

Cryptography

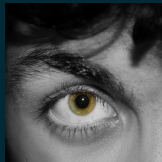
Hackers ahead of time

Gaspare Ferraro
ferraro@gaspa.re

November 16, 2018



Visit us!



@GaspareG

Part I

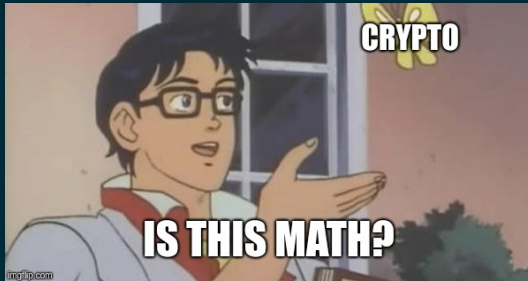
Introduzione

Warning!

In questo incontro si fa uso della *matematica*!

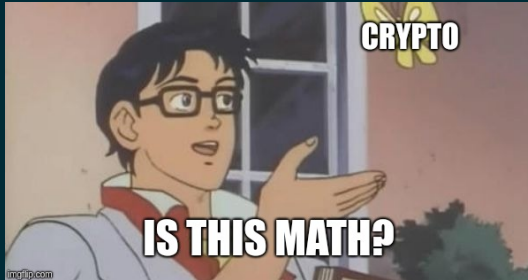
Warning!

In questo incontro si fa uso della *matematica*!



Warning!

In questo incontro si fa uso della *matematica*!



Non è sempre stato così però...

La crittografia ieri

La crittografia ieri



(c) Cifrario di Cesare

La crittografia ieri



(e) Cifrario di Cesare



(f) Scitala

La crittografia oggi

Le necessità, così come le risorse a disposizione, si sono evolute ed oggi possiamo suddividere la crittografia in:

La crittografia oggi

Le necessità, così come le risorse a disposizione, si sono evolute ed oggi possiamo suddividere la crittografia in:

(EN|DE)CRYPTION

ASYMMETRIC (RSA, ECC, ...)
SYMMETRIC (DES, AES, ...)

KEY EXCHANGE

RSA, DH, ECDH, ...

AUTHENTICATION

RSA, DSA, ECDSA, ...

HASHING

MD5, SHA-1, SHA-256, ...

Part II

Crittografia simmetrica

Crittografia simmetrica

I cifrari simmetrici sono quelli dove i messaggi m vengono cifrati e decifrati usando una stessa chiave k , che deve essere nota esclusivamente alle due parti.

$\mathcal{C}(m, k) = c$ (funzione di cifratura)

$\mathcal{D}(c, k) = m$ (funzione di decifratura)

Ovviamente deve valere che:

$\mathcal{D}(\mathcal{C}(m, k), k) = m$ (il messaggio originale non viene alterato durante lo scambio).

Per esempio nel cifrario di Cesare:

$\mathcal{C}(m, k)$ = ruota in avanti di k ogni singolo carattere.

$\mathcal{D}(c, k)$ = ruota indietro di k ogni singolo carattere.

XOR cipher

Crittoanalisi statistica

Spesso le vulnerabilità non è nell'algoritmo ma nella sua applicazione...



One-time Pad

Il Cifrario di Vernam .

Chiamato anche one-time pad poichè

Part III

Crittografia asimmetrica

Concetti base

La crittografia asimmetrica si basa sulle funzioni *one-way trapdoor* e sulla presenza di una coppia di chiavi (chiamate chiave pubblica e chiave privata).

Concetti base

La crittografia asimmetrica si basa sulle funzioni *one-way trapdoor* e sulla presenza di una coppia di chiavi (chiamate chiave pubblica e chiave privata).

Una funzione f si dice *trapdoor* se:

- ▶ Calcolare $y = f(x)$ è computazionalmente facile.
- ▶ Calcolare $x = f^{-1}(y)$ è computazionalmente difficile (*senza nessuna informazione aggiuntiva*).

Concetti base

La crittografia asimmetrica si basa sulle funzioni *one-way trapdoor* e sulla presenza di una coppia di chiavi (chiamate chiave pubblica e chiave privata).

Una funzione f si dice *trapdoor* se:

- ▶ Calcolare $y = f(x)$ è computazionalmente facile.
- ▶ Calcolare $x = f^{-1}(y)$ è computazionalmente difficile (*senza nessuna informazione aggiuntiva*).

Un esempio è il problema della fattorizzazione:

- ▶ $m = f(\{p, q\}) = (p \times q)$ (calcolo del prodotto tra i primi p e q).
- ▶ $\{p, q\} = f^{-1}(n) = ??$ (scomposizione in fattori primi di n).

Concetti base

La crittografia asimmetrica si basa sulle funzioni *one-way trapdoor* e sulla presenza di una coppia di chiavi (chiamate chiave pubblica e chiave privata).

Una funzione f si dice *trapdoor* se:

- ▶ Calcolare $y = f(x)$ è computazionalmente facile.
- ▶ Calcolare $x = f^{-1}(y)$ è computazionalmente difficile (*senza nessuna informazione aggiuntiva*).

Un esempio è il problema della fattorizzazione:

- ▶ $m = f(\{p, q\}) = (p \times q)$ (calcolo del prodotto tra i primi p e q).
- ▶ $\{p, q\} = f^{-1}(n) = ??$ (scomposizione in fattori primi di n).

$f(\{49171, 61843\}) = 3040882153$ (facile quanto aprire una calcolatrice).

$f^{-1}(1841488427) = ??$ (devo provare tutti i divisori da 2 a \sqrt{n}).

Concetti base

La crittografia asimmetrica si basa sulle funzioni *one-way trapdoor* e sulla presenza di una coppia di chiavi (chiamate chiave pubblica e chiave privata).

Una funzione f si dice *trapdoor* se:

- ▶ Calcolare $y = f(x)$ è computazionalmente facile.
- ▶ Calcolare $x = f^{-1}(y)$ è computazionalmente difficile (*senza nessuna informazione aggiuntiva*).

Un esempio è il problema della fattorizzazione:

- ▶ $m = f(\{p, q\}) = (p \times q)$ (calcolo del prodotto tra i primi p e q).
- ▶ $\{p, q\} = f^{-1}(n) = ??$ (scomposizione in fattori primi di n).

$f(\{49171, 61843\}) = 3040882153$ (facile quanto aprire una calcolatrice).

$f^{-1}(1841488427) = ??$ (devo provare tutti i divisori da 2 a \sqrt{n}).

Il problema diventa banale se conosco uno dei due divisori:

$$1841488427 / 58049 = 31723$$

Aritmetica modulare

Diciamo che due interi a e b sono congrui modulo n , scritto $a \equiv b \pmod{n}$, se $(a \% n) = (b \% n)$ dove $\%$ è il resto della divisione intero (modulo).

Aritmetica modulare

Diciamo che due interi a e b sono congrui modulo n , scritto $a \equiv b \pmod{n}$, se $(a \% n) = (b \% n)$ dove $\%$ è il resto della divisione intero (modulo).

Alcune proprietà matematiche:

Aritmetica modulare

Diciamo che due interi a e b sono congrui modulo n , scritto $a \equiv b \pmod{n}$, se $(a \% n) = (b \% n)$ dove $\%$ è il resto della divisione intero (modulo).

Alcune proprietà matematiche:

- ▶ $a + k \equiv b + k \pmod{n}$, invariante per addizione.
- ▶ $k * a \equiv k * b \pmod{n}$, invariante per moltiplicazione.
- ▶ $a^k \equiv b^k \pmod{n}$, invariante per potenza.

Aritmetica modulare

Diciamo che due interi a e b sono congrui modulo n , scritto $a \equiv b \pmod{n}$, se $(a \% n) = (b \% n)$ dove $\%$ è il resto della divisione intero (modulo).

Alcune proprietà matematiche:

- ▶ $a + k \equiv b + k \pmod{n}$, invariante per addizione.
- ▶ $k * a \equiv k * b \pmod{n}$, invariante per moltiplicazione.
- ▶ $a^k \equiv b^k \pmod{n}$, invariante per potenza.
- ▶ $\sqrt{a} \equiv b \pmod{n}$ se $a \equiv b^2 \pmod{n}$, radice quadrata.
- ▶ $a^{-1} \equiv b \pmod{n}$ se $ab \equiv 1 \pmod{n}$, inverso moltiplicativo.
- ▶ ...

RSA pt.1

RSA pt.1

L'algoritmo di cifratura asimmetrica più famoso è l'RSA (da Rivest Shamir Adleman) che si basa sul problema della fattorizzazione e sull'algebra modulare.

RSA pt.1

L'algoritmo di cifratura asimmetrica più famoso è l'RSA (da Rivest Shamir Adleman) che si basa sul problema della fattorizzazione e sull'algebra modulare.

Il funzionamento di base è:

RSA pt.1

L'algoritmo di cifratura asimmetrica più famoso è l'RSA (da Rivest Shamir Adleman) che si basa sul problema della fattorizzazione e sull'algebra modulare.

Il funzionamento di base è:

- ▶ Si scelgono due numeri primi a caso p e q (in modo *sicuro*).

RSA pt.1

L'algoritmo di cifratura asimmetrica più famoso è l'RSA (da Rivest Shamir Adleman) che si basa sul problema della fattorizzazione e sull'algebra modulare.

Il funzionamento di base è:

- ▶ Si scelgono due numeri primi a caso p e q (in modo *sicuro*).
- ▶ Si calcola il prodotto $n = p * q$ (che sarà il nostro modulo).

RSA pt.1

L'algoritmo di cifratura asimmetrica più famoso è l'RSA (da Rivest Shamir Adleman) che si basa sul problema della fattorizzazione e sull'algebra modulare.

Il funzionamento di base è:

- ▶ Si scelgono due numeri primi a caso p e q (in modo *sicuro*).
- ▶ Si calcola il prodotto $n = p * q$ (che sarà il nostro modulo).
- ▶ Si calcola il toziente $\phi(n) = (p - 1) * (q - 1)$ (il numero dei coprimi con n).

RSA pt.1

L'algoritmo di cifratura asimmetrica più famoso è l'RSA (da Rivest Shamir Adleman) che si basa sul problema della fattorizzazione e sull'algebra modulare.

Il funzionamento di base è:

- ▶ Si scelgono due numeri primi a caso p e q (in modo *sicuro*).
- ▶ Si calcola il prodotto $n = p * q$ (che sarà il nostro modulo).
- ▶ Si calcola il toziente $\phi(n) = (p - 1) * (q - 1)$ (il numero dei coprimi con n).
- ▶ Si sceglie a caso un numero e , coprimo e minore di $\phi(n)$.

RSA pt.1

L'algoritmo di cifratura asimmetrica più famoso è l'RSA (da Rivest Shamir Adleman) che si basa sul problema della fattorizzazione e sull'algebra modulare.

Il funzionamento di base è:

- ▶ Si scelgono due numeri primi a caso p e q (in modo *sicuro*).
- ▶ Si calcola il prodotto $n = p * q$ (che sarà il nostro modulo).
- ▶ Si calcola il toziente $\phi(n) = (p - 1) * (q - 1)$ (il numero dei coprimi con n).
- ▶ Si sceglie a caso un numero e , coprimo e minore di $\phi(n)$.
- ▶ Si calcola il numero d come inverso moltiplicativo di e , ovvero $ed \equiv 1 \pmod{\phi(n)}$.

RSA pt.1

L'algoritmo di cifratura asimmetrica più famoso è l'RSA (da Rivest Shamir Adleman) che si basa sul problema della fattorizzazione e sull'algebra modulare.

Il funzionamento di base è:

- ▶ Si scelgono due numeri primi a caso p e q (in modo *sicuro*).
- ▶ Si calcola il prodotto $n = p * q$ (che sarà il nostro modulo).
- ▶ Si calcola il toziente $\phi(n) = (p - 1) * (q - 1)$ (il numero dei coprimi con n).
- ▶ Si sceglie a caso un numero e , coprimo e minore di $\phi(n)$.
- ▶ Si calcola il numero d come inverso moltiplicativo di e , ovvero $ed \equiv 1 \pmod{\phi(n)}$.

Chiamiamo quindi:

RSA pt.1

L'algoritmo di cifratura asimmetrica più famoso è l'RSA (da Rivest Shamir Adleman) che si basa sul problema della fattorizzazione e sull'algebra modulare.

Il funzionamento di base è:

- ▶ Si scelgono due numeri primi a caso p e q (in modo *sicuro*).
- ▶ Si calcola il prodotto $n = p * q$ (che sarà il nostro modulo).
- ▶ Si calcola il toziente $\phi(n) = (p - 1) * (q - 1)$ (il numero dei coprimi con n).
- ▶ Si sceglie a caso un numero e , coprimo e minore di $\phi(n)$.
- ▶ Si calcola il numero d come inverso moltiplicativo di e , ovvero $ed \equiv 1 \pmod{\phi(n)}$.

Chiamiamo quindi:

$k_{pub} = (n, e)$ la chiave pubblica che distribuiamo.

RSA pt.1

L'algoritmo di cifratura asimmetrica più famoso è l'RSA (da Rivest Shamir Adleman) che si basa sul problema della fattorizzazione e sull'algebra modulare.

Il funzionamento di base è:

- ▶ Si scelgono due numeri primi a caso p e q (in modo *sicuro*).
- ▶ Si calcola il prodotto $n = p * q$ (che sarà il nostro modulo).
- ▶ Si calcola il toziente $\phi(n) = (p - 1) * (q - 1)$ (il numero dei coprimi con n).
- ▶ Si sceglie a caso un numero e , coprimo e minore di $\phi(n)$.
- ▶ Si calcola il numero d come inverso moltiplicativo di e , ovvero $ed \equiv 1 \pmod{\phi(n)}$.

Chiamiamo quindi:

$k_{pub} = (n, e)$ la chiave pubblica che distribuiamo.

$k_{priv} = (n, d)$ la chiave privata che teniamo segreta.

RSA pt.2

Una volta calcolata la chiave pubblica e quella privata possiamo cifrare e decifrare i messaggi in questo modo:

RSA pt.2

Una volta calcolata la chiave pubblica e quella privata possiamo cifrare e decifrare i messaggi in questo modo:

$$\mathcal{C}(m, k_{pub}) = m^e \pmod{n}$$

RSA pt.2

Una volta calcolata la chiave pubblica e quella privata possiamo cifrare e decifrare i messaggi in questo modo:

$$\mathcal{C}(m, k_{pub}) = m^e \pmod{n}$$

$$\mathcal{D}(c, k_{priv}) = c^d \pmod{n}$$

RSA pt.2

Una volta calcolata la chiave pubblica e quella privata possiamo cifrare e decifrare i messaggi in questo modo:

$$\mathcal{C}(m, k_{pub}) = m^e \pmod{n}$$

$$\mathcal{D}(c, k_{priv}) = c^d \pmod{n}$$

Si ma perchè funziona?

RSA pt.2

Una volta calcolata la chiave pubblica e quella privata possiamo cifrare e decifrare i messaggi in questo modo:

$$\mathcal{C}(m, k_{pub}) = m^e \pmod{n}$$

$$\mathcal{D}(c, k_{priv}) = c^d \pmod{n}$$

Si ma perchè funziona?

Dal teorema di Eulero sappiamo che $m^{\phi(n)} \equiv 1 \pmod{n}$.

I valori e e d sono calcolati in modo che $ed \equiv 1 \pmod{\phi(n)}$.

$$\mathcal{D}(\mathcal{C}(m, k), k) = m^{ed} = m^{\phi(n)*t+1} = m^{\phi(n)^t} * m = 1 * m = m$$

Un esempio per capire

Un esempio per capire

Scegliamo i seguenti parametri:

- ▶ $p = 13$ e $q = 23$
- ▶ $n = p * q = 299$
- ▶ $\phi(n) = (13 - 1) * (23 - 1) = 264$
- ▶ $e = 7$, $\gcd(264, 7) = 1$
- ▶ $d = 151$, $7 * 151 \equiv 1 \pmod{264}$

Un esempio per capire

Scegliamo i seguenti parametri:

- ▶ $p = 13$ e $q = 23$
- ▶ $n = p * q = 299$
- ▶ $\phi(n) = (13 - 1) * (23 - 1) = 264$
- ▶ $e = 7$, $\gcd(264, 7) = 1$
- ▶ $d = 151$, $7 * 151 \equiv 1 \pmod{264}$

Quindi $k_{pub} = (299, 7)$ e $k_{priv} = (299, 151)$. Vogliamo cifrare $m = 42$.

Un esempio per capire

Scegliamo i seguenti parametri:

- ▶ $p = 13$ e $q = 23$
- ▶ $n = p * q = 299$
- ▶ $\phi(n) = (13 - 1) * (23 - 1) = 264$
- ▶ $e = 7$, $\gcd(264, 7) = 1$
- ▶ $d = 151$, $7 * 151 \equiv 1 \pmod{264}$

Quindi $k_{pub} = (299, 7)$ e $k_{priv} = (299, 151)$. Vogliamo cifrare $m = 42$.

$$\mathcal{C}(m, k_{pub}) = m^e = 42^7 = 230539333248 = 107 \pmod{299}$$

Un esempio per capire

Scegliamo i seguenti parametri:

- ▶ $p = 13$ e $q = 23$
- ▶ $n = p * q = 299$
- ▶ $\phi(n) = (13 - 1) * (23 - 1) = 264$
- ▶ $e = 7$, $\gcd(264, 7) = 1$
- ▶ $d = 151$, $7 * 151 \equiv 1 \pmod{264}$

Quindi $k_{pub} = (299, 7)$ e $k_{priv} = (299, 151)$. Vogliamo cifrare $m = 42$.

$$\mathcal{C}(m, k_{pub}) = m^e = 42^7 = 230539333248 = 107 \pmod{299}$$

$$\mathcal{D}(c, k_{priv}) = c^d = 107^{151} = 2743956545...948643 = 42 \pmod{299}$$

(Non tanto) a caso

(Non tanto) a caso

"Si scelgono due numeri primi *a caso* p e q (in modo *sicuro*)"

- ▶ Scegliere p e q di almeno 1024 bit.
- ▶ Scegliere p e q non troppo vicini tra loro.
- ▶ Non riusare uno dei primi per altri moduli.

(Non tanto) a caso

"Si scelgono due numeri primi *a caso* p e q (in modo *sicuro*)"

- ▶ Scegliere p e q di almeno 1024 bit.
- ▶ Scegliere p e q non troppo vicini tra loro.
- ▶ Non riutare uno dei primi per altri moduli.

Con la potenza di calcolo attuale è possibile fattorizzare semiprimi fino a (circa) 768 bit, i moduli più grandi sono (per ora) resistenti agli attacchi bruteforce.

(Non tanto) a caso

"Si scelgono due numeri primi *a caso* p e q (in modo *sicuro*)"

- ▶ Scegliere p e q di almeno 1024 bit.
- ▶ Scegliere p e q non troppo vicini tra loro.
- ▶ Non riutare uno dei primi per altri moduli.

Con la potenza di calcolo attuale è possibile fattorizzare semiprimi fino a (circa) 768 bit, i moduli più grandi sono (per ora) resistenti agli attacchi bruteforce.

Se p e q sono vicini allora abbiamo che $n \simeq p^2 \simeq q^2$ e quindi anche \sqrt{n} sarà vicino ai primi. Basterà quindi un attacco bruteforce che cerca i fattori vicino alla radice quadrata.

(Non tanto) a caso

"Si scelgono due numeri primi *a caso* p e q (in modo *sicuro*)"

- ▶ Scegliere p e q di almeno 1024 bit.
- ▶ Scegliere p e q non troppo vicini tra loro.
- ▶ Non riutare uno dei primi per altri moduli.

Con la potenza di calcolo attuale è possibile fattorizzare semiprimi fino a (circa) 768 bit, i moduli più grandi sono (per ora) resistenti agli attacchi bruteforce.

Se p e q sono vicini allora abbiamo che $n \simeq p^2 \simeq q^2$ e quindi anche \sqrt{n} sarà vicino ai primi. Basterà quindi un attacco bruteforce che cerca i fattori vicino alla radice quadrata.

Se $n_1 = p * q'$ e $n_2 = p * q''$ allora $p = \gcd(n_1, n_2)$.

Part IV

Hashing

Creare confusione

Chiamiamo hash una funzione *non invertibile* che associa

► Resis

Creare confusione

Chiamiamo hash una funzione *non invertibile* che associa

- ▶ Resis

Funzioni di hash famose sono l'MD5, SHA-1, SHA-256, SHA-384, ...

Creare confusione

Chiamiamo hash una funzione *non invertibile* che associa

- ▶ Resis

Funzioni di hash famose sono l'MD5, SHA-1, SHA-256, SHA-384, ...


Utilizzi pratici:

- ▶ Salvataggio delle password (preferibilmente con un salt).
- ▶ Creazione di hashtable (struttura dati con accesso diretto).
- ▶ Controllo di integrità (*md5sum*, *sha1sum*).
- ▶ Firma digitale (sull'hash invece che sul documento).

Alla ricerca dell'inversa

Un (pessimo) esempio...

Un (pessimo) esempio...




Gentile GASPARE FERRARO,

come da tua richiesta, ecco i dati relativi al tuo Account



Riepilogo dati registrazione

USER-ID	Gasparg
PASSWORD	XZA113254A

www.trenitalia.com



Un (pessimo) esempio...




Gentile GASPARE FERRARO,

come da tua richiesta, ecco i dati relativi al tuo Account

Riepilogo dati registrazione

USER-ID	Gasparg
PASSWORD	XZA13254A

www.trenitalia.com



Fine