

# CRITTOGRAFIA

Una (leggera) introduzione

# TABLE OF CONTENTS

- Introduzione
- Perché la Crittografia?
- Crittografia Classica
- Crittografia Moderna
- Conclusioni

# INTRODUZIONE

Sono **Leonardo Tamiano** e attualmente sono un ricercatore per il **CNIT** in tematiche di cybersecurity e applied cryptography.

Nel tempo libero porto avanti un progetto di  
condivisione di conoscenze tecniche informatiche  
tramite la piattaforma youtube.

## Qualche link utile:

- Youtube: <https://www.youtube.com/@LT123>
- Github: <https://github.com/LeonardoE95>
- Blog: <https://blog.leonardotamiano.xyz/>

Queste stesse slides si trovano al seguente URL

**PERCHÉ LA CRITTOGRAFIA?**

































Le società umane, nel corso degli anni, hanno sviluppato **sistemi informativi** sempre più complessi.

I **numeri** ad esempio sono stati introdotti intorno a 6.000 anni fa all'interno della civiltà dei **Sumeri**.

Il loro obiettivo?

**La burocrazia.** Tener traccia delle quantità dei vari oggetti di interesse (cibo, persone, armi, etc...)

1		11		100	
2		12		200	
3		20		300	
4		30		400	
5		40		500	
6		50		600	
7		60		700	
8		70		800	
9		80		900	
10		90		1000	

In alcuni contesti avere accesso a determinate informazioni può essere la differenza tra la vita e la morte.

# PROCESSO DI MARIA STUARDA

## Processo di Maria Stuarda (1/10)

---

Mercoledì 15 ottobre 1586.

Castello di Fotheringhay, Inghilterra centrale.

Maria Stuarda, nota come **la Regina degli Scozzesi**, è sotto processo per tradimento nei confronti della regina **Elisabetta I**.

## Processo di Maria Stuarda (2/10)

---

**Sir Francis Walsingham**, segretario di Stato, cerca prove schiaccianti contro di lei, in quanto consapevole che Elisabetta non firmerà la condanna altrimenti.

## Processo di Maria Stuarda (3/10)

---

Varie ragioni dietro al timore di Elisabetta:

- Maria è regina di Scozia
- Potenziale pericoloso precedente
- Maria è cugina di Elisabetta

## Processo di Maria Stuarda (4/10)

---

Maria rimane tranquilla, consapevole di aver precedentemente cifrato tutti i messaggi della congiura.



## Processo di Maria Stuarda (5/10)

---

**Francis Walsingham**, essendo consapevole di questo, chiamò immediatamente **Thomas Phelippes**, il migliore decifratore d'Inghilterra.

## Processo di Maria Stuarda (6/10)

---

Il metodo di cifratura utilizzato dalla Stuarda per comunicare con gli altri cospiratori, primo tra tutti il giovane **Anthony Babington**, è chiamato **nomenclatore**.

## Processo di Maria Stuarda (7/10)

---

Si utilizzavano 23 simboli da sostituire alle tipiche lettere dell'alfabeto chiaro (escludendo j , v , w) e di 35 simboli che rappresentavano parole o frasi.

# Processo di Maria Stuarda (8/10)

a b c d e f g h i k l m n o p q r s t u x y z  
 o † ʌ # a □ θ ∞ i ð n ll ø ▽ s m f Δ ε c 7 8 9

Nulles ff. — . — . d.

Dowbleth σ

and for with that if but where as of the from by

2 3 4 4 4 3 ʝ ʒ m ʒ x σ

so not when there this in wich is what say me my wyrt

ʝ x † ʝ ʝ x ʝ m n m m d

send lre receave bearer I pray you Mte your name myne

ʝ ʝ † T l l — ʝ ʝ ss

## Processo di Maria Stuarda (9/10)

---

A sua insaputa però, tutta la sua corrispondenza veniva letta e decifrata da **Walsingham**, che alla fine la inganno forgiando un messaggio falso nello scrivere una lista dei suoi collaboratori.

## Processo di Maria Stuarda (10/10)

---

Maria Stuarda viene decapitata l'8 febbraio 1587.

**PERCHÉ ABBIAMO BISOGNO DELLA CRITTOGRAFIA?**

Perché oramai le **informazioni** hanno un diretto e  
irreversibile effetto sulla realtà.

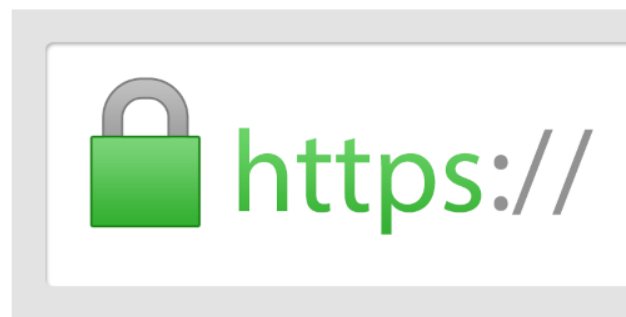


La **crittografia** offre strumenti, tecniche e tecnologie che ci permettono di avere più controllo sul modo in cui le informazioni che ci riguardano influenzano la nostra vita.

Molte realtà di oggi si basano sull'offerta di servizi di crittografia



**Signal**



Cerchiamo quindi di capire come la crittografia si è evoluta nel corso del tempo.

# CRITTOGRAFIA CLASSICA

Iniziamo con qualche **etimologia** (dal greco)

---

- **steganografia:**
  - steganós → "coperto"
  - graphía → "scrittura"
- **crittografia:**
  - kryptós → "nascosto"
  - graphía → "scrittura"

In altre parole,

La **steganografia** vuole nascondere l'intero messaggio, sia il **contenuto** che il **contenitore**.

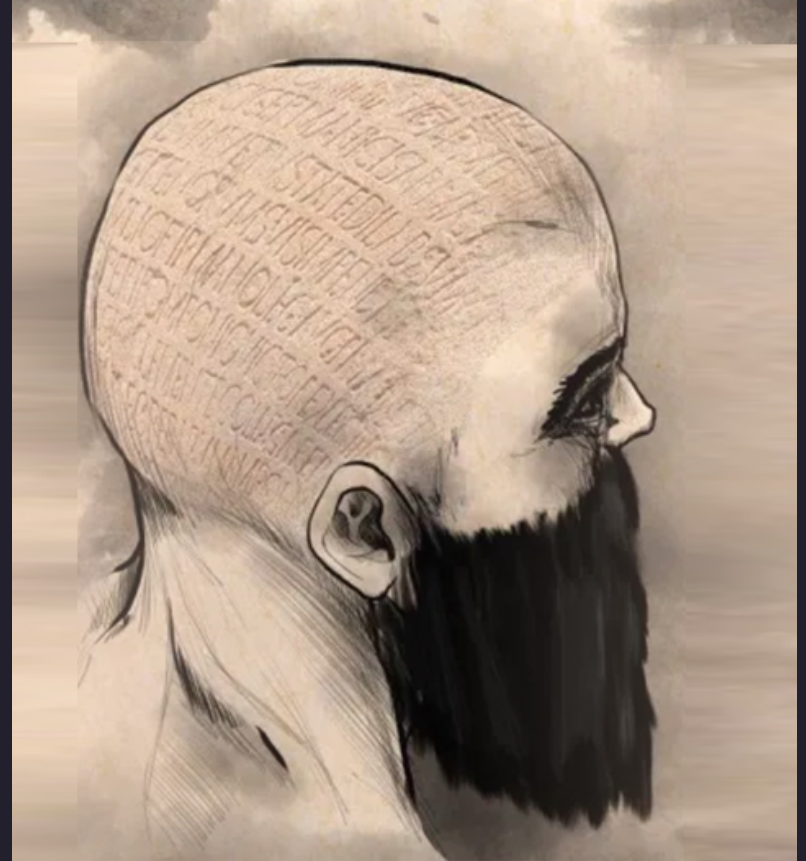
La **crittografia**, invece, vuole nascondere solo il **significato** del messaggio, ovvero solo il contenuto, ma non il contenitore.

**Queste tecniche possono essere combinate tra loro.**

**ESEMPIO DI STEGANOGRAFIA**



Erodoto, uno dei primi scrittori della Storia, racconta la pratica, utilizzata durante le **Guerre persiane** (+2500 anni fa), di radere il capo dei corrieri, scrivere dei messaggi ed aspettare la ricrescita per nascondere i messaggi durante il tragitto.



# ESEMPIO DI CRITTOGRAFIA

## Cifrario di Cesare (1/5)

---

Nascondiamo il significato di un messaggio andando a **spostare** le lettere dell'alfabeto per una data quantità  $c = 3$ .

## Cifrario di Cesare (2/5)

---

Partiamo da un alfabeto in chiaro

ABCDEFGHIJKLMNOPQRSTUVWXYZ

## Cifrario di Cesare (2/5)

---

Per ottenere un alfabeto cifrante

ABCDEFGHIJKLMNOPQRSTUVWXYZ



DEFGHIJKLMNOPQRSTUVWXYZABC

## Cifrario di Cesare (3/5)

---

Data una singola lettera, otteniamo il cifrato utilizzando l'alfabeto cifrante

$$A \longrightarrow A + 3 = D$$

## Cifrario di Cesare (4/5)

---

Se abbiamo tante lettere, ne cifriamo una alla volta

HELLO WORLD



KH00R ZRU0G

## Cifrario di Cesare (5/5)

---

```
#!/usr/bin/env python3
```

```
def main():  
    shift_value = 3  
    cipher = Caesar(shift=shift_value)  
    plaintext = "HELLO WORLD"  
    ciphertext = cipher.encrypt(plaintext)  
    print(f"[c={shift_value}] '{plaintext}' -> '{ciphertext}'")
```

`./code/caesar.py`



# TRASPOSIZIONE E SOSTITUZIONE

Il **cifrario di Cesare** è un **cifrario mono-alfabetico**  
basato sulla **sostituzione**.

In generale i **cifrari classici** lavorano sulle lettere dell'**alfabeto tradizionale** in due modi diversi:

**trasposizione**: le lettere del messaggio sono spostate di posto.

**sostituzione**: le lettere del messaggio sono sostituite con altre lettere.

# CIFRARIO DI VIGENÈRE

Il **cifrario di Vigenère** è una generalizzazione del cifrario di cesare. Al posto di avere un solo alfabeto cifrante, **abbiamo tanti alfabeti cifranti**, che sono utilizzati in modo alternato.

## Esempio (1/4)

---

Supponiamo di avere tre alfabeti cifranti

ABCDEFGHIJKLMNOPQRSTUVWXYZ



ABCDEFGHIJKLMNOPQRSTUVWXYZ

DEFGHIJKLMNOPQRSTUVWXYZABC

CDEFGHIJKLMNOPQRSTUVWXYZAB

## Esempio (2/4)

---

Per cifrare una sequenza di lettere scegliamo in modo sequenziale i vari alfabeti cifranti, e dopo aver cifrato tre lettere torniamo ad utilizzare il primo alfabeto cifrante.

## Esempio (3/4)

---

HELLO WORLD



HHNLR WRTLГ



## Esempio (4/4)

---

Piuttosto che descrivere gli alfabeti cifrante in modo interamente, possiamo abbreviarli utilizzando la prima lettera dell'alfabeto.

ABCDEFGHIJKLMNOPQRSTUVWXYZ → A

DEFGHIJKLMNOPQRSTUVWXYZABC → D

CDEFGHIJKLMNOPQRSTUVWXYZAB → C

La nostra **chiave di cifratura** è dunque ADC.

```
def main():  
    key = "ADC"  
    cipher = Vigenere(key)  
    plaintext = "HELLO WORLD"  
    ciphertext = cipher.encrypt(plaintext)  
    print(f"[key='{key}'] '{plaintext}' -> '{ciphertext}'")
```

`./code/vigenere.py`

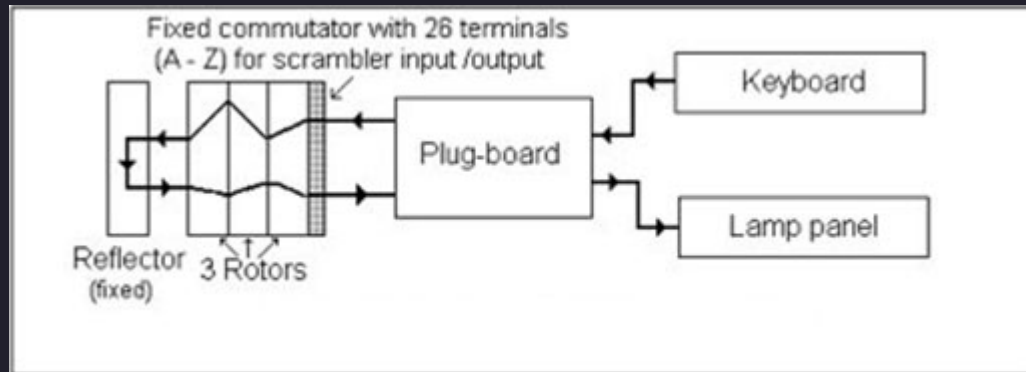
**MACCHINA ENIGMA**

La macchina **Enigma** è un dispositivo **elettro-meccanico** che implementa un cifrario a sostituzione molto complesso.



Enigma è stata utilizzata dai tedeschi e dalle forze dell'Asse durante la seconda guerra mondiale per proteggere le informazioni di guerra.

Premendo un tasto sulla tastiera si chiude un circuito elettrico, accendendo una lampadina.



- tasto sulla tastiera → lettera in chiaro
- lampadina illuminata → lettera cifrata

Per chi fosse interessato, ho implementato un emulatore della macchina enigma in C. Il progetto è disponibile nella seguente github repository

<https://github.com/LeonardoE95/enigma-machine>

```
Enigma> info
Enigma> Current configuration...
      Rotors (from left to right): M3-II, M3-I, M3-III
            Position: 0, 0, 0
            Ring: 0, 0, 0
      Reflector: M3-B
      Plugboard: 6 plugs
                (A, M)
                (F, I)
                (N, V)
                (P, S)
                (T, U)
                (W, Z)
Enigma> encrypt HELLO
MIJEN
```

<https://github.com/LeonardoE95/enigma-machine>



# I PROBLEMI DELLA CRITTOGRAFIA CLASSICA

I primi cifrari, tra cui quello di Cesare e Vigenère, soffrivano di un problema di dimensione rispetto allo **spazio delle chiavi**. Lo spazio delle chiavi di questi cifrari è, semplicemente, troppo piccolo.

Nel **Cifrario di Cesare** abbiamo 26 possibili chiavi.

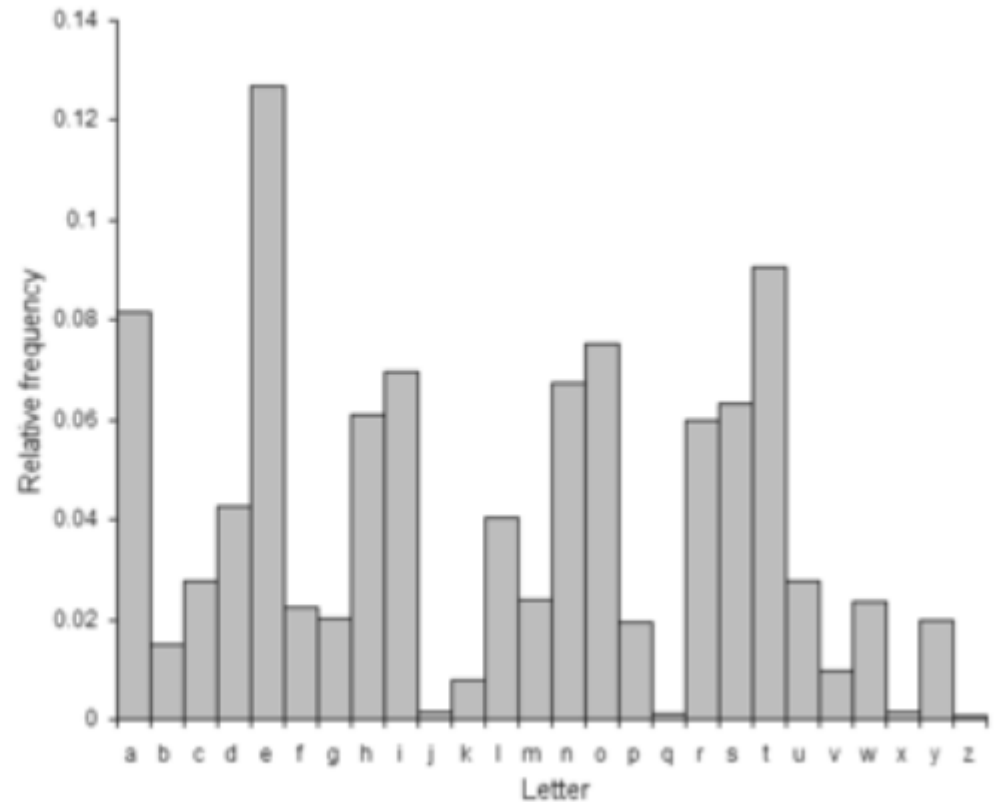
Nel **Cifrario di Vigenéré** abbiamo  $26^n$  possibili chiavi  
per una chiave di dimensione  $n$ .

Oltre alla dimensione dello spazio delle chiavi, un altro problema, assai più profondo, è legato al fatto che questi cifrari lavorano al livello delle **singole lettere**.

Il problema, in particolare, è che **la frequenza delle lettere nei linguaggi naturali NON è uniforme**.

# Frequenza delle lettere in inglese

E	11.1607%	56.88	M	3.0129%	15.36
A	8.4966%	43.31	H	3.0034%	15.31
R	7.5809%	38.64	G	2.4705%	12.59
I	7.5448%	38.45	B	2.0720%	10.56
O	7.1635%	36.51	F	1.8121%	9.24
T	6.9509%	35.43	Y	1.7779%	9.06
N	6.6544%	33.92	W	1.2899%	6.57
S	5.7351%	29.23	K	1.1016%	5.61
L	5.4893%	27.98	V	1.0074%	5.13
C	4.5388%	23.13	X	0.2902%	1.48
U	3.6308%	18.51	Z	0.2722%	1.39
D	3.3844%	17.25	J	0.1965%	1.00
P	3.1671%	16.14	Q	0.1962%	(1)



Questa osservazione ha portato alcuni arabi, intorno all'800, allo sviluppo delle prime tecniche di **crittoanalisi**, il cui obiettivo è quello di rompere i cifrari

- capire la chiave
- decifrare i testi cifrati

# Manuscript on Deciphering Cryptographic Messages



(al-Kindi)

# CRITTOGRAFIA MODERNA



Per passare dalla crittografia classica alla crittografia moderna iniziamo da un principio, il **principio di Kerckhoffs**.

## Principio di Kerckhoffs

---

La sicurezza di un crittosistema non deve dipendere dal tenere celato il critto-algoritmo. La sicurezza deve dipendere solo dal tenere celata la chiave

Un altro cambiamento fondamentale tra la crittografia classica e la crittografia moderna è data dall'**introduzione del bit** come unità fondamentale di informazione.

Lavoro di **Claude Shannon**, che nel suo lavoro di tesi  
dimostrò la connessione tra

**algebra booleana  $\leftrightarrow$  circuiti logici**

**Claude Shannon, A symbolic analysis of relay and switching circuits, 1937**

Per chi fosse interessato, sto portando avanti una serie sui **fondamenti di informatica** che tratta i concetti più elementari di questa nuova arte.



0 , 1

Youtube – Perché usiamo il sistema binario?

**TRIADE CIA**

La crittografia moderna si basa sull'erogazione di **specifici servizi di sicurezza**, tra cui troviamo

- confidentiality (confidenzialità)
- integrity (integrità)
- authentication (autenticazione)

Questi servizi formano la famosa **triade CIA**.

**CONFIDENZIALITÀ**



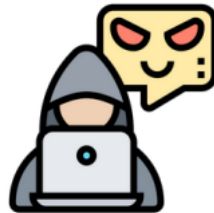
# Confidentiality (1/2)

---

Senza confidenzialità.

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

username=bob&password=12345



## Confidentiality (2/2)

---

Con confidenzialità.

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

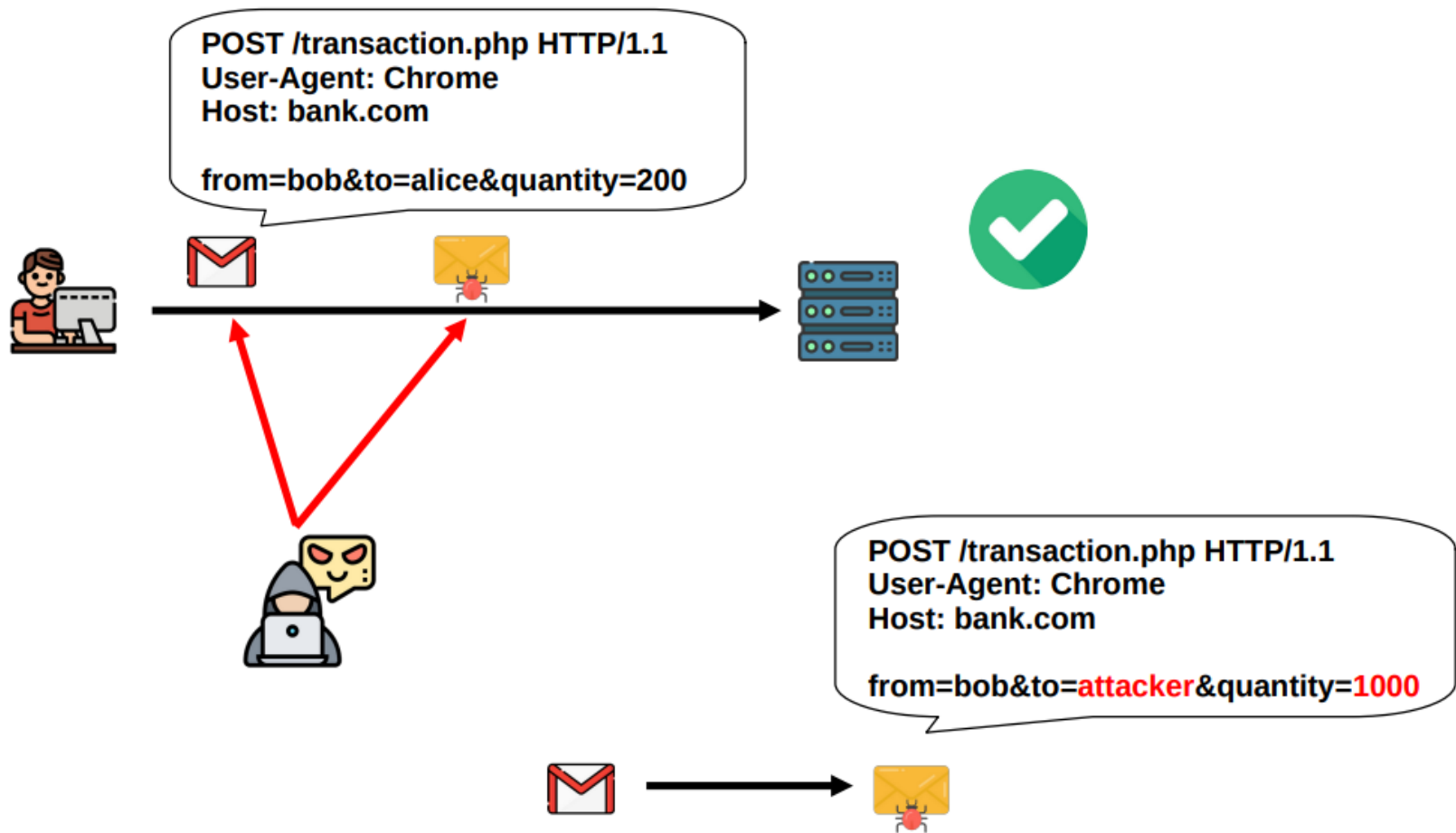


**INTEGRITÀ**

# Integrity (1/2)

---

Senza integrità.

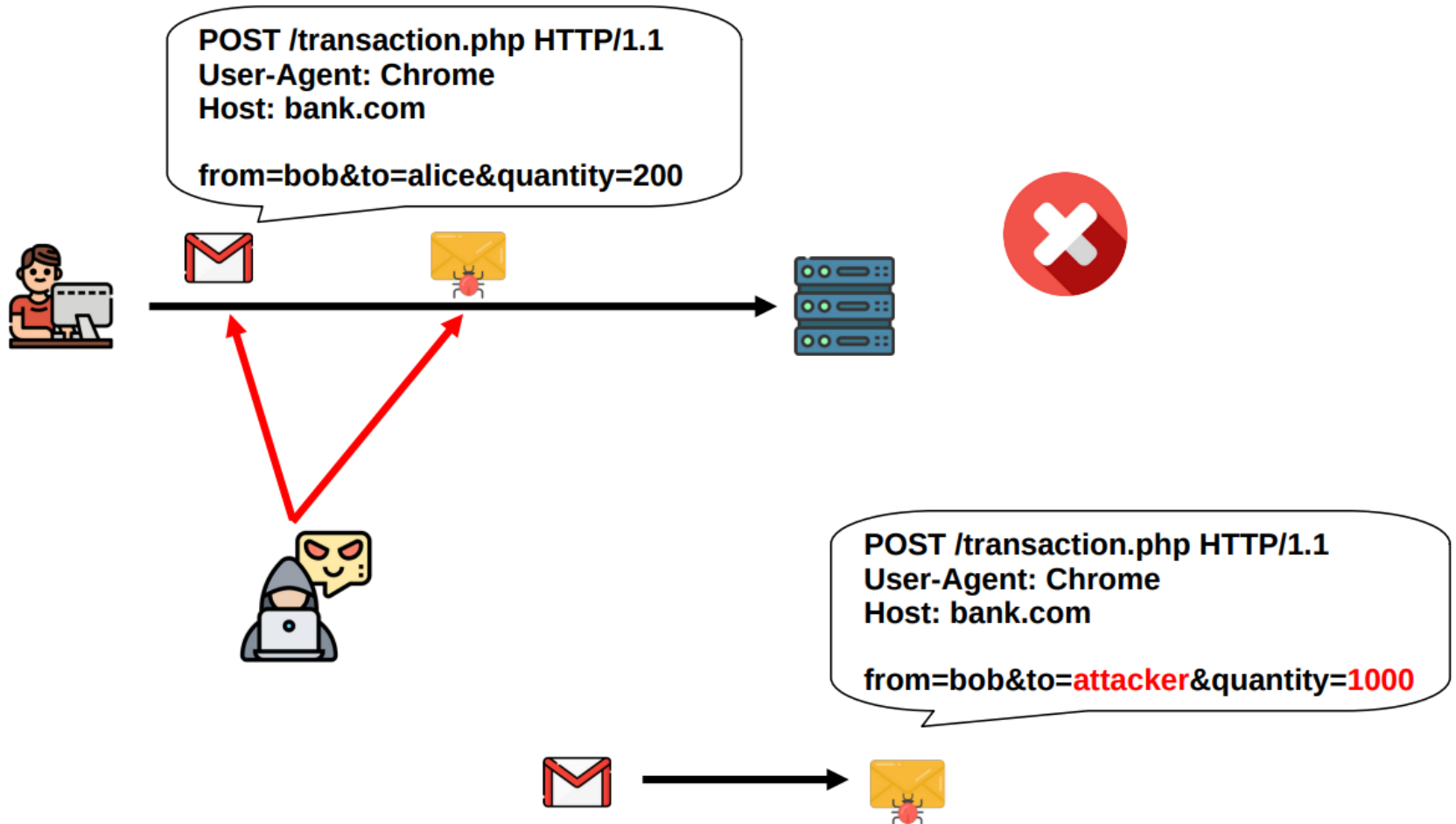


## Integrity (2/2)

---

Con integrità.





**AUTENTICAZIONE**

# Authentication (1/2)

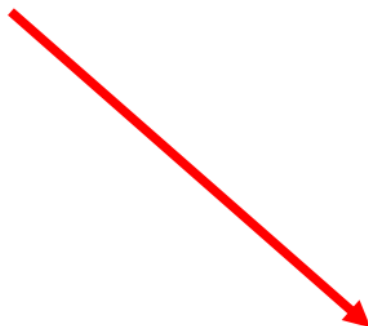
---

Senza autenticazione.



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

username=bob&password=12345



## Authentication (2/2)

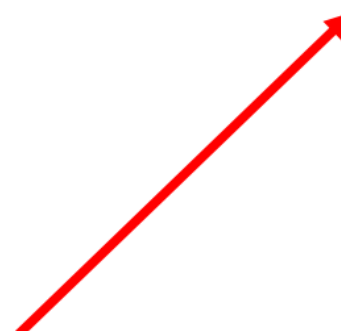
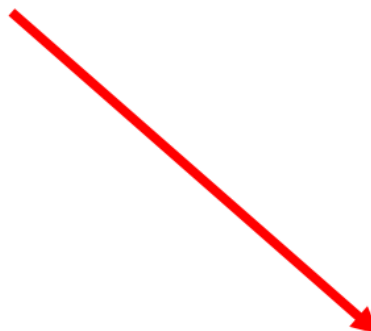
---

Con autenticazione.



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

username=bob&password=12345



# PROTOCOLLI DI CRITTOGRAFIA

La **triade CIA** è implementata in modi diversi in diversi  
**protocolli di rete crittografici.**



Tra i più importanti protocolli di rete crittografici troviamo anche i seguenti

- Transport Layer Security (TLS)
- Secure Socket Shell (SSH)
- Internet Protocol Security (IPsec)

# TRANSPORT LAYER SECURITY

## Transport Layer Security (1/4)

---

Il protocollo **TLS** ad esempio è utilizzato per offrire confidenzialità, integrità e autenticazione in modo **point-to-point** a qualsiasi protocollo di rete che esiste sul web.

## Transport Layer Security (2/4)

---

Inizialmente è stato sviluppato per proteggere le comunicazioni **http**.



+



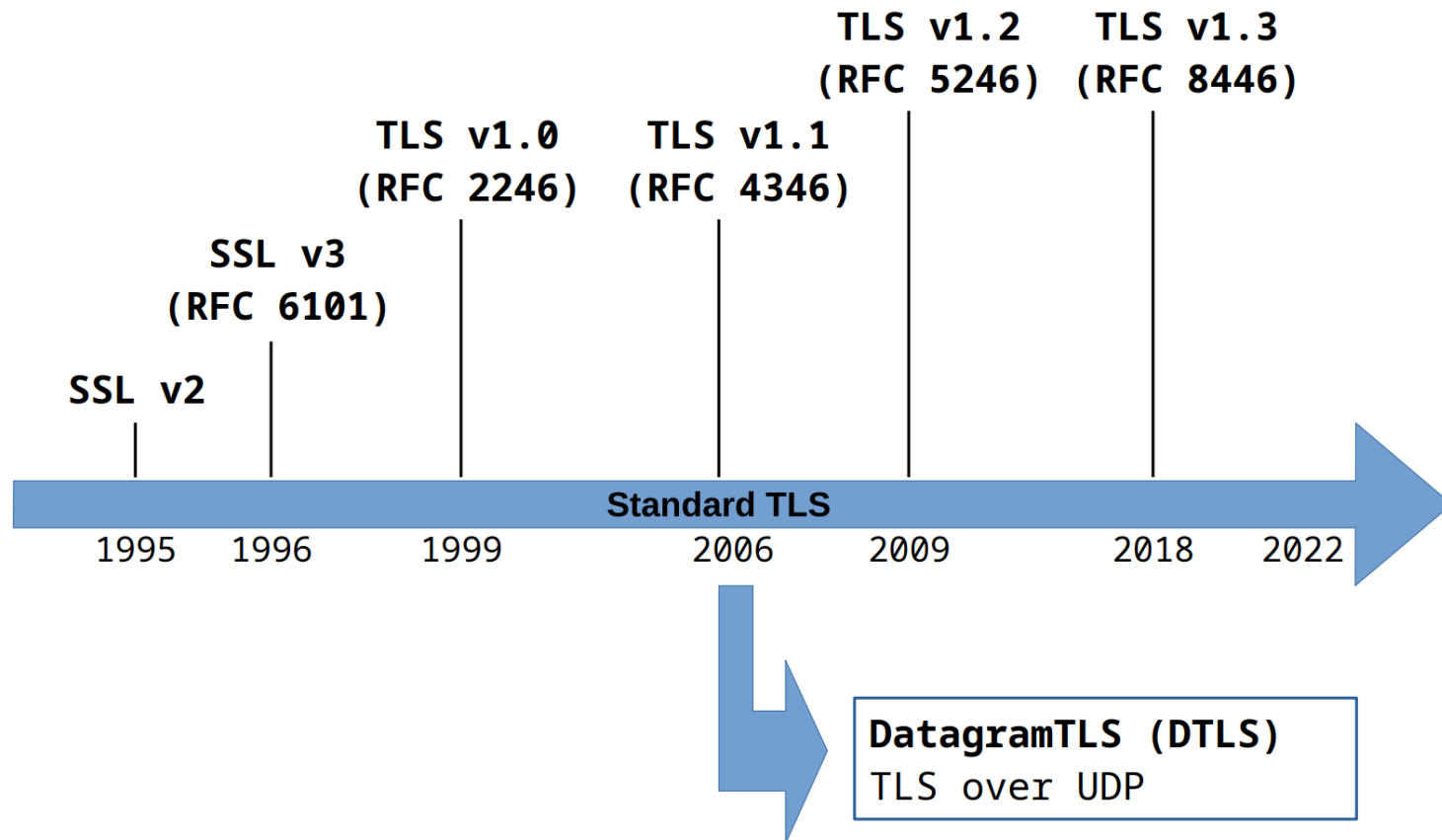
=



## Transport Layer Security (3/4)

---

Il suo funzionamento è piuttosto complesso ed ha subito varie modifiche nel corso degli anni.



## Transport Layer Security (4/4)

---

Per chi è interessato ad approfondire...

Tesi Magistrale Informatica – Introduzione al TLS,  
Attacchi al TLS, TLSPLOIT



# SECURE SOCKET SHELL

## Secure Socket Shell (1/2)

---

Un altro protocollo fondamentale è il protocollo **ssh**, che permette di connettersi tramite il terminale a server remoti in modo sicuro.

## Secure Socket Shell (2/2)

---

```
[leo@archlinux]$ ssh pi@raspberrypi
```

```
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
```

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
```

```
Last login: Thu Nov 16 16:45:30 2023 from 192.168.3.35
```

```
pi@raspberrypi:~$ whoami
pi
```

# PRIMITIVE CRITTOGRAFICHE

I **protocolli di crittografia** sono implementati combinando tra loro delle **primitive crittografiche**.

Tra queste, troviamo:

- **One Time Pad** (Stream Cipher)
- **Advanced Encryption Standard** (Block Cipher)
- **Diffie-Hellman Key Exchange** (Key-Exchange)
- **Rivest-Shamir-Adleman** (Public-key-crypto)

## ONE-TIME-PAD (OTP)

Il **one-time-pad** è un cifrario che lavora a livello del **bit**.

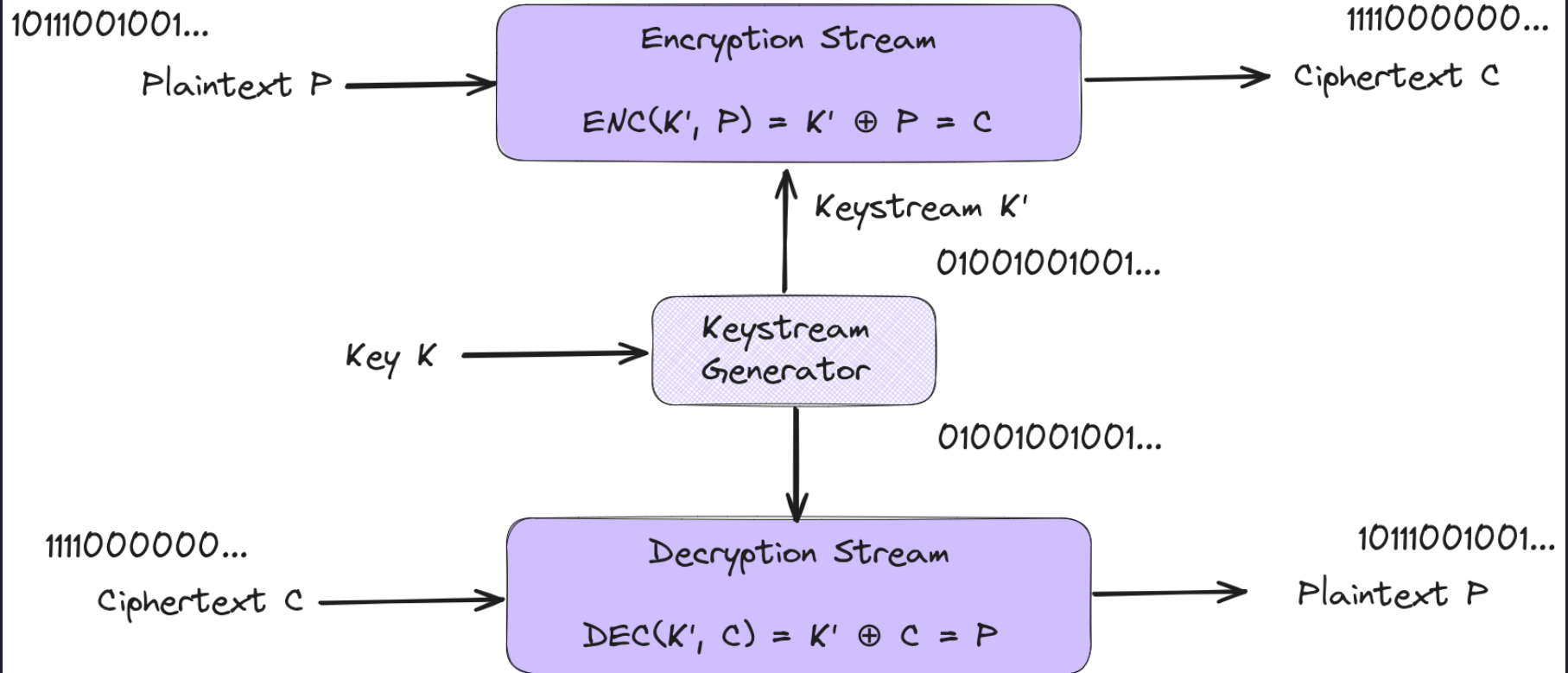
Il cifrario si basa sull'operazione dello **XOR**.

$A$	$B$	$A \oplus B$
0	0	0
1	0	1
0	1	1
1	1	0



I bit cifrati (**ciphertext**) sono calcolati tramite lo XOR dei bit del testo in chiaro (**plaintext**) con i bit di una chiave randomica (**random key**).

$$\text{Plaintext} \oplus \text{Random key} \longrightarrow \text{Ciphertext}$$



Assumendo di avere una chiave  $k$  completamente randomica, l'OTP raggiunge il **livello di sicurezza massimo**

$$P(k == 0) = \frac{1}{2} \quad , \quad P(k == 1) = \frac{1}{2}$$

In altre parole, per un potenziale attaccante il testo cifrato è completamente random.

Per maggiori informazioni vedere

CNS Lecture notes - 02 Semantic Security

```
#!/usr/bin/env python3
```

```
def main():  
    key = list(b"keyyo")  
    otp = OTP(key)  
    plaintext_bytes = list(b"HELLO")  
    ciphertext_bytes = otp.encrypt(plaintext_bytes)  
    print(f"[key={key}] {plaintext_bytes} -> {ciphertext_bytes}")
```

**./code/otp.py**

Altri esempi di stream ciphers sono

- RC4
- Salsa20 / ChaCha
- SNOW

# ADVANCED ENCRYPTION STANDARD (AES)

Nel 26 novembre del 2001 il NIST introduce il cifrario a blocchi **AES** per rimpiazzare il vecchio cifrario a blocchi **DES**.

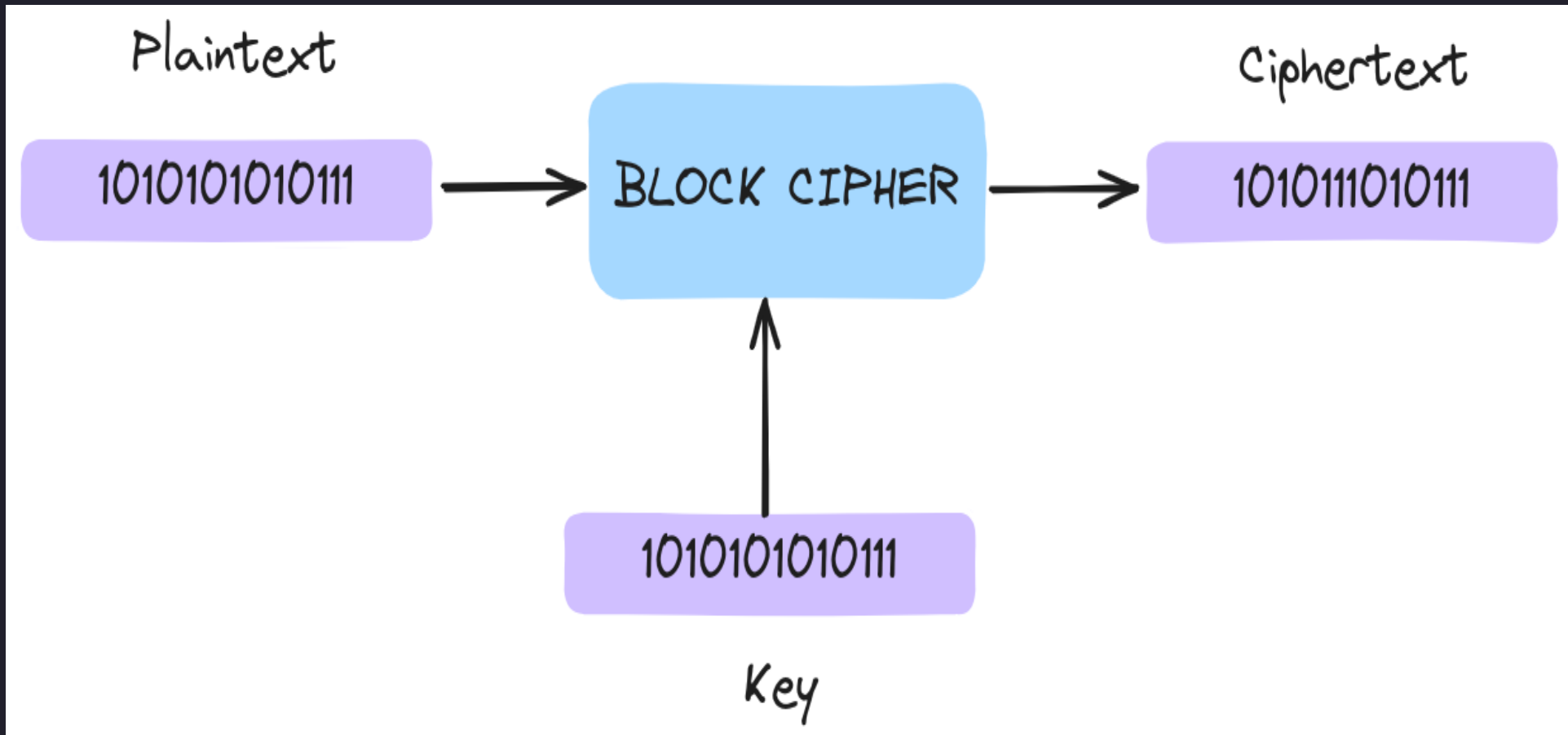
DES → Data Encryption Standard

AES → Advanced Encryption Standard

(NIST → National Institute of Standards and Technology)



A differenza di uno **stream cipher**, i **block ciphers** lavorano su blocchi di dimensione fissa.



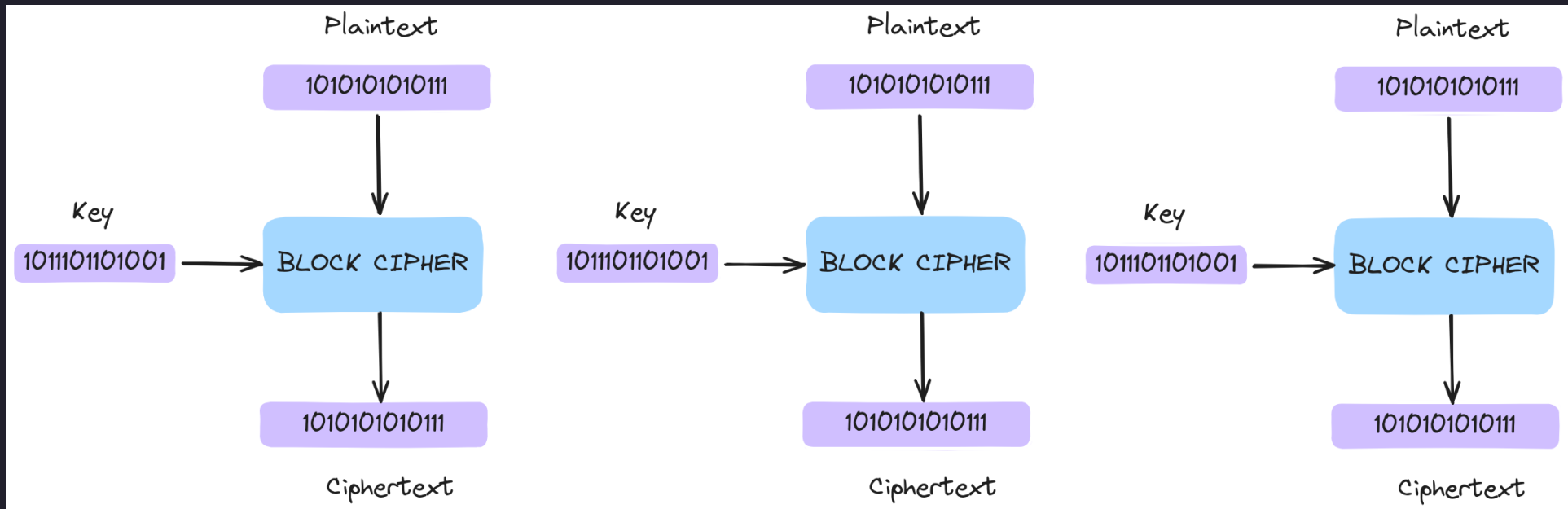
Per quanto riguarda AES, sono state standardizzate tre particolari variazioni, che lavorano su blocchi di dimensione diversa

**AES-128 , AES-192 , AES-256**

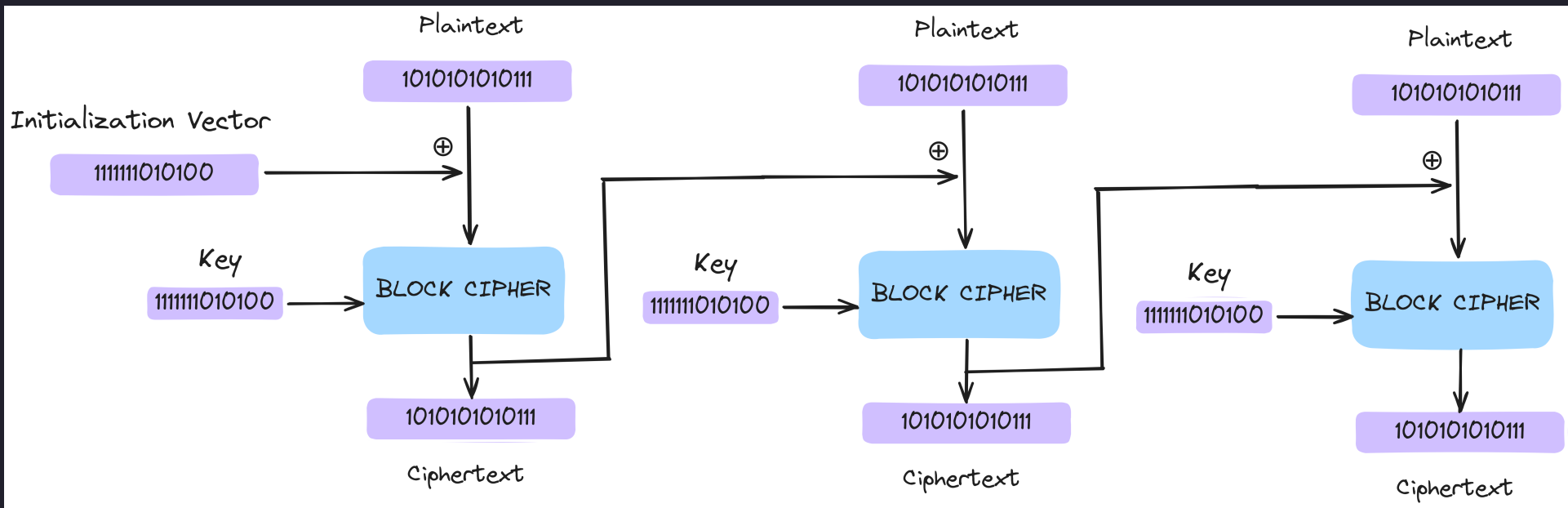
Se abbiamo tanti dati da cifrare, questi vengono suddivisi in blocchi, si aggiunge del **padding** e si cifrano tutti i blocchi. Varie modalità di utilizzo:

- Electronic Codebook (ECB)
- Cipher Block Chaining (CBC)
- Counter (CTR)

# Block Ciphers, ECB Mode



# Block Ciphers, CBC Mode



# pycryptodome library (1/2)

---

```
def encryption_example(plaintext_bytes):  
    global IV, KEY  
  
    b64_plaintext = b64encode(plaintext_bytes)  
  
    aes = AES.new(KEY, AES.MODE_CBC, IV)  
    ciphertext_bytes = aes.encrypt(pad(b64_plaintext, AES.block_size))  
  
    b64_ciphertext = b64encode(ciphertext_bytes)  
  
    return b64_ciphertext
```

./code/aes.py

# pycryptodome library (1/2)

---

```
def decryption_example(b64_ciphertext):  
    global IV, KEY  
  
    ciphertext_bytes = b64decode(b64_ciphertext)  
  
    aes = AES.new(KEY, AES.MODE_CBC, IV)  
    b64_plaintext = unpad(aes.decrypt(ciphertext_bytes), AES.block_size)  
  
    plaintext_bytes = b64decode(b64_plaintext)  
  
    return plaintext_bytes
```

./code/aes.py





## DIFFIE-HELLMAN KEY EXCHANGE (DHE)

Il problema dei cifrari OTP e AES è che per funzionare mittente e destinatario devono condividere una **chiave simmetrica segreta**.

Nasce dunque il problema della **distribuzione delle chiavi**.

Come condivido una chiave segreta in un **canale**  
**insicuro**, se per cifrare i dati ho bisogno di una chiave  
segreta già condivisa?

Nel 1967 I ricercatori Diffie e Hellman mostrarono un modo che può essere utilizzato per condividere una chiave segreta in un canale insicuro senza segreti pre-condivisi.

# New Directions in Cryptography

644

IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. IT-22, NO. 6, NOVEMBER 1976

## New Directions in Cryptography

*Invited Paper*

WHITFIELD DIFFIE AND MARTIN E. HELLMAN, MEMBER, IEEE

**Abstract**—Two kinds of contemporary developments in cryptography are examined. Widening applications of teleprocessing have given rise to a need for new types of cryptographic systems, which minimize the need for secure key distribution channels and supply the equivalent of a written signature. This paper suggests ways to solve these currently open problems. It also discusses how the theories of communication and computation are beginning to provide the tools to solve cryptographic problems of long standing.

The best known cryptographic problem is that of privacy: preventing the unauthorized extraction of information from communications over an insecure channel. In order to use cryptography to insure privacy, however, it is currently necessary for the communicating parties to share a key which is known to no one else. This is done by sending the key in advance over some secure channel such as private courier or registered mail. A private conversation

Questa modalità di arrivare ad una chiave segreta  
condivisa ha preso il nome di

**Diffie Hellman Key Exchange (DHE)**

Il DHE è un primo esempio di **crittografia asimmetrica**, basata sulla presenza di due chiavi:

- **chiave pubblica**
- **chiave privata**





# RIVEST-SHAMIR-ADLEMAN (RSA)

Per quanto DHE rappresenta il primo esempio di crittografia asimmetrica, DHE da solo non basta per avere un vero sistema crittografico, in quanto può essere utilizzato solo per condividere un segreto.

Nel 1977, tre ricercatori Rivest, Shami e Adleman, descrivono il primo vero crittosistema asimmetrico, basato sull'utilizzo di chiavi pubbliche e chiavi private.

Il sistema RSA si basa sull'operazione di **esponenziazione** nel contesto dell'**aritmetica modulare** e sul **teorema di Eulero**.

Il sistema RSA permette di:

- cifrare messaggi
- firmare messaggi

**COSA RENDE LA CRITTOGRAFIA SICURA?**

Le primitive crittografiche, a loro volta, si basano sullo sviluppo di alcune branche della **matematica**.

Il crittosistema RSA è basato sull'osservazione empirica che, dato un numero  $N$ , è computazionalmente difficile fattorizzare tale numero nei suoi fattori primi.



In RSA si scelgono due primi grandi  $p$  e  $q$  e si calcola

$$N = p \cdot q$$

RSA rimane sicuro assumendo che un attaccante non è in grado di fattorizzare  $N$  nei suoi fattori primi  $p$  e  $q$ .

Attualmente non si hanno certezze (teoriche) rispetto alla difficoltà di tali problemi.

Dal punto di vista pragmatico però nessuno ha ancora dimostrato (pubblicamente), una soluzione efficiente a questi problemi.

(Ipotesi  $P=NP$  in **teoria della complessità**)

# CONCLUSIONI

Molte cose non sono state menzionate in questa breve introduzione.

La crittografia è un campo che col passare del tempo sta diventando sempre più sofisticato ed importante per l'umanità.

## Stratificazione della conoscenza crittografica:

- servizi di crittografia
- protocolli di crittografia
- primitive crittografiche
- problemi matematici

## Crittografia applicata

---

Ricordiamo, infine, che questi protocolli devono essere implementati tramite del codice.

<https://heartbleed.com/>



# REFS

- <https://blog.miniserver.it/pfsense/openvpn-vs-ipsec/>
- <https://www-ee.stanford.edu/~hellman/publications/24.pdf>

