# Robustezza crittografica della PEC

Prof. Massimiliano Sala

Università degli Studi di Trento, Lab di Matematica Industriale e Crittografia

Trento, 21 Novembre 2011

1. Regole tecniche
   - La struttura
   - La sicurezza
2. Aspetti matematici
   - La firma digitale
   - Network Security
3. Possibili debolezze
   - Chiavi deboli

# Regole tecniche

# Le parti in gioco

Alla trasmissione di un messaggio PEC partecipano diverse entita':

- Il mittente, che vuole inviare un messaggio PEC
- Il destinatario, al quale il mittente vuole recapitare il messaggio PEC
- Il gestore del mittente, che mantiene un rapporto contrattuale con il mittente per quanto riguarda i servizi PEC
- Il gestore del destinatario, che mantiene un rapporto contrattuale con il destinatario per quanto riguarda i servizi PEC
- La rete internet (piu' in generale la rete di comunicazione)
- Il messaggio PEC

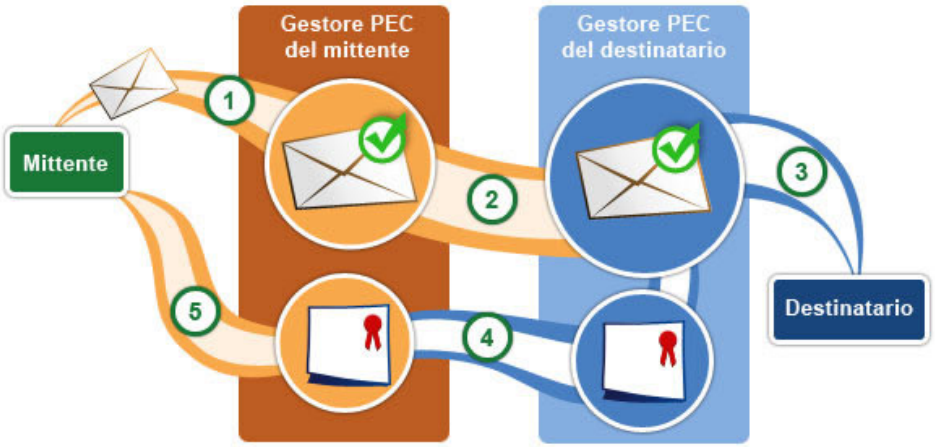
# Trasmissione di un messaggio PEC

sistemare

- Il mittente sottopone il messaggio PEC al gestore mittente.
  Il gestore mittente riconoscera' il mittente solo dopo la sua autenticazione.
- Il gestore mittente verifica la correttezza formale del messaggio PEC e, in caso positivo, restituisce al mittente la ricevuta di accettazione come riconoscimento dell'avvenuto invio del messaggio.
  La ricevuta e' firmata digitalmente dal gestore e garantisce l'integrita' dell'intero messaggio con i suoi allegati
- Il gestore mittente invia il messaggio al gestore destinatario inserendolo in una busta di trasporto **firmata** per permettere al gestore destinatario di verificarne l'inalterabilita' durante il trasporto.
  La busta, per definizione, contiene il messaggio e i suoi allegati, che quindi sono a loro volta protetti dalla firma del gestore.

# Trasmissione di un messaggio PEC

- Il gestore destinatario, ricevuto il messaggio PEC, consegnera' al gestore mittente una ricevuta di presa in carico che attesta il passaggio di consegne tra i due gestori.
  Il gestore destinatario verifica la correttezza e integrita' del messaggio e si accerta che non siano presenti virus informatici.
- Nel caso il messaggio superi i suddetti controlli, viene consegnato alla casella di posta del destinatario che puo' quindi leggerne il contenuto.
- Al mittente perviene una ricevuta di avvenuta consegna, che attesta la disponibilita' del messaggio presso il destinatario.
  La ricevuta e' ancora una volta firmata digitalmente e attesta l'integrita' del contenuto trasmesso.



# Il nodo cruciale

Il nodo fondamentale nella trasmissione di un messaggio PEC e' il passaggio di consegne certificato tra il gestore mittente e il gestore destinatario.

gestore mittente: firma digitalmente il messaggio

gestore destinatario: controlla la validita' della firma e recapita il messaggio

# Da cosa e' garantita la sicurezza?

Essenzialmente la sicurezza della PEC risiede nella **robustezza** della **firma digitale**

Trattandosi di transizioni on-line, bisogna ovviamente considerare anche la network security $\longrightarrow$ sicurezza della comunicazione browser-server.

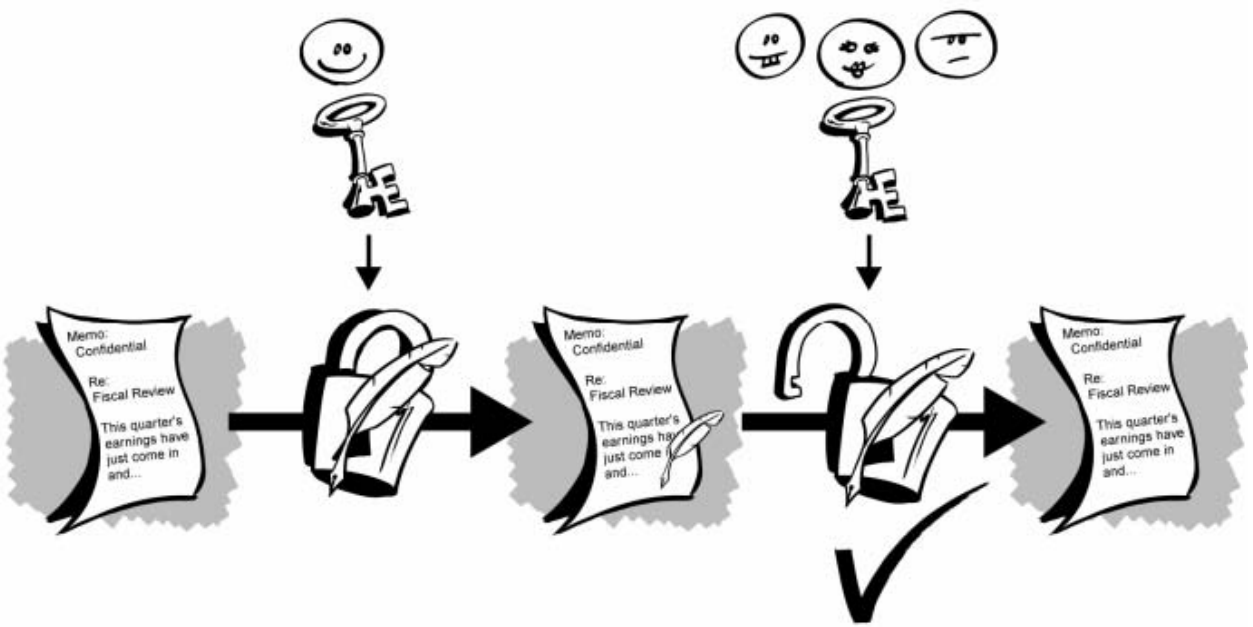
# Aspetti matematici

# Firma Digitale

Cos'e': un espediente per autenticare una qualunque sequenza di cifre binarie, indipendentemente dal significato.

E' ottenuta tramite l'utilizzo di

Funzione Hash + Algoritmo di crittografia asimmetrica



# Cosa si usa in italia?

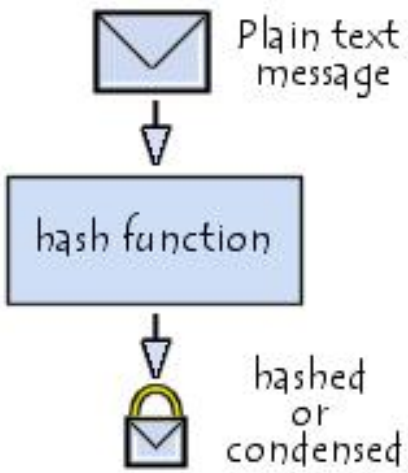
La PEC si basa su standard internazionali IETF e ISO, in particolare:

Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List

Compressione $\Longrightarrow$ SHA-1/SHA-256
Firma $\Longrightarrow$ DSA/RSA o curve ellittiche
(specifiche nel documento **ETSI TS 102 176.1 V2.0.0**)

# hash functions

# Funzioni HASH




Una **funzione hash** è una funzione che converte un vettore arbitrariamente lungo in un vettore di lunghezza fissata:

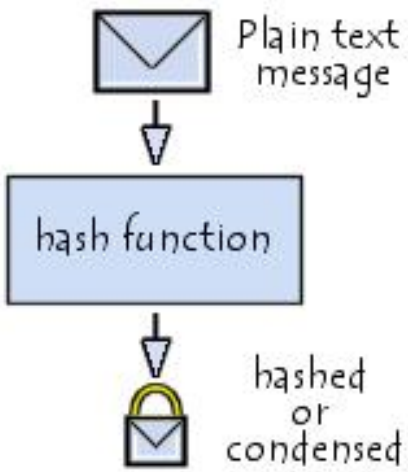
$$H : \mathbb{F}^{\infty} \rightarrow \mathbb{F}^{h}$$

dove $\mathbb{F} = \{ 0, 1 \}$.

Proprieta':

- one-way function
- collision-resistant: $\nexists m \neq m'$ tc $H(m) = H(m')$

# SHA-1




- SHA-1: Security Hash Algorithm [FIPS 180-1]
- dato un messaggio in input di lunghezza massima $2^{64}$ bits, SHA-1 restituisce come output una stringa di 160 bit

input: ''m'illumino di immenso''

output:

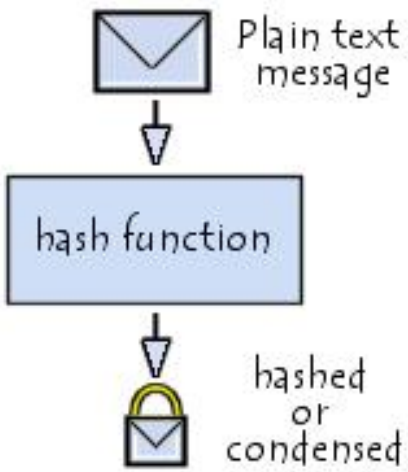
04DEC8C39C14B4E5AB284EE204C81D58F1A59936

input: ''Roma''

output:

DE5429D6F4FA2C86427A50757791DE88A0B75C85

# SHA-1




- SHA-1: Security Hash Algorithm [FIPS 180-1]
- dato un messaggio in input di lunghezza massima $2^{64}$ bits, SHA-1 restituisce come output una stringa di 160 bit

input: ''mi illumino di immenso''

output:

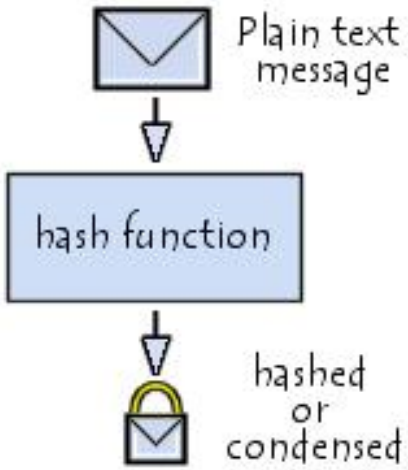
666BCFA1CC6D6580F316AF077B85B9DE34055A57

input: ''roma''

output:

A6B6EA31C49A8E944EFE9ECBC072A26903A1461A

# SHA-2




- SHA-256: Security Hash Algorithm [FIPS 180-2]
- dato un messaggio in input di lunghezza massima $2^{64}$ bits, SHA-256 restituisce come output una stringa di 256 bit

SHA-256 e' ritenuto nettamente piu' sicuro di SHA-1

# Randomicita'

E' essenziale che le chiavi siano generate random, perche' altrimenti un attaccante potrebbe ricrearle.

# Randomicita'

E' essenziale che le chiavi siano generate random, perche' altrimenti un attaccante potrebbe ricrearle.

In realta', il problema avrebbe peso maggiore se ci fosse un grande numero di chiavi da selezionare, ma ogni gestore ha la sua, quindi il problema e' abbastanza limitato, anche se non da trascurare.

# Creazione randomicità

Esistono principalmente tre modi per ottenere una chiave random:

# Creazione randomicità

Esistono principalmente tre modi per ottenere una chiave random:

- utilizzare sorgenti random naturali (es. campionamento del campo elettrico e della voce, rilevamento fenomeni quantistici)

# Creazione randomicità

Esistono principalmente tre modi per ottenere una chiave random:

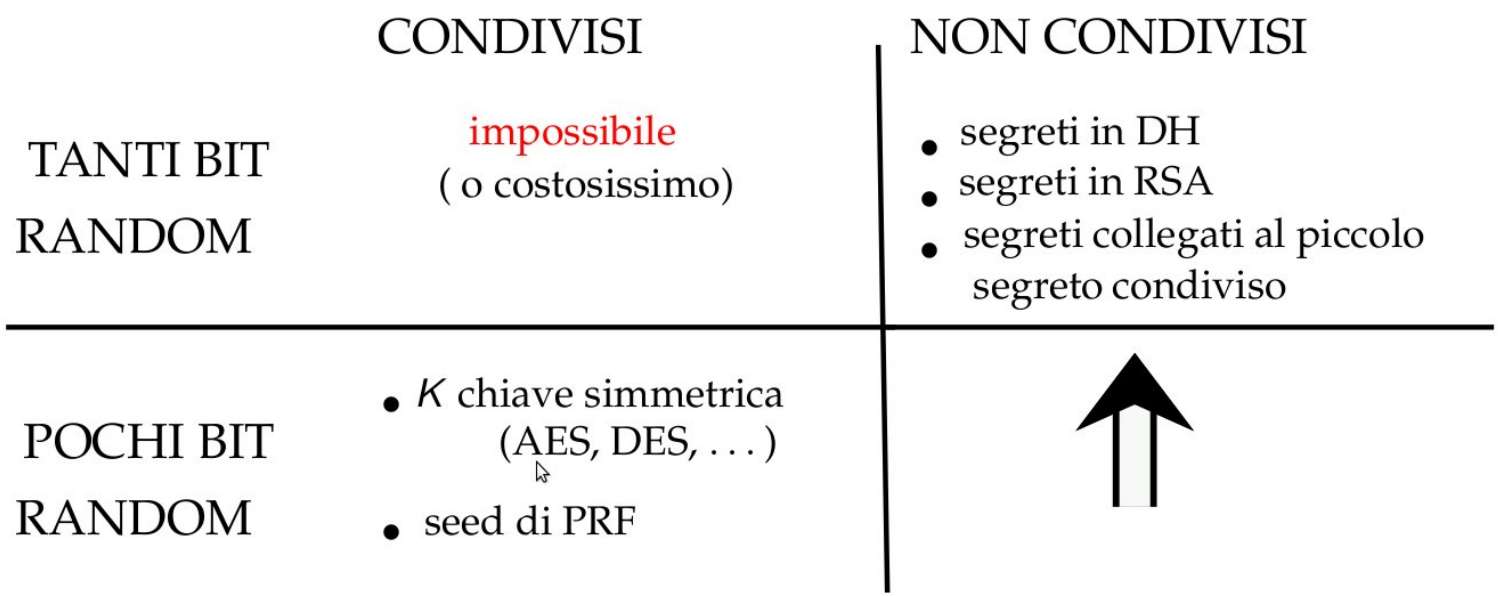
- utilizzare sorgenti random naturali (es. campionamento del campo elettrico e della voce, rilevamento fenomeni quantistici)
- utilizzare algoritmi matematici pseudo-random (PRF)

# Creazione randomicità

Esistono principalmente tre modi per ottenere una chiave random:

- utilizzare sorgenti random naturali (es. campionamento del campo elettrico e della voce, rilevamento fenomeni quantistici)
- utilizzare algoritmi matematici pseudo-random (PRF)
- utilizzare una commistione di sorgenti random e pseudo-random

# Quanti random?




PRF: seed$\rightarrow$ 101000111 . . . . . .

# Crittografia Asimmetrica

# Algoritmi di crittografia asimmetrica

La sicurezza dell'algoritmo di cifratura asimmetrica utilizzato si basa sulla complessita' del problema matematico che sta alla base.

I principali algoritmi sono:

- DSA (Digital Signature Algorithm), basato sul **Discrete Logarithm Problem** (DLP)
- RSA, basato sull' **Integer Factorization Problem** (IFP)

# DSA

# DLOG - Discrete Logarithm

Il problema del Logaritmo Discreto su $\mathbb{F}_p$ è il seguente:

$$\text{dati } a, b \in \mathbb{F}_p, \quad \text{trovare } x \in \mathbb{F}_p \text{ t.c. } a^x = b$$

$$[n = \log_2(p)]$$

# DLOG - Discrete Logarithm

Il problema del Logaritmo Discreto su $\mathbb{F}_p$ è il seguente:

$$\text{dati } a, b \in \mathbb{F}_p, \quad \text{trovare } x \in \mathbb{F}_p \text{ t.c. } a^x = b$$

$$[n = \log_2(p)]$$

Per quanto il problema sia apparentemente semplice, si conoscono solo attacchi subesponenziali.
Spesso il migliore è l'algoritmo Function Field Sieve, che gira con complessità

$$2^{n^{\frac{1}{3}}(\log_2(n))^{\frac{2}{3}}}$$

$k \sim \sqrt[3]{n}$, ma per $80 \leq k \leq 256$ si ha $n \sim k^{1.7}$.

# Descrizione di DSA

Descriviamo la struttura dell'algoritmo. Alice  vuole trasmettere il messaggio $m$ a Bob  e insieme ad esso la sua firma digitale.

Occorre una coppia di chiavi asimmetriche.

- Alice deve creare una chiave pubblica $k_{p_a} = y$ ed una privata $k_{s_a} = x$ stando attenta a non divulgare $x$
- Alice trasmette $y$ a Bob

# Descrizione di DSA: generazione delle chiavi

Come crea Alice la chiave pubblica e privata?

- Alice sceglie due numeri primi $p$ e $q$, tali che $p = qz + 1$ per un qualche numero intero $z$,
- Alice sceglie un numero $h$ tale che $1 < h < p - 1$ e $g = h^z \mod p > 1$
- Alice genera un numero casuale $x$ tale che $0 < x < q$.
- Alice calcola $y = g^x \mod p$
- solo Alice  conosce la chiave segreta $x$ mentre $y$ è pubblica
- I parametri $(p, q, g)$ sono pubblici e possono essere condivisi da diversi utenti.

# Descrizione di DSA: generazione della firma

Come avviene la firma del messaggio?

- Alice genera un numero casuale $k$ tale che $0 < k < q$ e calcola $r = (g^k \mod p) \mod q$
- Alice calcola $s = (k^{-1}(H(m) + x * r)) \mod q$ dove $H(m)$ e' una funzione di hash SHA-d applicata al messaggio $m$
- Nel caso in cui $r = 0$ o $s = 0$ bisogna ricalcolare la firma
- Altrimenti Alice trasmette il messaggio $m$ e la firma $(r, s)$ a Bob

# Descrizione di DSA: verifica della firma

Come avviene la verifica della firma?

- Bob rifiuta la firma se non sono soddisfate le condizioni $0 < r < q$ e $0 < s < q$
- Bob calcola $w = s^{-1} \mod q$
- Bob calcola $u_1 = (H(m) * w) \mod q$
- Bob calcola $u_2 = (r * w) \mod q$
- Bob calcola $((g^{u_1} * y^{u_2}) \mod p) \mod q$
- La firma e' verificata se $v = r$




# RSA

# IFP - Integer Factorization Problem

Il problema della fattorizzazione degli interi e' il sequente.
Siano $p$ e $q$ numeri primi, sia $N = pq$, dato $N$ trovare $p$ e $q$.

$$[n = \log_2(N)]$$

# IFP - Integer Factorization Problem

Il problema della fattorizzazione degli interi e' il sequente.
Siano $p$ e $q$ numeri primi, sia $N = pq$, dato $N$ trovare $p$ e $q$.

$$[n = \log_2(N)]$$

Il miglior algoritmo è una versione del General Number Field Sieve, che gira con complessità

$$2^{n^{\frac{1}{3}}(\log_2(n))^{\frac{2}{3}}}$$

$k \sim \sqrt[3]{n}$, ma per $80 \leq k \leq 256$ si ha $n \sim k^{1.7}$.

# Descrizione di RSA

Possiamo così passare a descrivere il cifrario. Alice  vuole trasmettere il messaggio $m$ a Bob  .

- Bob deve creare una chiave pubblica $k_{p_b} = (N, e)$ ed una privata $k_{s_b} = (N, d)$ stando attento a non divulgare $d$
- Bob trasmette $(N, e)$ ad Alice

# Descrizione di RSA *

Come crea Bob la chiave pubblica e privata?

- Bob sceglie due numeri primi $p$ e $q$, molto grandi e li moltiplica $N = p \cdot q$
- Bob sceglie un numero $e$, coprimo con $\varphi(N)$ e più piccolo di $(p-1)\cdot(q-1)$
- Bob calcola $d$ tale che sia l'inverso moltiplicativo di $e$ mod $\varphi(N)$, cioé $(x^d)^e \equiv x \bmod N$, qualunque $x$.
- solo Bob  conosce la chiave segreta $(N, d)$ mentre $(N, e)$ è pubblica

# Descrizione di RSA *

Come avviene la cifratura e decifratura?

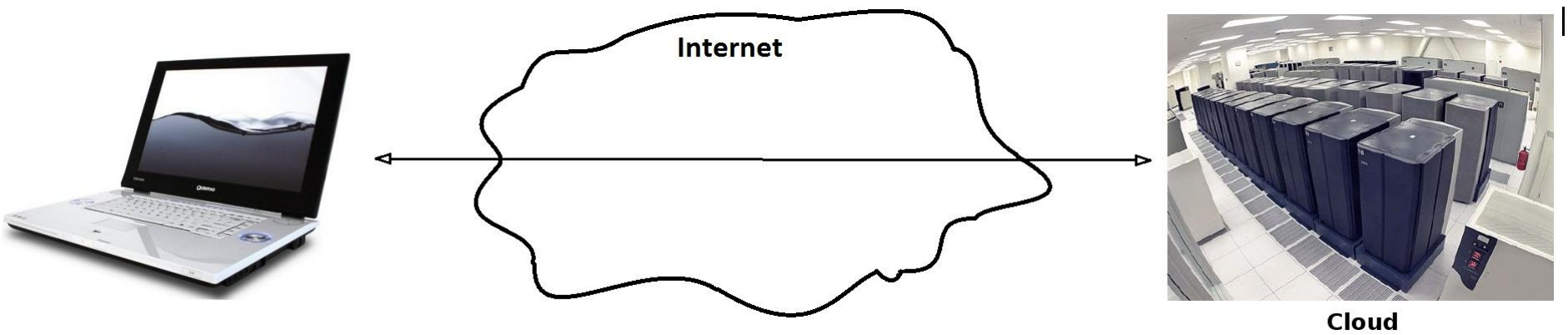
- Alice calcola $c = m^e \mod N$
- Alice trasmette $c$ a Bob
- Bob riceve $c$ e lo decripta calcolando:

$$c^d \equiv m \mod N$$

# Network Security

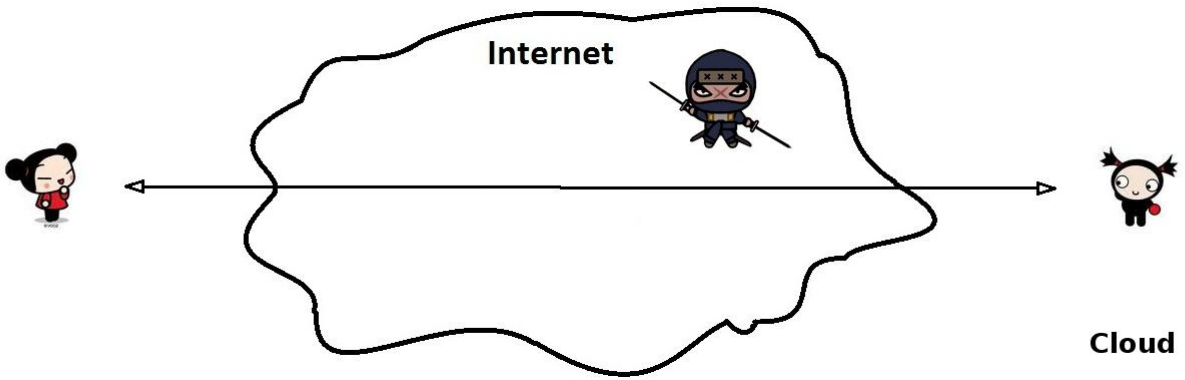
# SSL/TLS

É il protocollo usato a livello di comunicazione tra browser e server per la trasmissione sicura dei dati.



# SSL/TLS

É il protocollo usato a livello di comunicazione tra browser e server per la trasmissione sicura dei dati.



# Fasi SSL/TLS

1. $A$ e $B$ si mettono d'accordo sulla crittografia da usare (Cipher Suite)

# Fasi SSL/TLS

1. $A$ e $B$ si mettono d'accordo sulla crittografia da usare (Cipher Suite)
2. $A$ si crea per conto suo dei bit segreti

# Fasi SSL/TLS

1. $A$ e $B$ si mettono d'accordo sulla crittografia da usare (Cipher Suite)
2. $A$ si crea per conto suo dei bit segreti
3. $B$ fa lo stesso

# Fasi SSL/TLS

1. $A$ e $B$ si mettono d'accordo sulla crittografia da usare (Cipher Suite)
2. $A$ si crea per conto suo dei bit segreti
3. $B$ fa lo stesso
4. $A$ e $B$ scambiano una chiave segreta $K$ (segreto condiviso) tramite RSA o DH

# Fasi SSL/TLS

1. $A$ e $B$ si mettono d'accordo sulla crittografia da usare (Cipher Suite)
2. $A$ si crea per conto suo dei bit segreti
3. $B$ fa lo stesso
4. $A$ e $B$ scambiano una chiave segreta $K$ (segreto condiviso) tramite RSA o DH
5. $A$ e $B$ usano $K$ con AES, 3DES o DES per crittare la conversazione

# Possibili Debolezze

# Forgery

Due possibili strade percorribili:

- rompere la chiave privata
- ottenere un hash finto, ossia trovare un $m' \neq m$ tc $H(m') = H(m)$

Poiche' RSA e' l'algoritmo maggiormente usato nelle PEC, ci soffermiamo sui possibili attacchi ad esso.

# Chiavi forti di RSA

Come si può ricavare la chiave privata da quella pubblica?

Apparentemente ci sono 3 alternative:

1. fattorizzare $N$
2. Calcolare $\varphi(N)$
3. ricavare il valore $d$ direttamente dalla chiave pubblica $(N, e)$

Si può facilmente dimostrare che queste tre metodi sono equivalenti:

$$(1) \Leftrightarrow (2) \Leftrightarrow (3)$$

La resistenza di una chiave forte e' $n \sim k^{1.7}$.

# Chiavi forti di RSA: un caso pratico

Usando l'algoritmo GNFS si riescono a fattorizzare numeri $N$ dell'ordine di $2^{768} \sim 10^{231}$.

Spendendo un milione di euro per dotarsi di un cluster potente, si potrebbe rompere una chiave RSA-768 (cioé $N$ è dell'ordine di $2^{768}$) ogni 6 mesi (in media).

Lo sforzo per rompere una chiave RSA-1024 e' circa mille forte quello per rompere una chiave di RSA-768, quindi chiavi RSA-1024 o superiori (purché non siano chiavi deboli) sono al momento fuori dalla portata di un attacco pratico.

# Paragone con AES-256

Il miglior attacco noto a AES256 (che recupera la chiave) ha un costo di $2^{254}$ crittazioni. Supponendo che

- un core medio riesca a eseguire $2^{50}$ crittazioni al secondo,
- esistano 1000 core ogni abitante del pianeta, compresi neonati e le persone che vivono nella giungla amazzonica,
- al mondo ci siano 20 miliardi di persone,

# Paragone con AES-256

Il miglior attacco noto a AES256 (che recupera la chiave) ha un costo di $2^{254}$ crittazioni. Supponendo che

- un core medio riesca a eseguire $2^{50}$ crittazioni al secondo,
- esistano 1000 core ogni abitante del pianeta, compresi neonati e le persone che vivono nella giungla amazzonica,
- al mondo ci siano 20 miliardi di persone,

Allora TUTTA la potenza del mondo unita dovebbe lavorare senza sosta per $10^{31}$ MILIARDI di anni.

# Chiavi Deboli

Spesso si pensa che piu' una chiave e' lunga, piu' questa e' sicura.
In realta' non e' vero.

# Chiavi Deboli

Spesso si pensa che piu' una chiave e' lunga, piu' questa e' sicura.
In realta' non e' vero.
Le chiavi deboli sono molto piu' pericolose.

# Chiavi Deboli

Spesso si pensa che piu' una chiave e' lunga, piu' questa e' sicura.
In realta' non e' vero.
Le chiavi deboli sono molto piu' pericolose.

É facile rompere il sistema se ad esempio:

- se $m$ ed $e$ sono così piccoli che $m^e < N$; allora risulta facile trovare la radice $e$-esima di $c$, poichè $c = m^e$
- se $p - q < N^{\frac{1}{4}}$, ci sono algoritmi veloci per trovare $p$ e $q$
- se $p - 1$ o $q - 1$ hanno solo fattori piccoli, $N$ si fattorizza velocemente con l'algoritmo di Pollard $p - 1$

# Chiavi Deboli

Spesso si pensa che piu' una chiave e' lunga, piu' questa e' sicura.
In realta' non e' vero.
Le chiavi deboli sono molto piu' pericolose.

É facile rompere il sistema se ad esempio:

- se $m$ ed $e$ sono così piccoli che $m^e < N$; allora risulta facile trovare la radice $e$-esima di $c$, poichè $c = m^e$
- se $p - q < N^{\frac{1}{4}}$, ci sono algoritmi veloci per trovare $p$ e $q$
- se $p - 1$ o $q - 1$ hanno solo fattori piccoli, $N$ si fattorizza velocemente con l'algoritmo di Pollard $p - 1$

Purtroppo molte implementazioni dell'algoritmo RSA non controllano l'eventualità di avere una chiave debole.

# Chi genera le chiavi?

Dispositivi chiamati HSM (Hardware Security Module) sono solitamente chiamati a generare le chiavi.

# Chi controlla le chiavi?

Aruba:

```
30:82:01:22:30:0D:06:09:2A:86:48:86:F7:0D:01:01:01:05:00:03:82:01:0F:00:30:82:01:0A:02:82:01:0
1:00:B2:BB:6E:3A:CC:22:F2:9D:90:C7:01:F1:9F:6D:E8:35:5B:1C:5B:FB:21:27:7C:EB:DB:F3:27:18:E9:DB
:E0:F3:62:9E:B5:D9:19:CB:87:86:60:B8:B9:BE:98:EA:96:89:10:02:87:20:30:5D:62:C8:F7:1E:6D:48:13:
38:D2:1E:3F:A2:F0:96:12:25:41:85:F6:16:C7:F7:A5:F0:CB:AC:29:0D:B6:D6:B5:DF:AF:DB:55:91:95:F3:1
1:A5:AE:FB:80:42:DC:0A:63:1F:80:1F:F2:54:6E:23:9D:73:46:2D:A4:68:92:6E:3C:98:62:AB:AF:DE:78:44
:62:3C:94:1D:88:35:C2:2C:27:87:C5:17:7E:B4:F8:58:53:55:70:13:A8:6E:BD:76:B8:A8:05:5D:FD:AB:58:
33:FF:56:2E:C4:2E:F9:15:36:87:F2:AD:21:DE:BA:F1:21:4B:3A:6C:C0:C4:0F:1D:5B:80:84:E5:DF:AD:22:4
9:DA:31:E3:3B:FA:BE:75:7E:4D:13:FC:7F:47:6D:A9:A0:99:75:BC:36:F7:8E:E0:78:83:06:BE:B2:9A:2D:DA
:8B:15:E8:55:86:77:D9:DD:BE:AD:FB:D6:DE:A1:92:B2:00:41:DA:A0:03:68:B4:E7:56:D4:1F:2A:0F:BC:01:
8F:F7:6B:1C:2E:66:F3:02:03:01:00:01
```

Infocert:

```
30:82:01:22:30:0D:06:09:2A:86:48:86:F7:0D:01:01:01:05:00:03:82:01:0F:00:30:82:01:0A:02:82:01:0
1:00:CE:AA:75:EB:4C:BD:7C:2F:87:38:03:82:0B:D7:99:BE:72:45:D9:5D:D0:65:47:48:79:1B:2C:8C:4D:01
:D6:18:24:D9:85:07:6F:74:F9:05:69:B3:12:53:D3:FC:F1:C0:5A:7E:0B:A8:03:45:A0:0A:30:9D:B8:25:A5:
BC:CA:D3:79:EE:55:E0:76:51:C8:B6:08:4B:8D:36:F0:AE:B6:CA:CD:C6:91:ED:6F:06:58:67:60:BD:AE:1E:E
1:D7:DD:32:C4:DD:7A:32:0B:2C:FE:62:6F:7F:20:3F:99:9C:0D:C8:FF:A4:FB:90:26:4D:88:CD:77:86:B8:30
:75:EC:B3:F0:8A:69:E2:F5:73:37:E7:45:0A:35:6D:06:BD:BD:8F:95:A9:EB:14:AE:CF:C8:A0:A2:3B:A6:07:
71:12:E5:19:BF:E7:D0:0D:89:42:63:34:72:6A:D5:A7:F8:E7:6C:4A:24:02:9B:4A:BC:A0:F3:7A:1F:11:46:3
E:4C:87:0D:8C:A3:02:78:57:A5:DE:37:24:B9:E9:67:C5:8B:65:3F:0B:18:60:0A:99:FB:A9:C7:5F:49:B6:3F
:76:DF:27:0C:4A:FC:FC:CD:06:1B:25:83:A6:AD:5A:B0:69:20:B1:64:78:21:2B:07:C5:33:7E:72:28:6D:0E:
83:94:66:16:07:B6:69:02:03:01:00:01
```

Namirial:

```
30:82:01:22:30:0D:06:09:2A:86:48:86:F7:0D:01:01:01:05:00:03:82:01:0F:00:30:82:01:0A:02:82:01:0
1:00:A8:CD:0E:55:A2:27:33:60:C7:90:6E:83:66:80:5C:C5:FD:17:67:71:21:36:81:78:7A:CE:AB:A3:7F:51
:3B:93:C4:5B:01:13:76:06:B7:E5:6E:8E:46:26:3F:8C:99:7B:1D:E4:68:AA:DF:E2:BA:B6:1F:FC:6E:BF:80:
7D:B7:FC:3C:37:A3:35:3F:41:87:8E:61:FD:31:AE:42:FC:D1:42:BD:60:F7:48:DA:E3:61:A8:73:E2:6E:26:A
3:87:9C:B3:CB:8F:4D:74:FD:31:AA:16:E4:3E:3E:B4:D8:FD:DD:3F:F4:3C:3C:DA:6F:86:27:05:6B:8C:9F:5A
:B3:E1:54:5F:9F:D6:DF:EC:4D:D5:2E:A7:21:7F:E3:31:76:A9:AB:48:37:FF:52:A8:F5:64:57:8F:9C:9E:07:
A6:79:94:E5:89:AB:1D:EA:D2:18:21:D1:23:4F:5F:4B:B1:CF:CD:45:17:3E:9A:4D:7D:5E:F6:02:7C:9F:96:E
9:51:5C:BF:65:94:DD:80:EB:21:93:DA:27:A9:37:C8:E7:24:06:32:D5:21:6C:2A:68:62:83:A9:F5:C4:B0:A0
:CD:E7:BD:9E:46:18:9E:AE:BA:3E:16:22:8A:E6:9F:B5:2A:ED:48:25:8B:73:5C:5D:F2:EF:AE:E6:34:1B:18:
B5:18:FF:CD:10:5C:49:02:03:01:00:01
```

Grazie dell'attenzione.