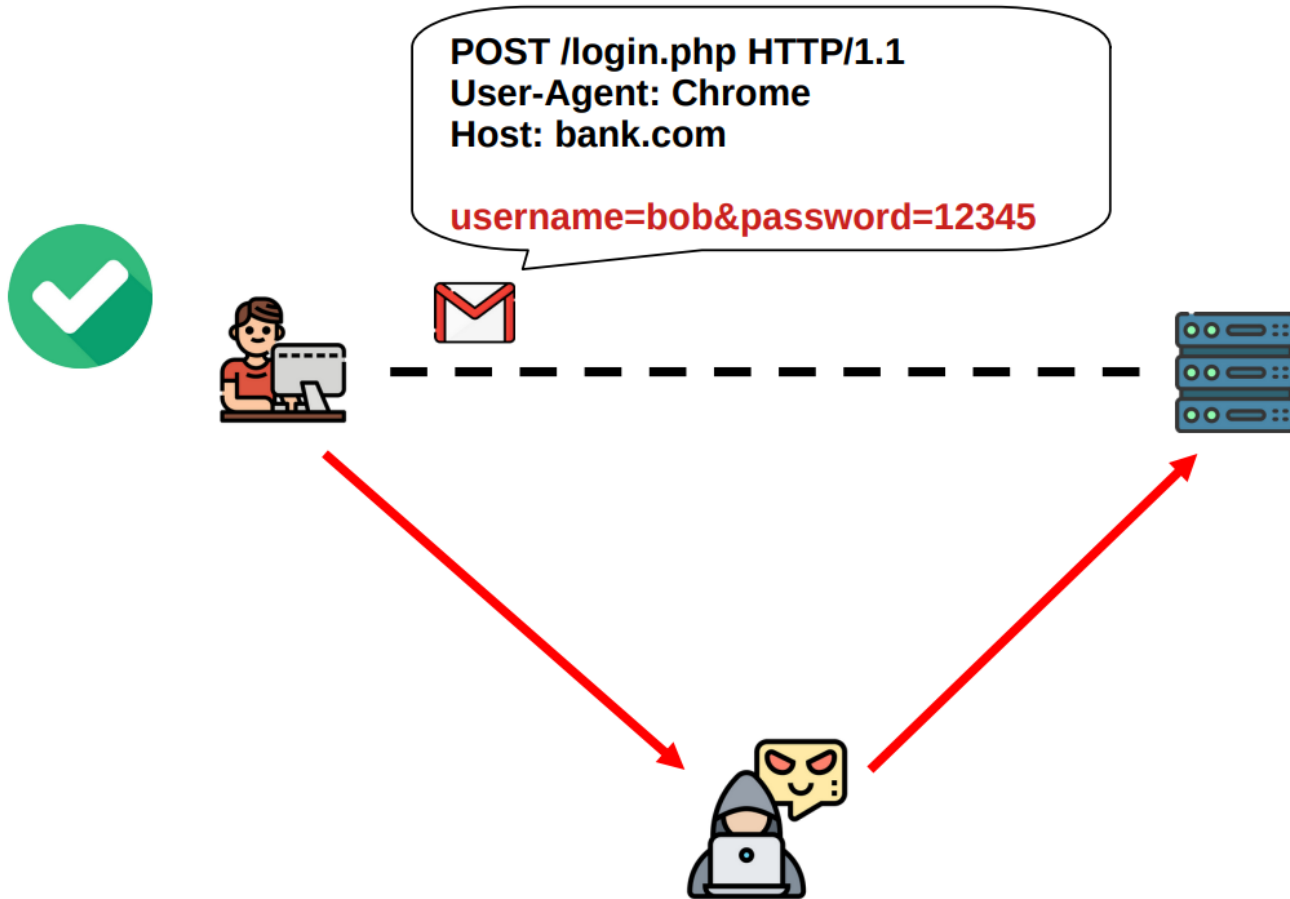
Integrity (3/3)



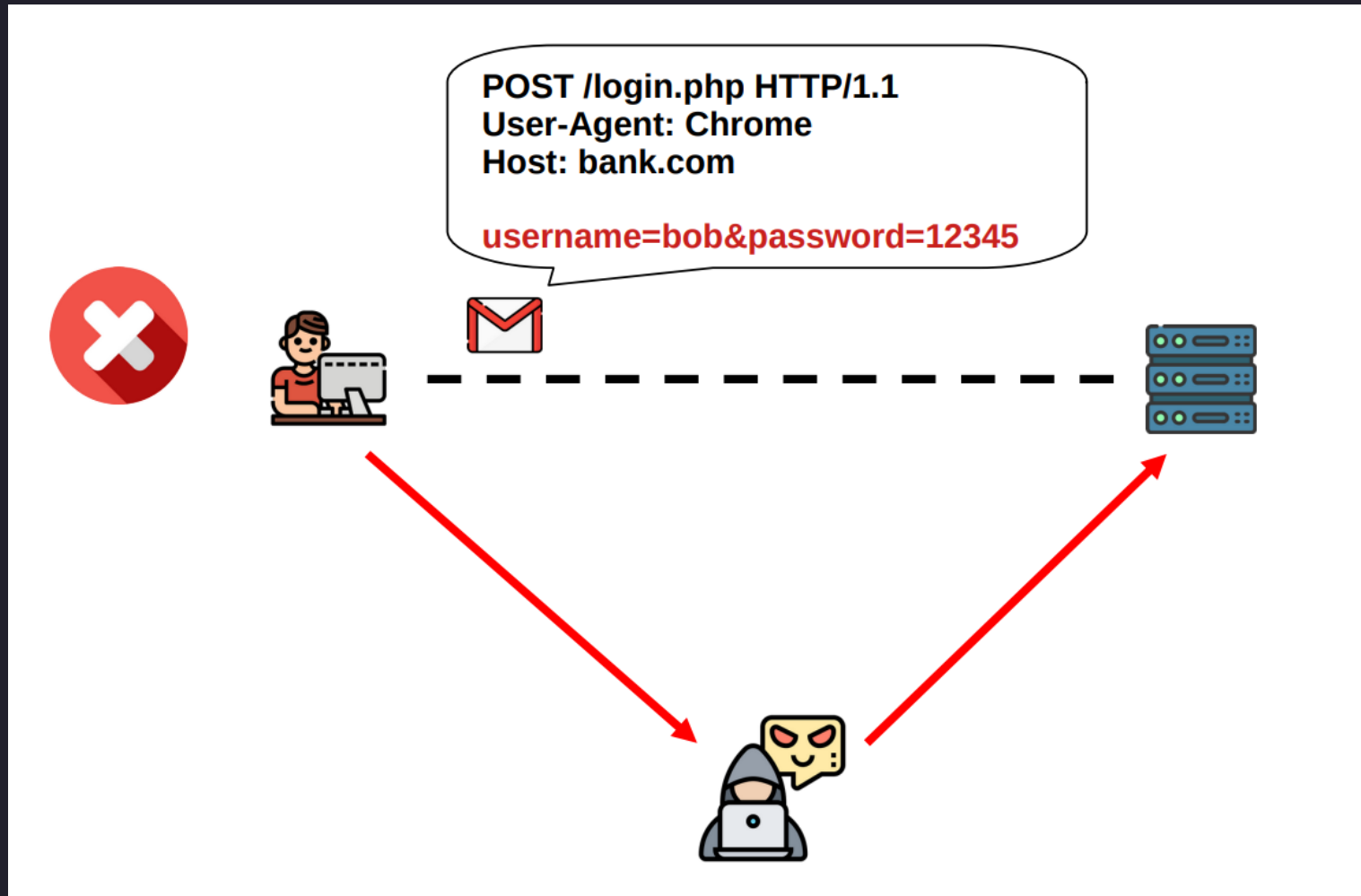
Authentication (1/3)

We have **authentication** when you can verify if the server you're connecting to is a legitimate server or not.

Authentication (2/3)



Authentication (3/3)





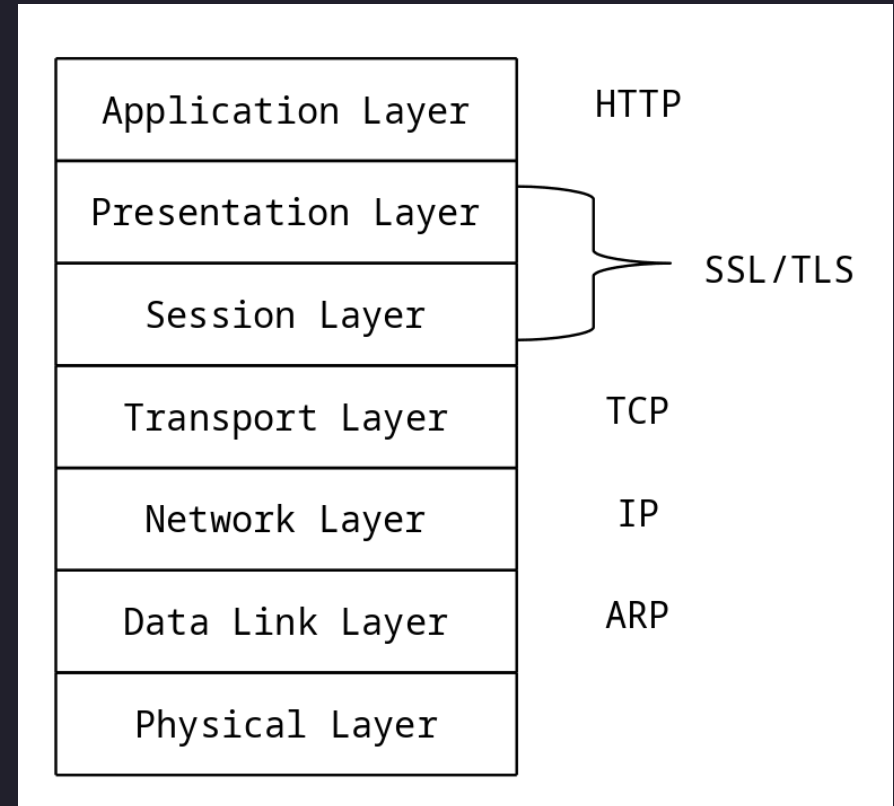
Netscape

To protect digital communications for **e-commerce** purposes, in 1995 the **Netscape** company released

SSL – Secure Socket Layer

SSL is a network protocol designed from its inception to offer cryptographic services such as **confidentiality**, **integrity** and **mutual authentication**

With respect to the standard **ISO/OSI** model, the **SSL** protocol operates between the **session** and **presentation** layers and it is used above a transport layer protocol like **TCP**.



Without SSL

POST /login.php HTTP/1.1
User-Agent: Chrome
Host: bank.com

username=bob&password=12345



TCP Channel



With SSL

*02 db d9 20 16 5a ff 78 4e 83 1a a4
6f 27 a9 d0 cc de b5 5f d6 6b 2c ca
dc 55 57 76 b1 17 f5 1b ce 5d a3 7e
7b de 30 e3 e4 87 a8 fa e9 b5 c7 77
3b 26 e3 1c 7b d5 ba f4 c0 44 0c fa*





+



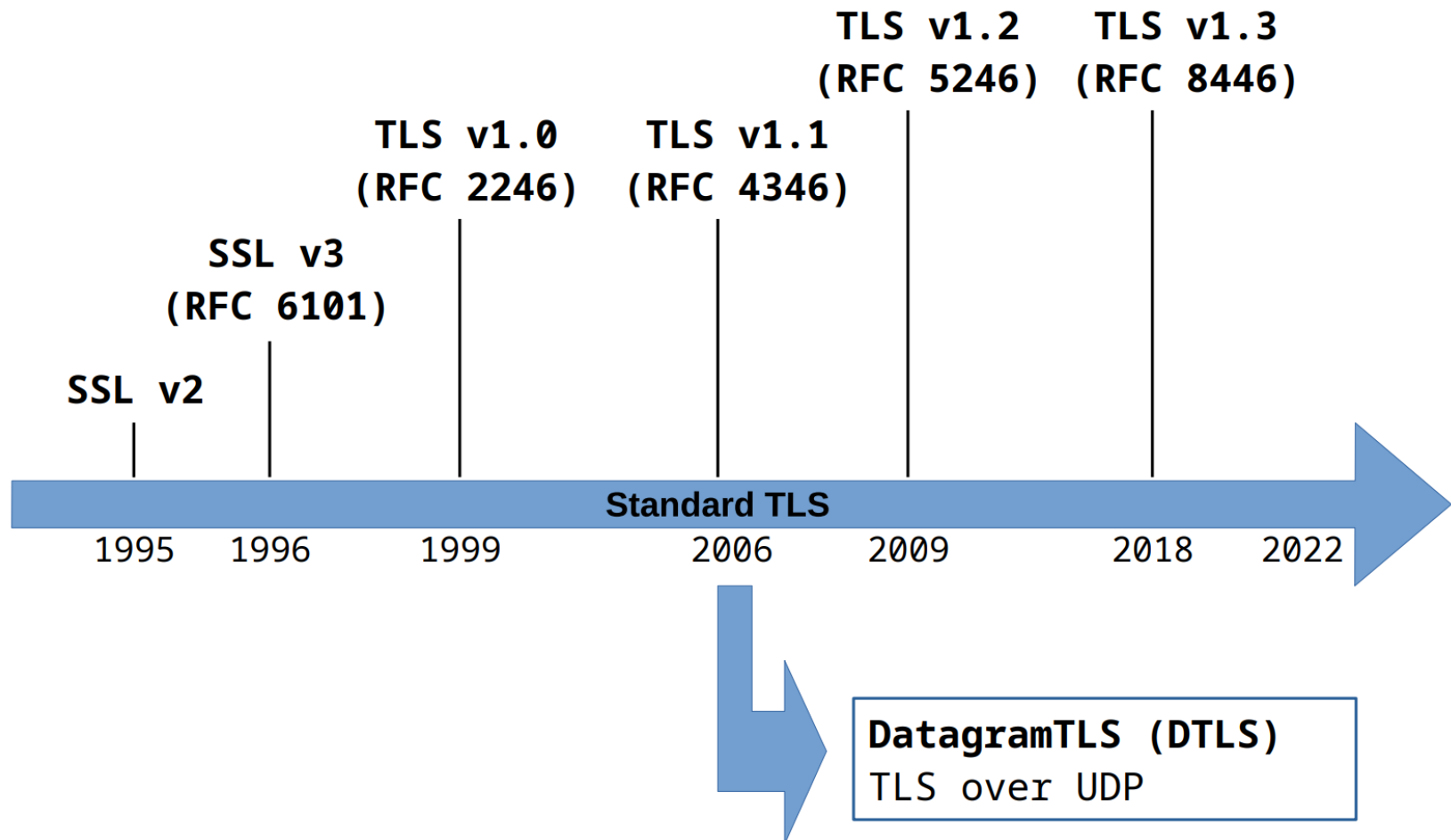
=



In 1999 **SSL** was standardized and its name changed to

TLS – Transport Layer Security

Throughout the years new versions were introduced and then standardized, making the **SSL/TLS** ecosystem of today extremely complex.



SSL/TLS

Main ideas behind TLS:

Main ideas behind TLS:

1. **Asymmetric cryptography** (TLS Handshake)
 - → **authentication**
 - → **key exchange**

Main ideas behind TLS:

1. **Asymmetric cryptography** (TLS Handshake)
 - → **authentication**
 - → **key exchange**
2. **Symmetric cryptography** (TLS Encryption)
 - → **confidentiality**
 - → **integrity**

TLS RECORD LAYER

All messages sent within a TLS session follow the same structure, defined by the **record protocol**.

```
struct {  
  ContentType type;  
  ProtocolVersion version;  
  uint16 length;  
  opaque fragment[TLSPlaintext.length];  
} TLSPlaintext;
```

The standardization defines four different
subprotocols:

- handshake
- change_{cipherspec}
- alert
- application_{data}

TLS HANDSHAKE

In the handshake phase the following things happen

In the handshake phase the following things happen

1. **Exchange of capabilities**

- → **protocol version**
- → **cipher suites**
- → **extensions**

In the handshake phase the following things happen

1. Exchange of capabilities

- → protocol version
- → cipher suites
- → extensions

2. Authentication

- → Public Key Infrastructure (PKI)
- → DSA, RSA

In the handshake phase the following things happen

1. **Exchange of capabilities**

- → **protocol version**
- → **cipher suites**
- → **extensions**

2. **Authentication**

- → **Public Key Infrastructure (PKI)**
- → **DSA, RSA**

3. **Key exchange**

- → **RSA, ECDHE, DHE**

In the handshake phase the following things happen

1. **Exchange of capabilities**

- → **protocol version**
- → **cipher suites**
- → **extensions**

2. **Authentication**

- → **Public Key Infrastructure (PKI)**
- → **DSA, RSA**

3. **Key exchange**

- → **RSA, ECDHE, DHE**

4. **Handshake integrity check**

Main message flows for the handshake

- Full handshake with server authentication
- Full handshake with mutual authentication
- Abbreviated handshake with session resumption

Full handshake with server auth (\leq TLSv1.2)



TLS ENCRYPTION

In TLS **confidentiality** and **integrity** to the application data are granted through the usage of three particular strategies:

In TLS **confidentiality** and **integrity** to the application data are granted through the usage of three particular strategies:

- **stream encryption** (→ RC4)

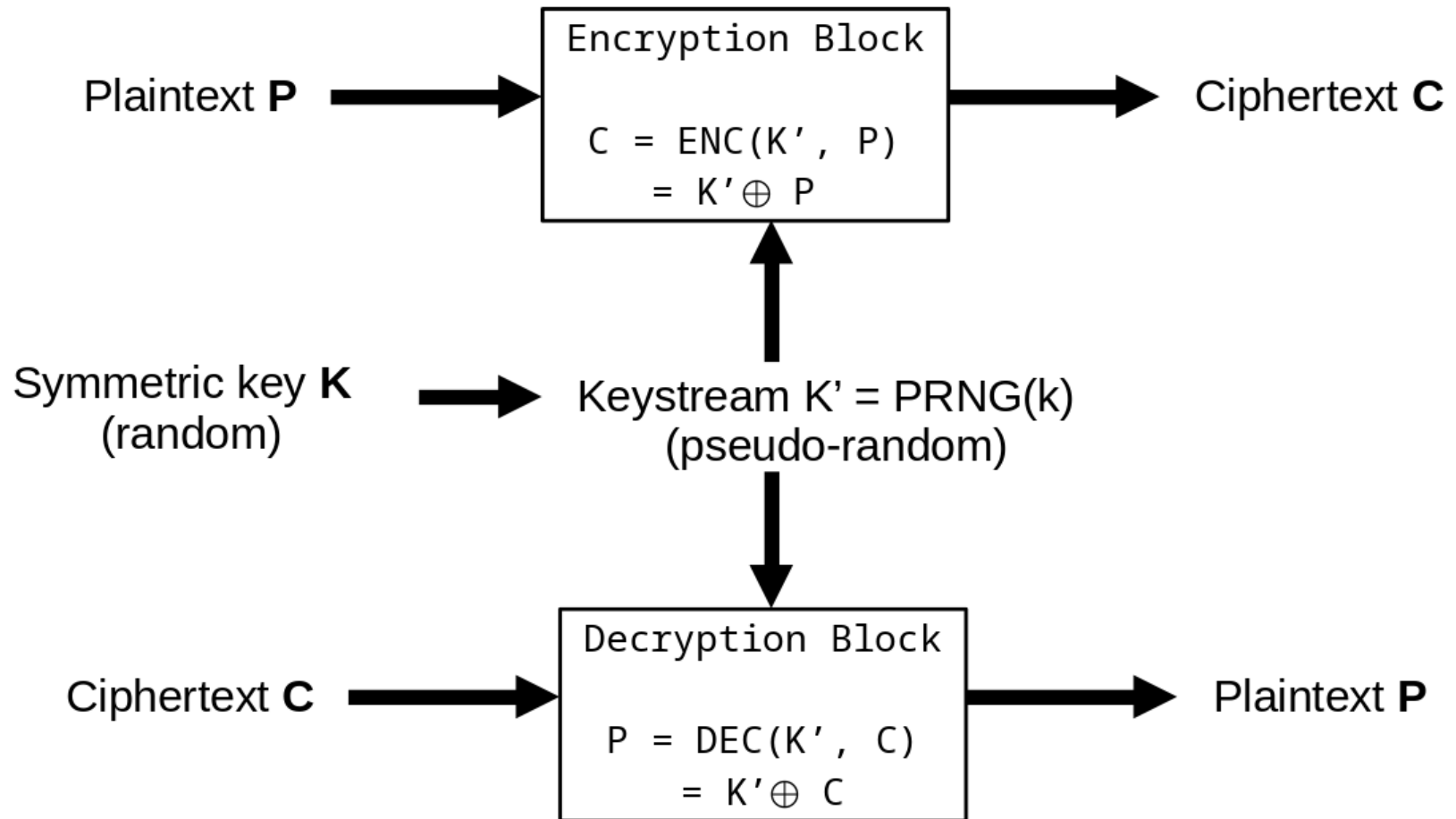
In TLS **confidentiality** and **integrity** to the application data are granted through the usage of three particular strategies:

- **stream encryption** (→ RC4)
- **block encryption** (→ AES)

In TLS **confidentiality** and **integrity** to the application data are granted through the usage of three particular strategies:

- **stream encryption** (→ RC4)
- **block encryption** (→ AES)
- **authenticated encryption** (→ AES-GCM)

Stream Encryption (1/2)



Stream Encryption (2/2)

Used with a **MAC-THEN-ENCRYPT** scheme:

1. **MAC** is computed on:
 - Sequence number (**replay attacks**)
 - TLS header
 - TLS record data
2. **Stream encryption.**

Block Encryption (1/3)

Block ciphers work on blocks of a specified size.

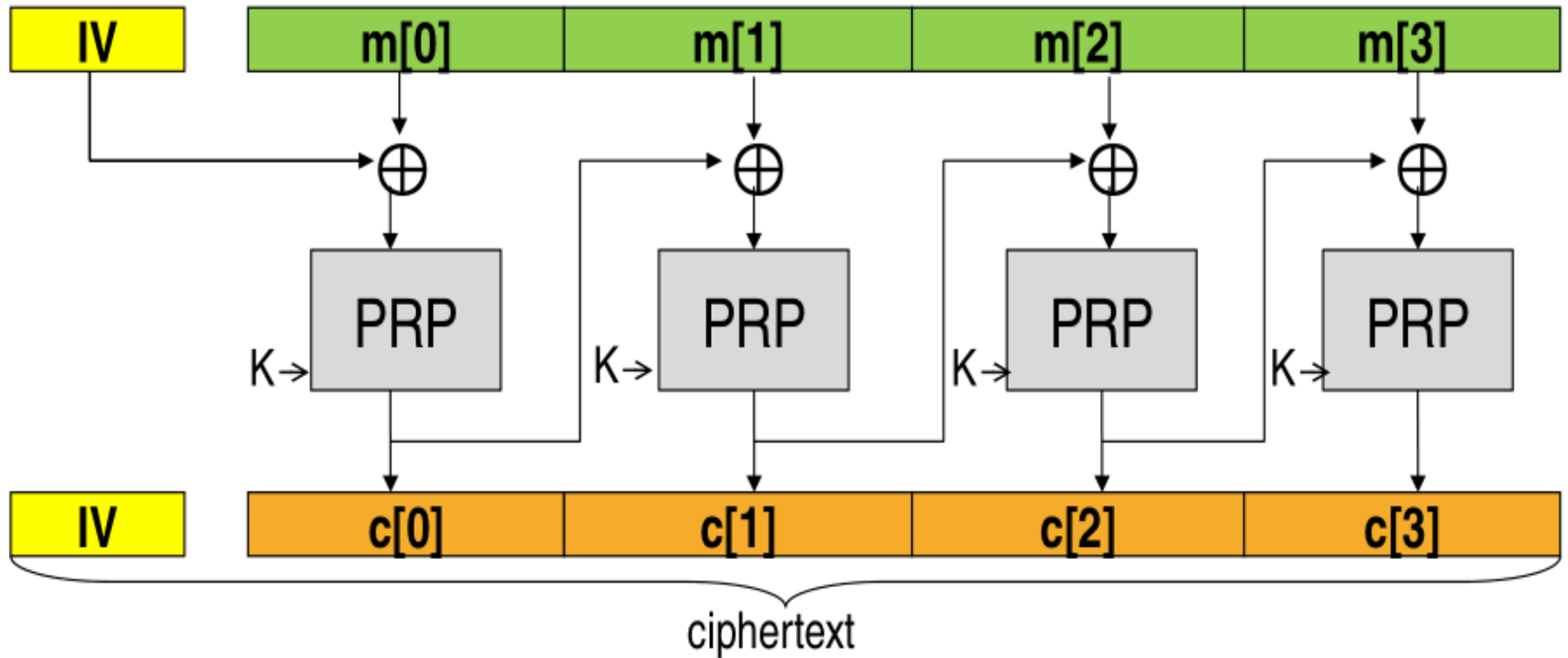
If the plaintext length is not an integer multiple of the block size, **padding** is added.

Block Encryption (2/3)

Used with a **MAC-THEN-ENCRYPT** scheme:

1. **MAC** is computed on:
 - Sequence number (**replay attacks**)
 - TLS header
 - TLS record data
2. **Padding** is added.
3. **Block encryption**.

Block Encryption (3/3)

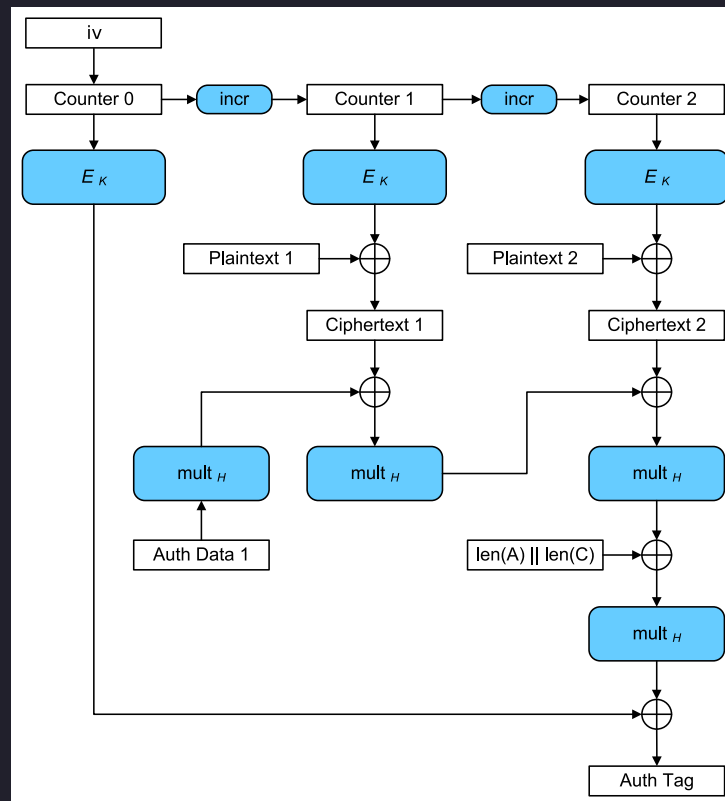


Authentication Encryption (1/2)

Authenticated Encryption schemes grant with a single algorithm protection for both confidentiality and integrity. These schemes do not use **IVs** but they use other special values called **nonces**, which typically must be unique per encryption.

Authentication Encryption (2/2)

AES-GCM is an example of authenticated encryption.



CIPHER SUITES

Cipher suites are identifiers that specify all the algorithms and cryptographic primitives that will be used to protect the confidentiality, authenticity and integrity of the TLS session.

Ephemeral Diffie-Hellman on
Elliptic Curves for key exchange

AES-GCM with 256 bit for
authenticated encryption
(confidentiality + integrity)

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

RSA for authentication

PRF used for the session.

EXTENSIONS

The TLS protocol can be extended through the usage of **TLS extension**, introduced with **RFC 3546**.

Examples of TLS extensions are:

- **Heartbeat** -> Heartbleed (CVE-2014-0160)
- **Session Ticket**
- **Server Name Indication**
- **Named Curve**

EXAMPLE

Let us try to connect to my website at
<https://leonardotamiano.xyz>

First, we start `tcpdump`

```
sudo tcpdump -i eno1 -w tls_example.pcap "port 443"
```

And then we perform a `curl` request

```
curl https://leonardotamiano.xyz
```

From this we can a **pcap** trace that we can analyze with **wireshark**

The image shows a Wireshark packet capture of a TLS handshake. The interface includes a menu bar, a toolbar, a filter bar, and a packet list pane. The packet list pane shows 23 packets, with packet 7 selected. The packet details pane shows the structure of the selected packet, and the packet bytes pane shows the raw data.

No.	Time	Protocol	Length	Info
1	0.000000	TCP	74	51760 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3343947812 TSecr=610452610
2	0.035637	TCP	74	443 → 51760 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=610452610 TSecr=3343947812
3	0.035687	TCP	66	51760 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3343947847 TSecr=610452610
4	0.039214	TLSv1.2	583	Client Hello
5	0.074645	TCP	66	443 → 51760 [ACK] Seq=1 Ack=518 Win=64768 Len=0 TSval=610452649 TSecr=3343947851
6	0.076928	TLSv1.2	1514	Server Hello
7	0.076954	TCP	66	51760 → 443 [ACK] Seq=518 Ack=1449 Win=64128 Len=0 TSval=3343947889 TSecr=610452651
8	0.077179	TCP	1514	443 → 51760 [ACK] Seq=1449 Ack=518 Win=64768 Len=1448 TSval=610452651 TSecr=3343947889
9	0.077198	TCP	66	51760 → 443 [ACK] Seq=518 Ack=2897 Win=63488 Len=0 TSval=3343947889 TSecr=610452651
10	0.077427	TLSv1.2	1266	Certificate
11	0.077427	TLSv1.2	379	Server Key Exchange, Server Hello Done
12	0.077446	TCP	66	51760 → 443 [ACK] Seq=518 Ack=4097 Win=62592 Len=0 TSval=3343947889 TSecr=610452651
13	0.077456	TCP	66	51760 → 443 [ACK] Seq=518 Ack=4410 Win=62336 Len=0 TSval=3343947889 TSecr=610452651
14	0.079375	TLSv1.2	151	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
15	0.114878	TLSv1.2	109	Change Cipher Spec, Encrypted Handshake Message
16	0.115087	TLSv1.2	169	Application Data
17	0.151426	TLSv1.2	2077	Application Data
18	0.151506	TCP	66	51760 → 443 [ACK] Seq=706 Ack=6464 Win=63488 Len=0 TSval=3343947963 TSecr=610452720
19	0.151911	TLSv1.2	89	Encrypted Alert
20	0.153588	TCP	66	51760 → 443 [FIN, ACK] Seq=729 Ack=6464 Win=64128 Len=0 TSval=3343947965 TSecr=610452720
21	0.187426	TCP	66	443 → 51760 [FIN, ACK] Seq=6464 Ack=729 Win=64768 Len=0 TSval=610452762 TSecr=3343947965
22	0.187473	TCP	66	51760 → 443 [ACK] Seq=730 Ack=6465 Win=64128 Len=0 TSval=3343947999 TSecr=610452762
23	0.189133	TCP	66	443 → 51760 [ACK] Seq=6465 Ack=730 Win=64768 Len=0 TSval=610452764 TSecr=3343947965

Frame 7: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
Ethernet II, Src: ASUSTekC_d5:a3:bc (14:da:e9:d5:a3:bc), Dst: AethraTe_5_ (14:da:e9:d5:a3:bc)

Pacchetti: 23 · visualizzati: 23 (100.0%)

From the pcap we see the message trace:

- First a TCP channel is established with typical **3-way TCP handshake**
- Then **TLS handshake** is performed
- Finally application data is sent encrypted

ATTACKS TO TLS

Even though TLS is designed to offer security, throughout the years various bugs and design mistakes have been discovered within the **TLS ecosystem**



Classes of bugs (just a few...)

Classes of bugs (just a few...)

- **Implementation bugs**
 - → Heartbleed
 - → Early CCS

Classes of bugs (just a few...)

- **Implementation bugs**
 - → Heartbleed
 - → Early CCS
- **Procol design bugs**
 - → Insecure renegotiation

Classes of bugs (just a few...)

- **Implementation bugs**
 - → Heartbleed
 - → Early CCS
- **Procol design bugs**
 - → Insecure renegotiation
- **Cryptographic bugs**
 - → Bleichenbacher's Oracle
 - → BEAST Attack
 - → CBC Padding Oracle

