TRANSPORT LAYER SECURITY

A Brief Introduction…

TABLE OF CONTENTS

THE CIA TRIAD

The first wave of **network protocols**

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

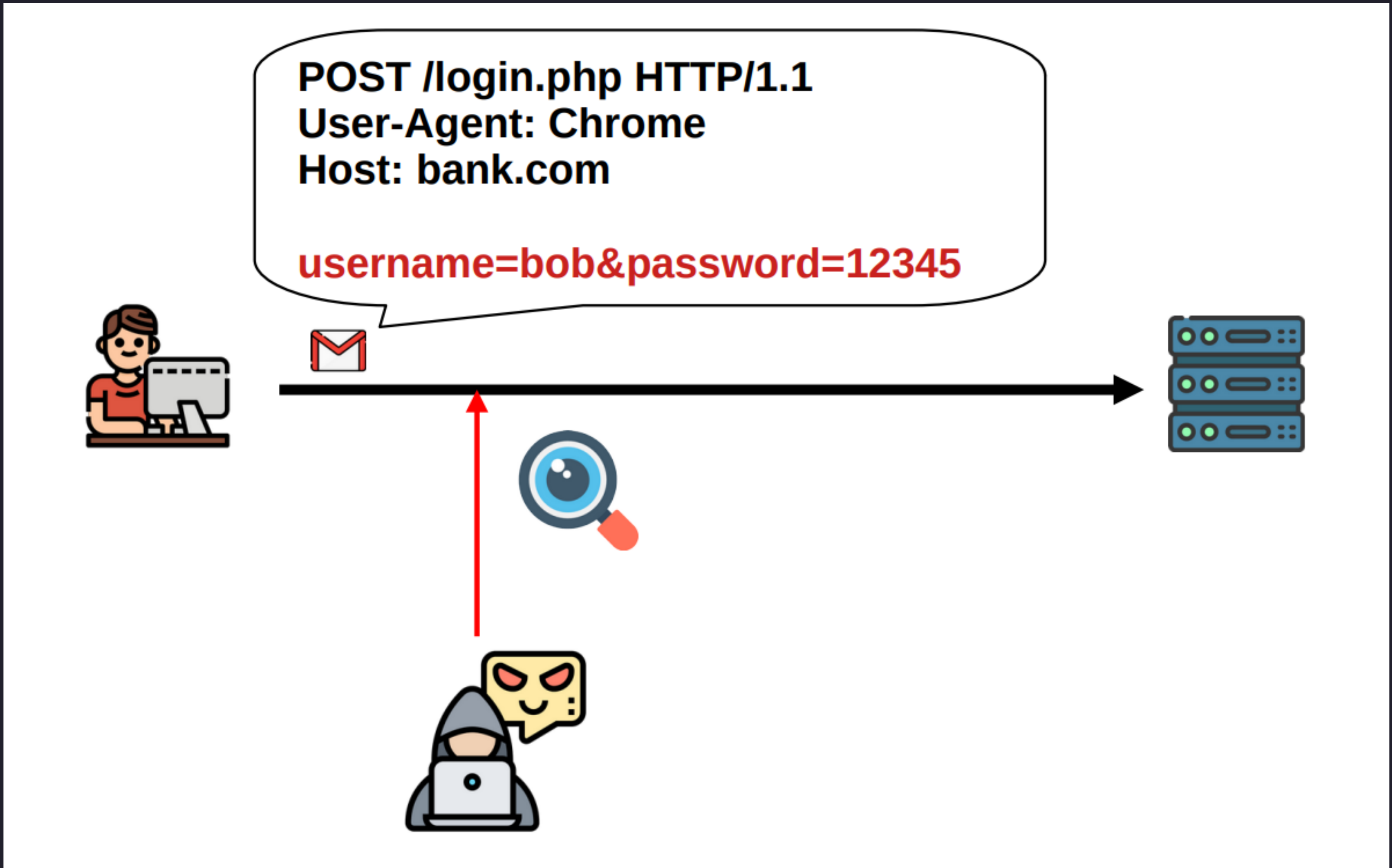
did not offer any **cryptographic services** such as

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

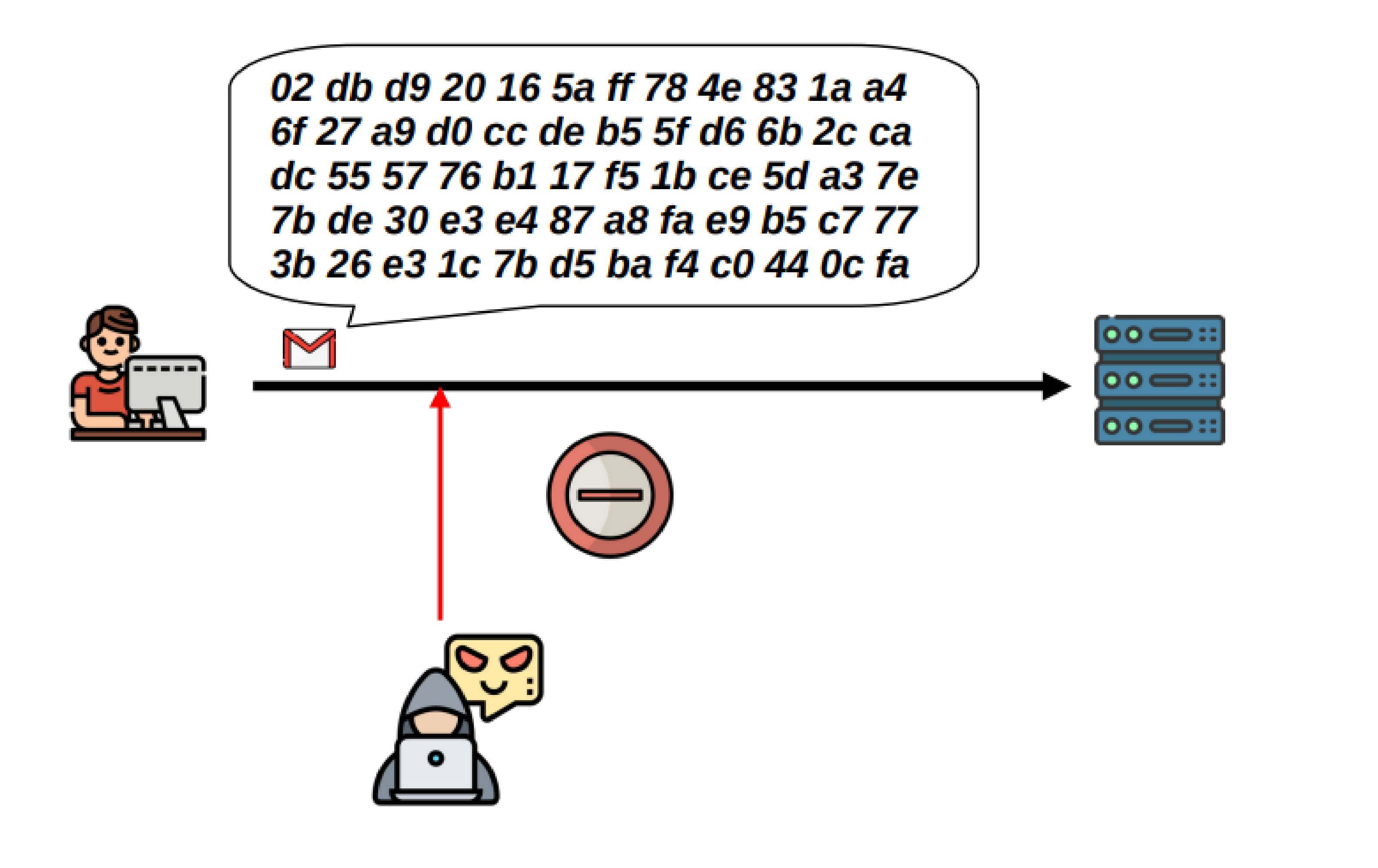
Confidentiality (1/3)

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

Confidentiality (2/3)



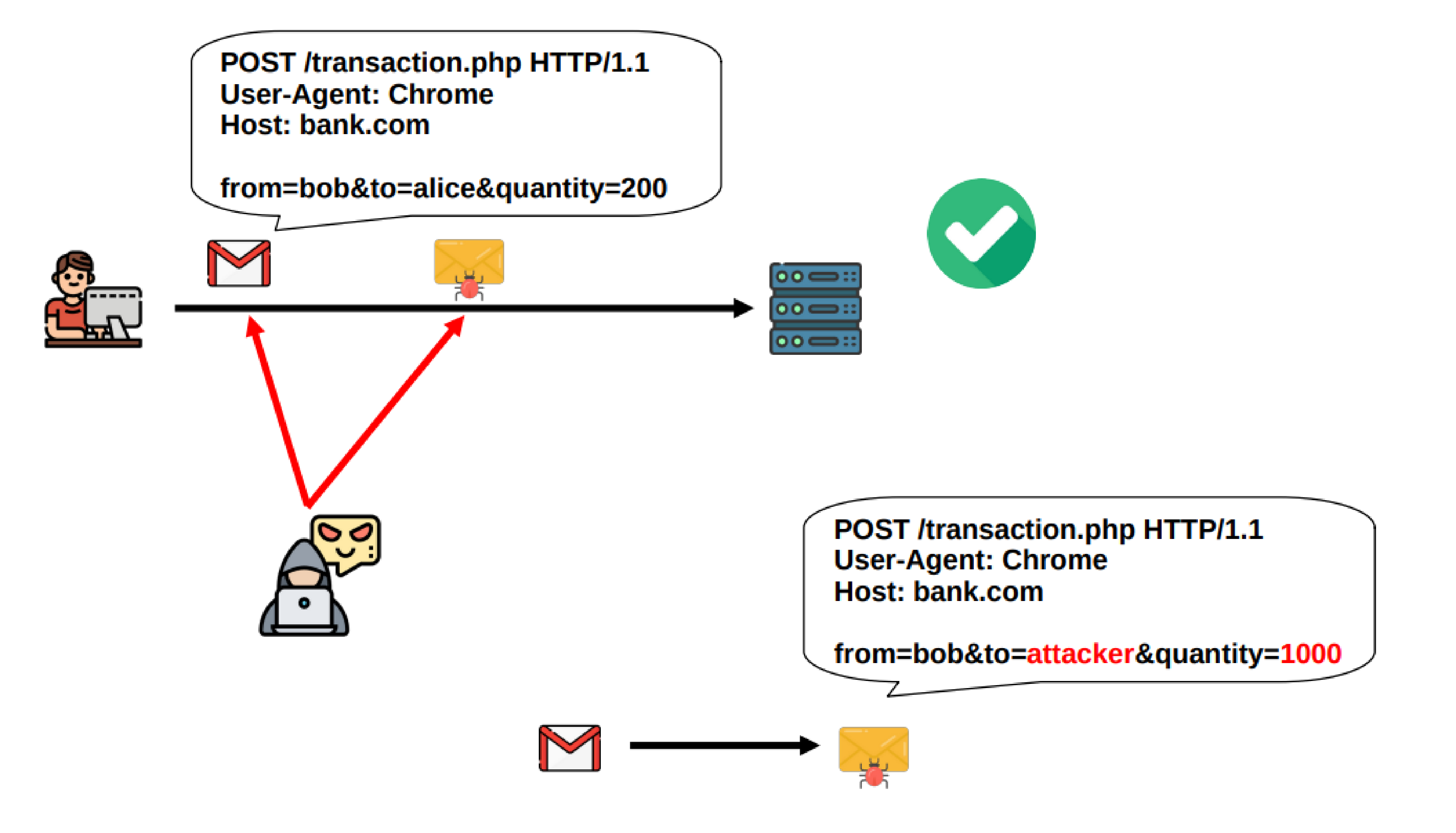
Confidentiality (3/3)



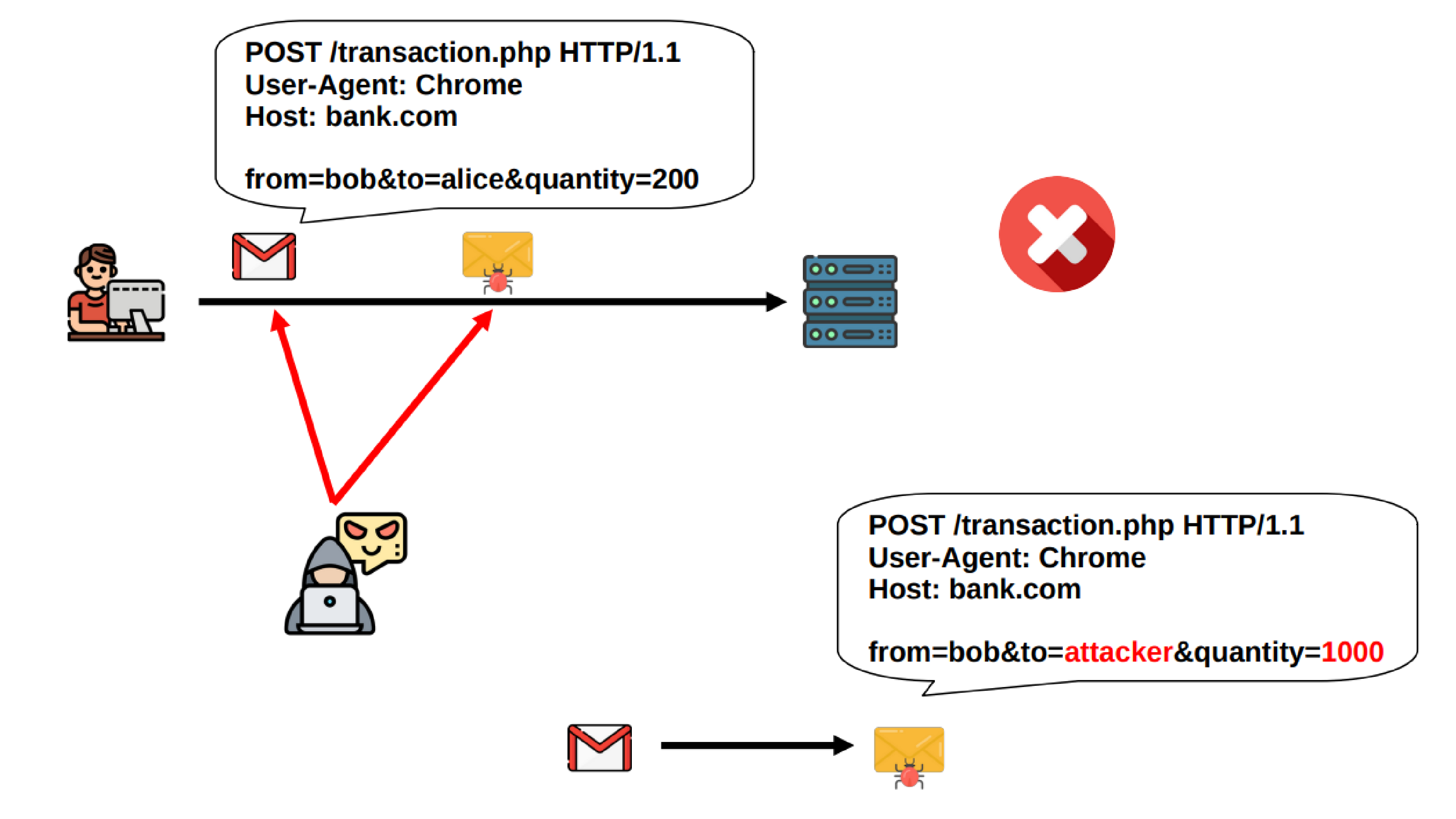
Integrity (1/3)

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

Integrity (2/3)



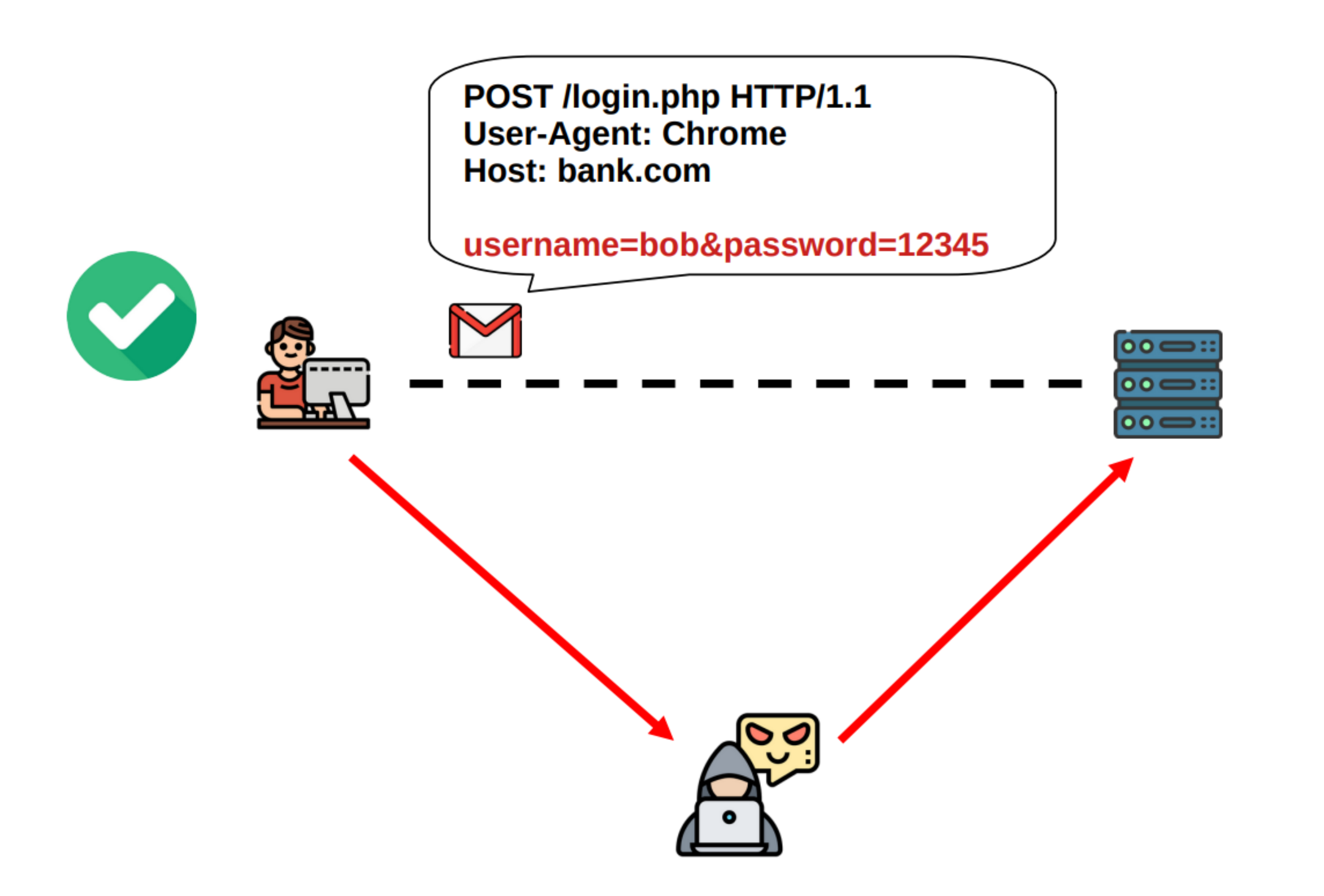
Integrity (3/3)



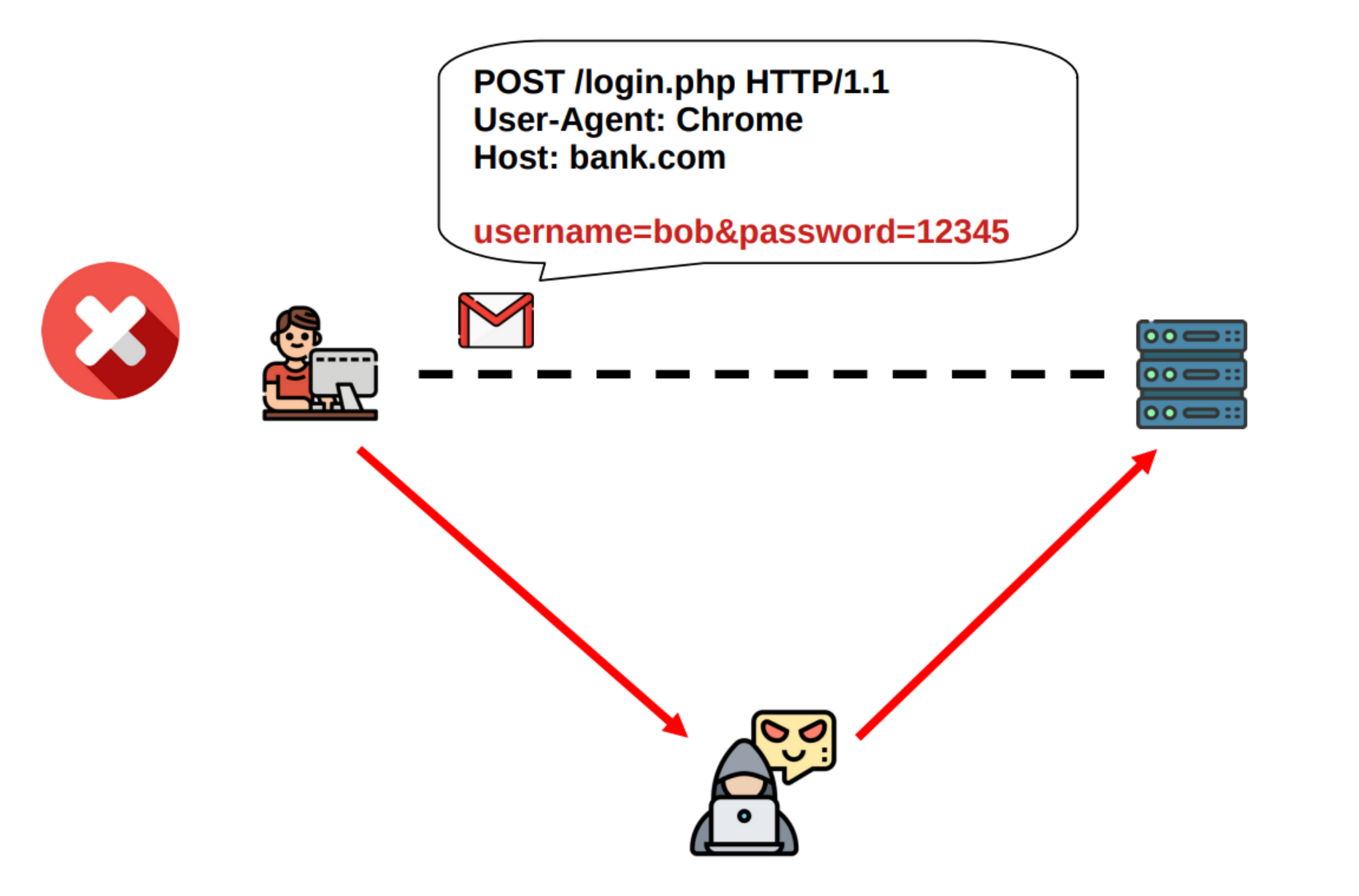
Authentication (1/3)

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)



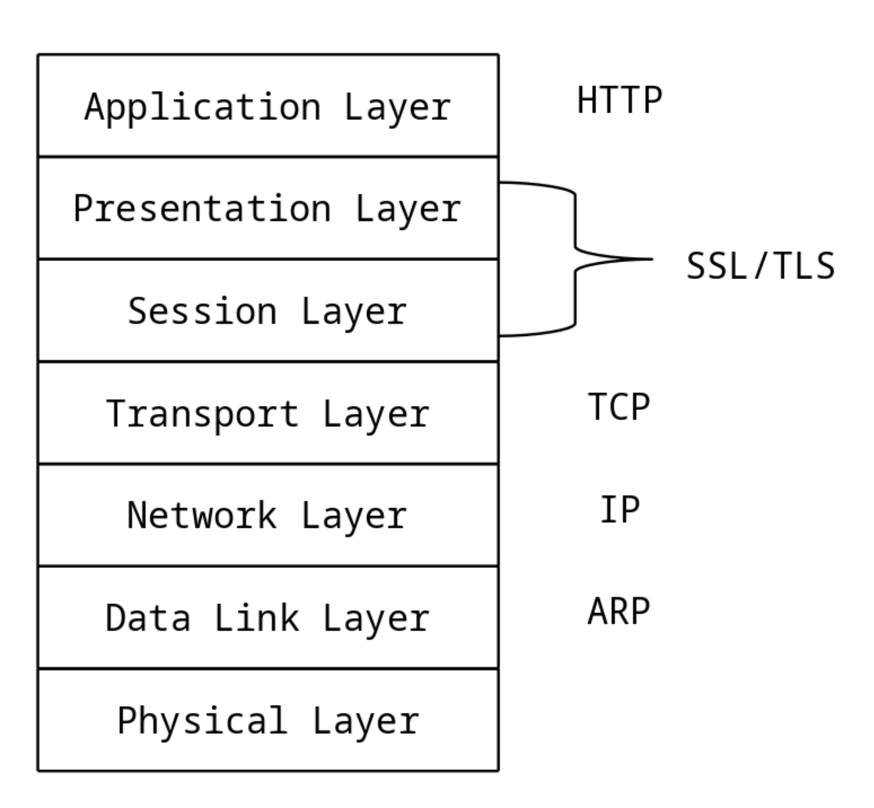
Netscape

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

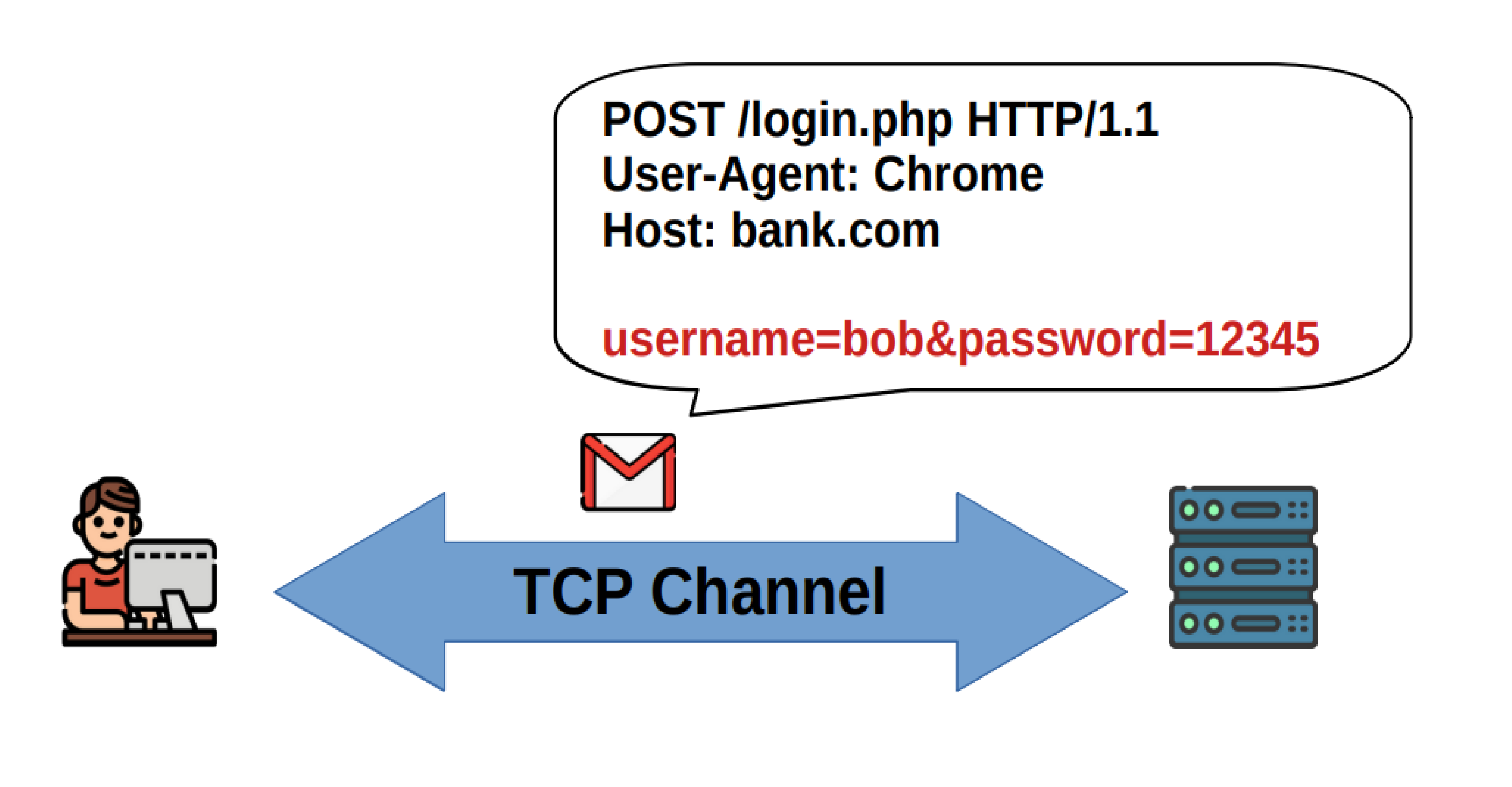
SSL – Secure Socket Layer

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**.

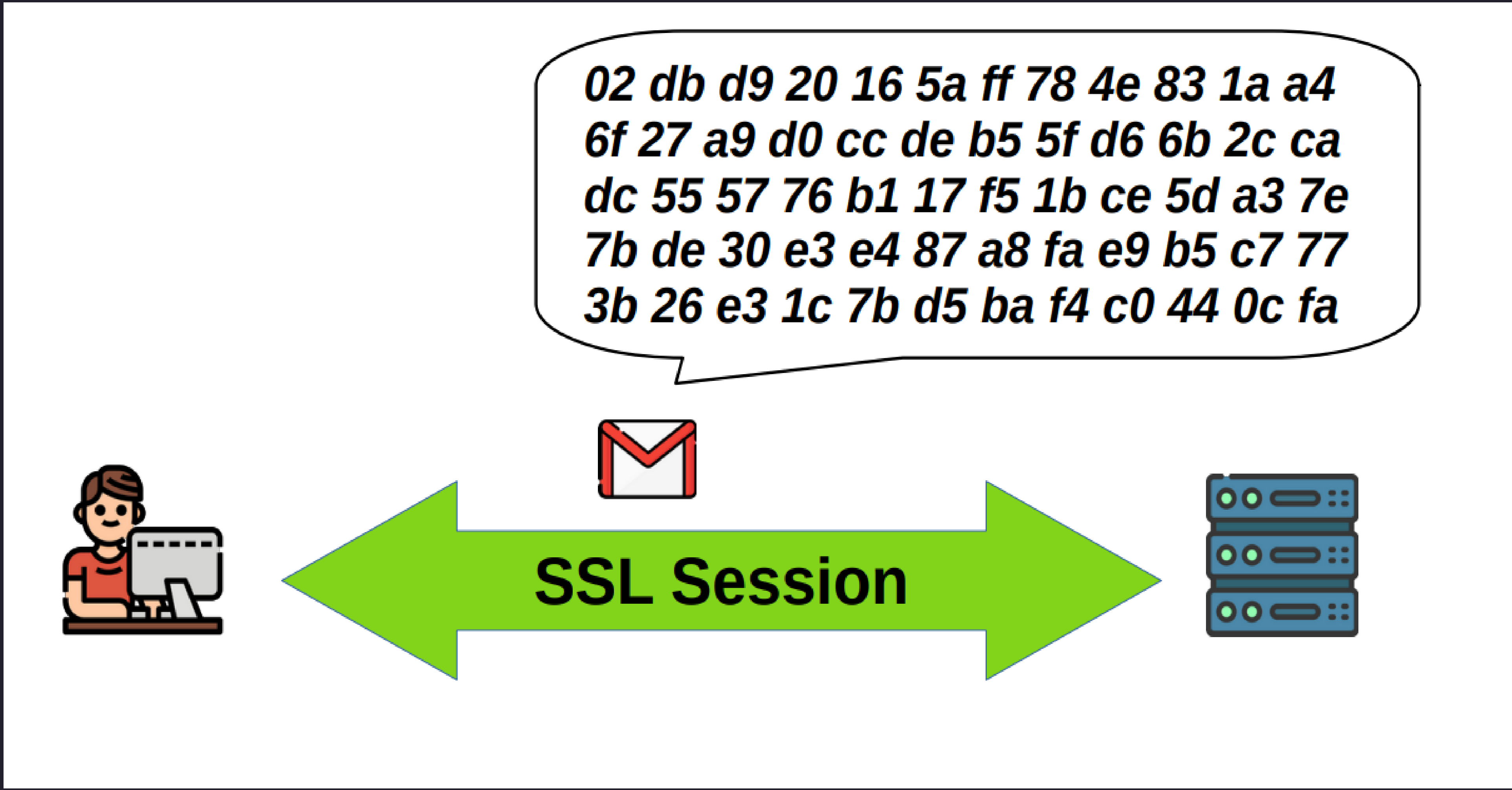


Without SSL



POST /login.php HTTP/1.1
User-Agent: Chrome
Host: bank.com
username=bob&password=12345
TCP Channel

With SSL

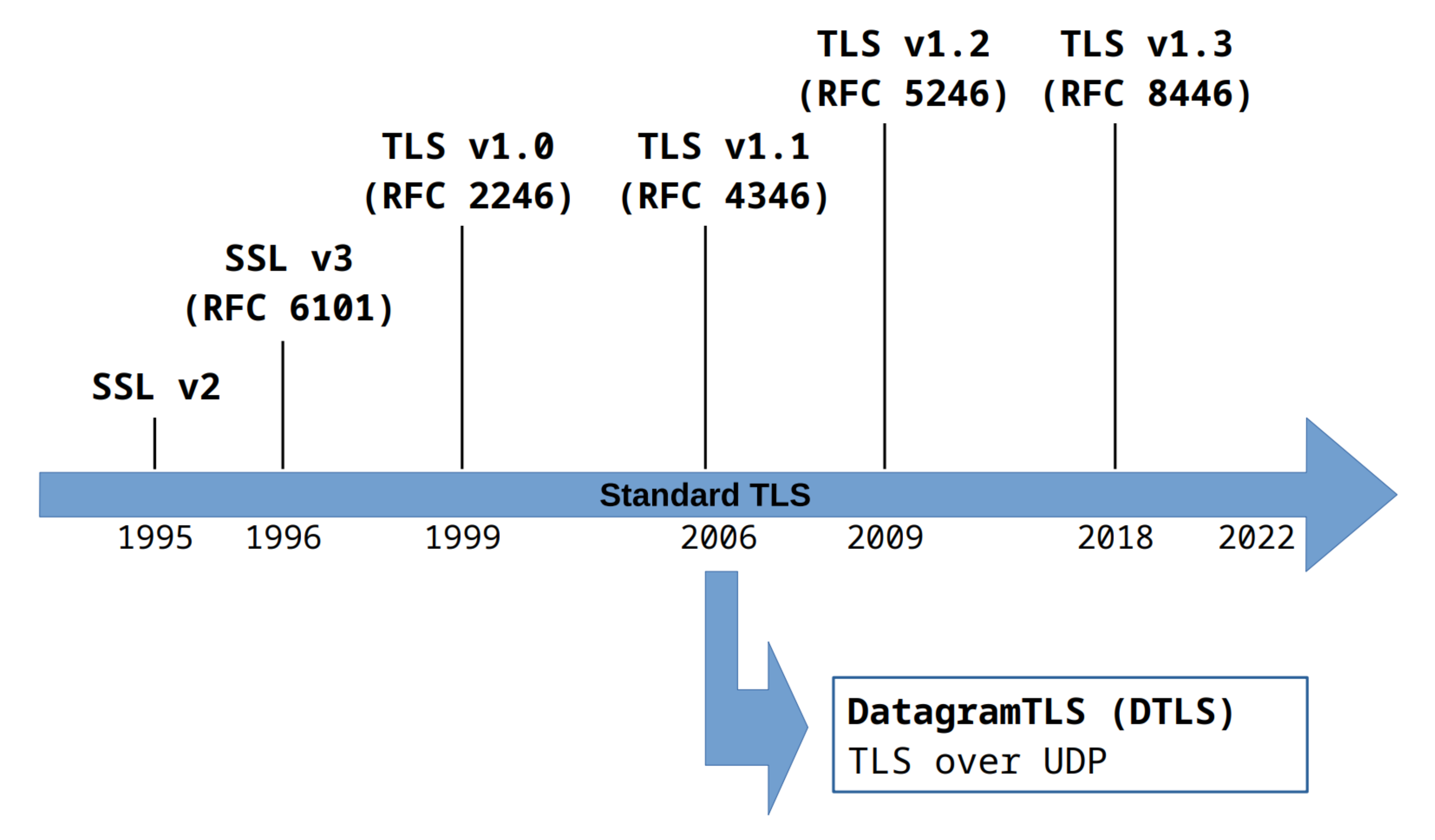

02 db d9 20 16 5a ff 78 4e 83 1a a4
6f 27 a9 d0 cc de b5 5f d6 6b 2c ca
dc 55 57 76 b1 17 f5 1b ce 5d a3 7e
7b de 30 e3 e4 87 a8 fa e9 b5 c7 77
3b 26 e3 1c 7b d5 ba f4 c0 44 0c fa
SSL Session


HTTP
+
SSL
=
HTTPS

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.



TLS v1.2
(RFC 5246)
TLS v1.3
(RFC 8446)
TLS v1.0
(RFC 2246)
TLS v1.1
(RFC 4346)
SSL v3
(RFC 6101)
SSL v2
Standard TLS
1995
1996
1999
2006
2009
2018
2022
DatagramTLS (DTLS)
TLS over UDP

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)

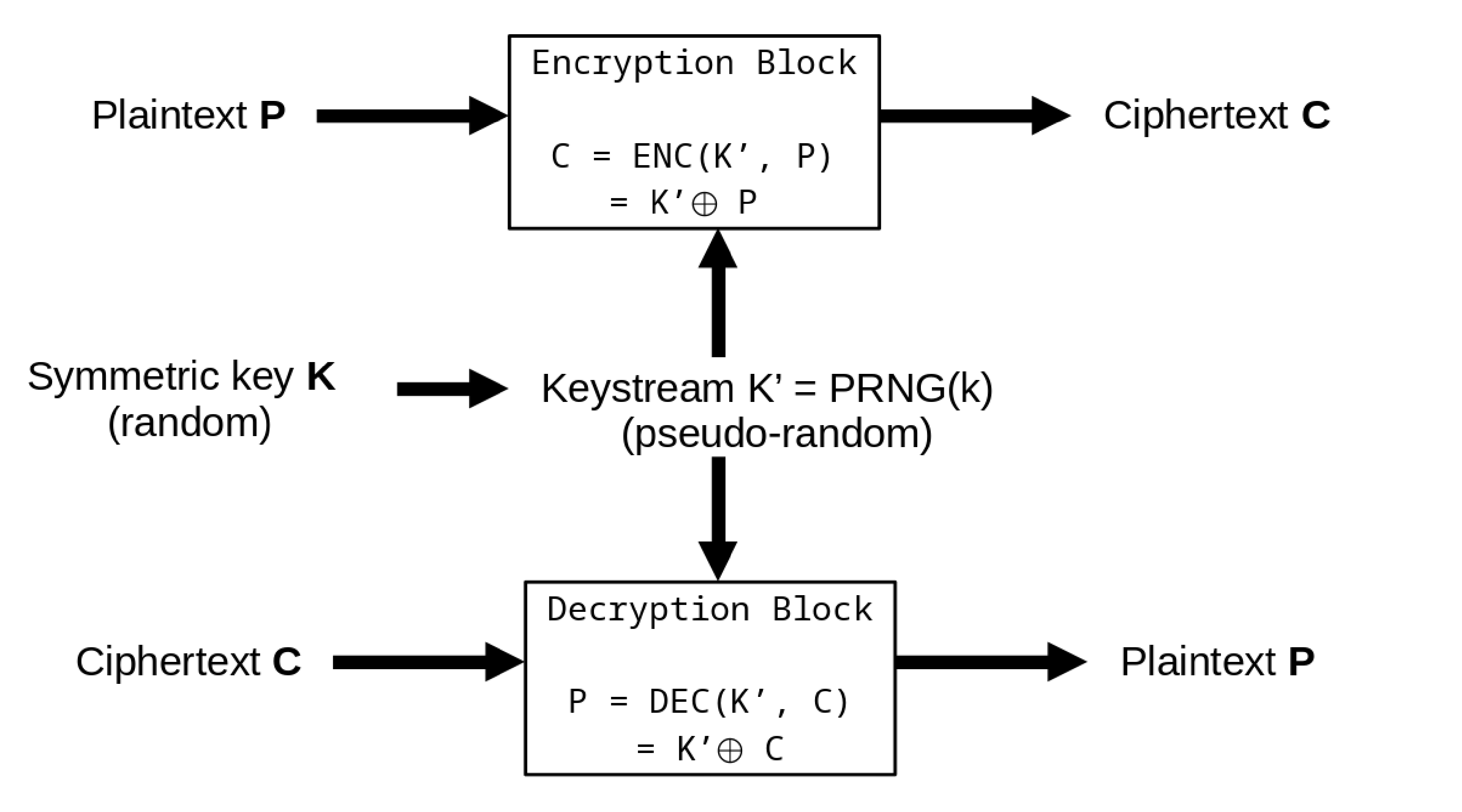
In TLS **confidentiality** and **integrity** to the application data are granted through the usage of three particular strategies:

- **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.

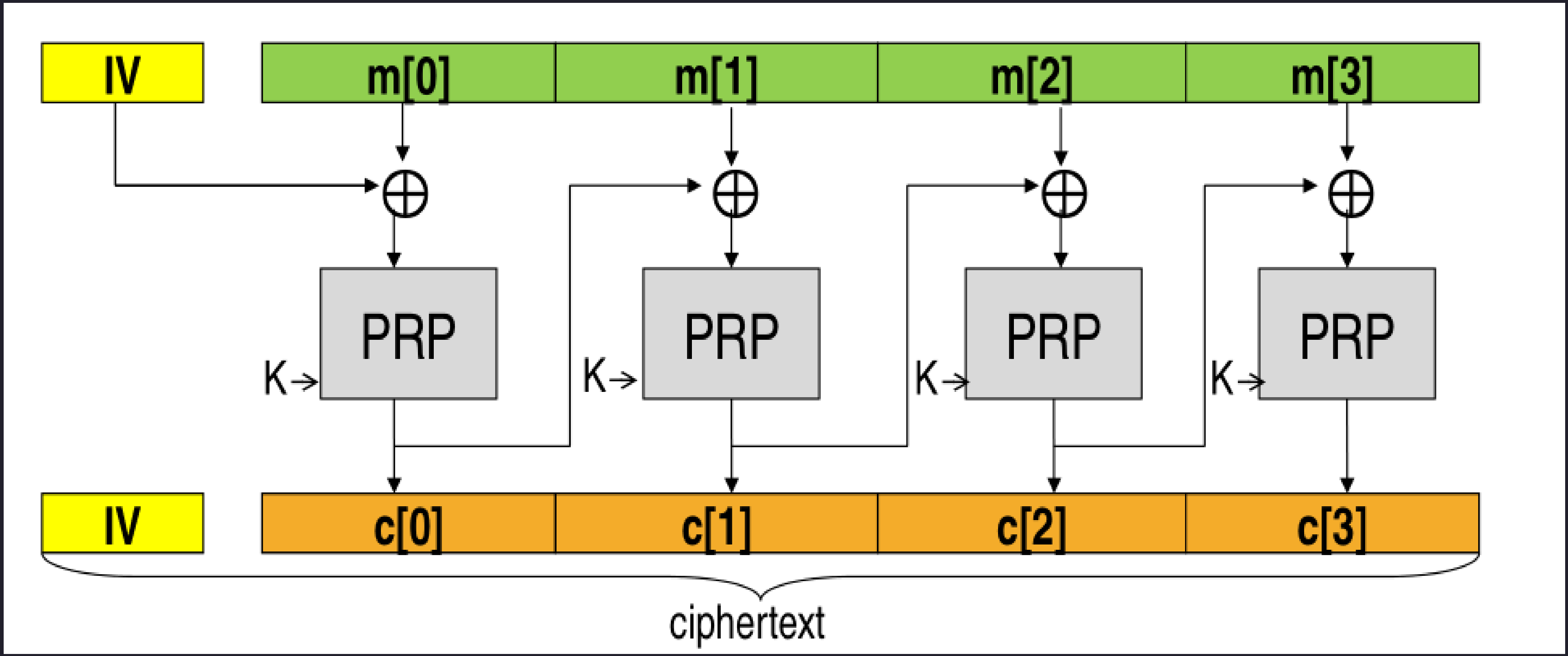
If the plaintext length is not an integer multiple of the block size, **padding** is added.

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)

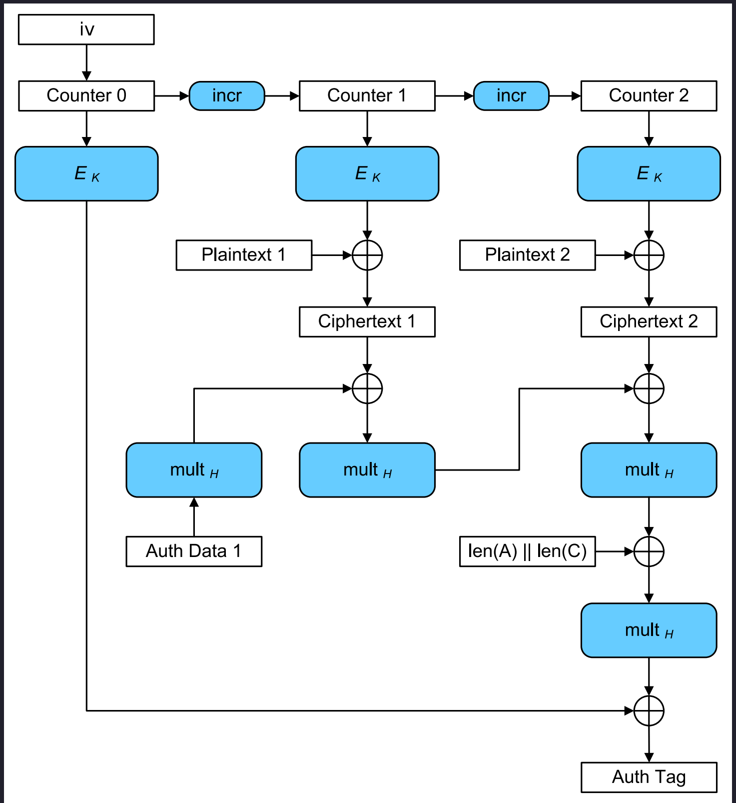


Authentication Encryption (1/2)

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.

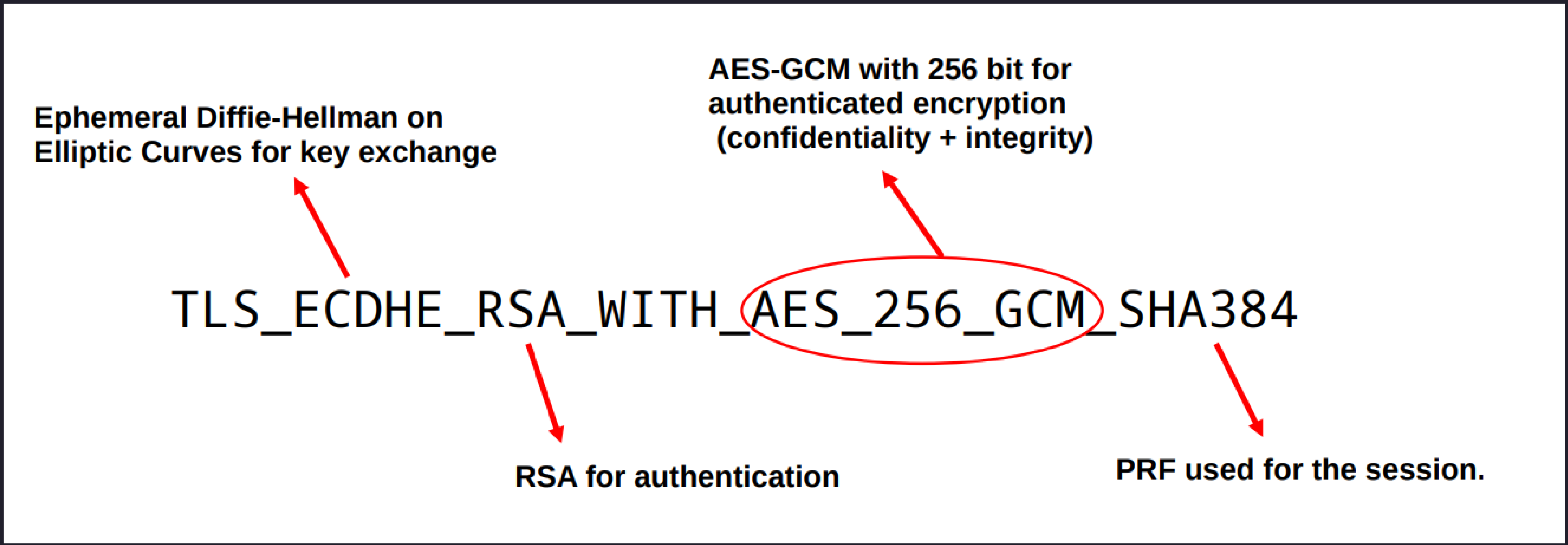
Authentication Encryption (2/2)

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`

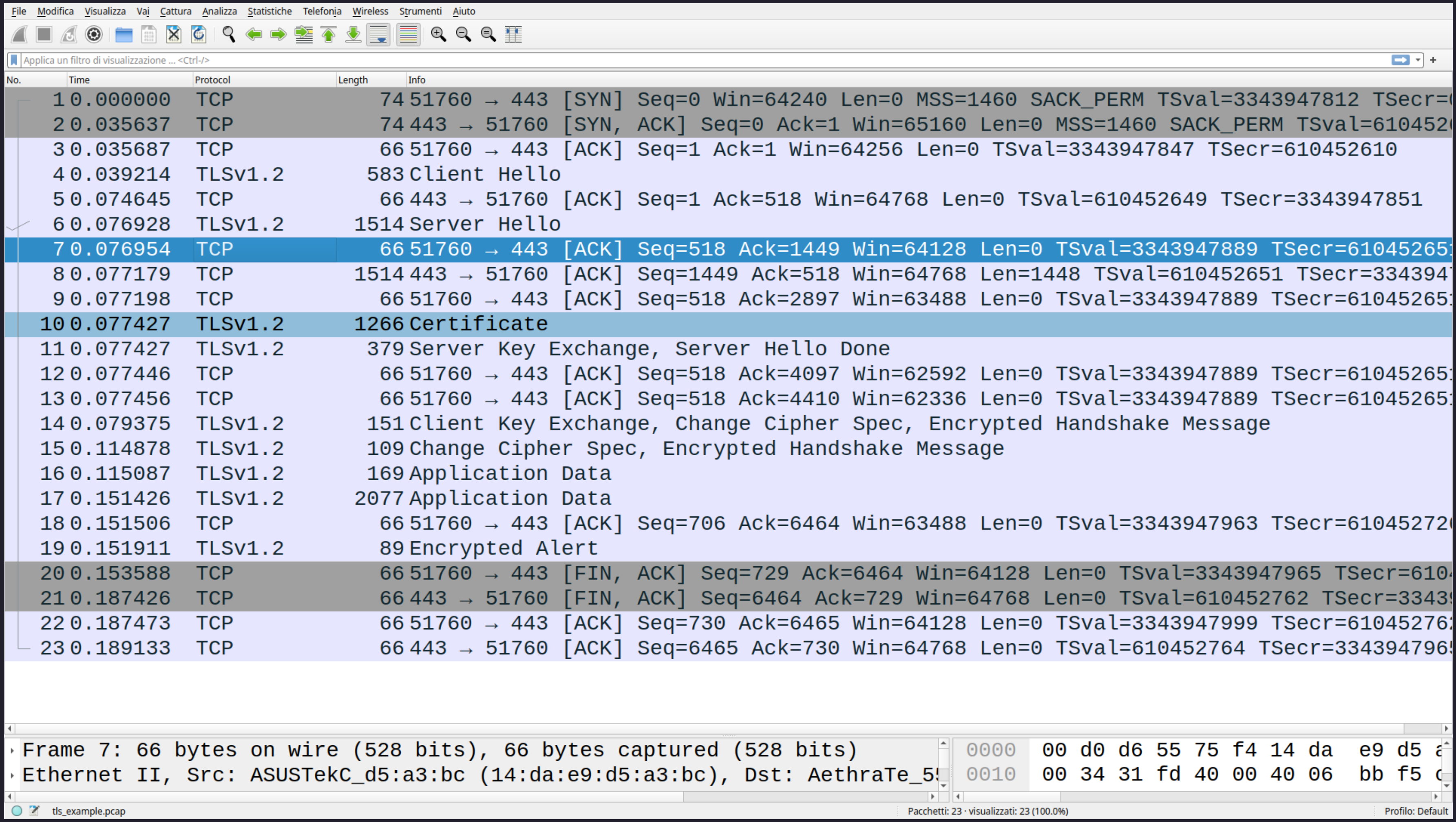
First, we start `tcpdump`

```
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**



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