Appunti per Orale di Crittografia

Simone Ianniciello

A.A. 2020/2021

Contents

Chapter 1

Domande per la prof

- Pag 148 : $n_A = n - 1 < n, n_B = 1 < n \implies n_A n_B B = nB = \mathcal{O}$

Chapter 2

Chapter 3

Chapter 4

Il ruolo del caso

4.1 Il significato algoritmico della casualità

- Casualità secondo Kolmogorov
  - $\mathcal{K}$: Complessità
  - S_i : Algoritmo
  - h : Sequenza "casuale"
  - p : Rappresentazione binaria dell'algoritmo
  - $\mathcal{K}_{S_i}(h) = min\{|p| : S_i(p) = h\}$
- Una sequenza e' casuale se
  - $\mathcal{K}(h) \geq |h| - \lceil log_2(h) \rceil$

4.2 Generatori di numeri pseudo-casuali

- Generatore di numeri pseudo-casuali: algoritmo che parte da un piccolo valore iniziale detto seme e genera una sequenza arbitrariamente lunga di numeri.
- Proprietà di un generatore:
  - **Frequenza** : Verifica se i diversi elementi appaiono in S approssimativamente lo stesso numero di volte
  - **Poker** : Verifica la equidistribuzione di sottosequenze di lunghezza arbitraria ma prefissata
  - **Autocorrelazione** : he verifica il numero di elementi ripetuti a distanza prefissata
  - **Run** : verifica se le sottosequenze massimali contenenti elementi tutti uguali hanno una distribuzione esponenziale negativa

  - **Prossimo bit** : Non esiste un algoritmo polinomiale in grado di predire l'$(i + 1)$-esimo bit della sequenza conoscendo i bit precedenti

- Generatore BBS : $x_i \leftarrow (x_{i-1})^2 \bmod n \wedge b_i = 1 \Leftrightarrow x_{m-i}$ e' dispari

4.3 Algoritmi randomizzati

- *Las Vegas* : Risultato **sicuramente** corretto in tempo **probabilmente** breve
- *Monte Carlo* : Risultato **probabilmente** corretto in tempo **sicuramente** breve
- *Miller e Rabin* : Algoritmo di tipo *Monte Carlo* per il controllo di un numero primo
  - n : Valore da controllare
  - z : Intero tale che $z = \dfrac{N-1}{2^w}$
  - y : $y \in [2, N-1]$
  - P_1 : $mcd(N, y) = 1$
  - P_2 : $(y^z \bmod N = 1) \vee (\exists i, 0 \leq i \leq w-1 : y^{2^i z} \bmod N = -1)$
  - $(P_1 \wedge P_2) = false$: N e' sicuramente composto
  - $(P_1 \wedge P_2) = true$: N e' primo con probabilità $p \geq \dfrac{3}{4}$

Chapter 5

Cifrari storici

5.1 Cifrari a sostituzione monoalfabetica

Ogni lettera l del messaggio viene trasformata in una lettera c non dipendente dalla posizione o dal contesto di essa.

- $pos(n)$: Posizione della lettera n
- p : Lunghezza dell'alfabeto
- Cifrario affine : $pos(y) = a \cdot pos(x) + b \bmod p$
  - $mcd(a, p) = 1$
  - $b \in [0, p-1]$
- Cifrario di Cesare : Cifrario affine con $k = (1, 3)$
- Attacchi
  - Brute-force
  - Analizzando la frequenza delle lettere e dei q-grammi del crittogramma e confrontandole con quelle della lingua si puo' risalire velocemente al messaggio e alla chiave

5.2 Cifrari a sostituzione polialfabetica

La stessa lettera del messaggio corrisponde a lettere diverse del crittogramma

- Cifrario di Alberti : Simile al cifrario affine ma permette il cambio della chiave dinamico

- Cifrario di Vigenere
  - $offset : x = m[offset]$
  - $i = offset \bmod |k|$
  - $pos(y) = (pos(x) + pos(k[i])) \bmod p$
  - Attacco : Dato che la chiave viene ripetuta con periodo $|k|$, si hanno $|k|$ sotto-messaggi cifrati con sistema monoalfabetico e si possono usare gli stessi metodi di esso

5.3 Cifrari a trasposizione

Le lettere del messaggio vengono permutate secondo una legge dettata dalla chiave.

- Permutazione semplice : Fissato un h e una permutazione π degli interi $\leq h$, il processo di crittazione consiste dividere il messaggio in blocchi di h lettere e permutare ciascuno di essi in accordo con π; Il numero di chiavi e' uguale a $h! - 1$
- Permutazione di colonne
  - $k = \langle c, r, \pi \rangle$
  - π e' una permutazione di c
  - Si cifrano le righe come in *permutazione semplice*
  - Si costruisce c leggendo la tabella per colonne $\downarrow\rightarrow$
- Cifrari a griglia
  - Il crittogramma e' scritto in una tabella $q \times q$
  - La chiave e' una scheda perforata con con $\frac{q}{4}$ celle trasparenti che permette di ricostruire il messaggio tramite quattro rotazioni della griglia sul crittogramma e leggendo i caratteri $\rightarrow\downarrow$
  - Il numero di chiavi possibili G e' $G = 2^{q^2/2}$

Chapter 6

Cifrari perfetti

6.1 Definizione

- $\mathcal{P}r(M = m)$: Probabilita' che m sia il messaggio che deve essere inviato
- $\mathcal{P}r(M = m|C = c)$: Probabilita' che il messaggio originale fosse m dato il crittogramma c in transito
- Un cifrario e' perfetto se :
  - $\forall m \in \text{Msg}, c \in \text{Critto} : \mathcal{P}r(M = m|C = c) = \mathcal{P}r(M = m)$
  - Cioe' m e c sono totalmente scorrelati tra loro percio' la conoscenza complessiva di un crittoanalista non cambia dopo che ha osservato un crittogramma sul canale
    - In un cifrario perfetto il numero di chiavi deve essere $\geq$ dei messaggi possibili
    - N_m : Numero dei messaggi possibili
    - N_k : Numero di chiavi
    - $N_k \geq N_m$: Se non fosse cosi', dato un crittogramma c, esisterebbe un messaggio m' non generabile tramite la decrittazione di c con tutte le chiavi del sistema
    - $\mathcal{P}r(M = m'|C = c) = 0 \neq \mathcal{P}r(M = m)$

6.2 One-Time Pad

- Si assume che i messaggi m, le chiavi k, e i crittogrammi c siano codificati come sequenze binarie
- $\mathcal{C}(m, k) = c = m \oplus k$
- $\mathcal{D}(c, k) = m = c \oplus k$

- Gen. chiave : Si costruisce una sequenza $k = k_1k_2 \dots k_n : n \geq |m|$, $\mathcal{P}r(k_i = 0) = \mathcal{P}r(k_i = 1) = 1/2$
- Cifratura : Dati $m = m_1m_2 \dots m_n$ e $k = k_1k_2 \dots k_n \implies c = c_1c_2 \dots c_n$ con $c_i = m_i \oplus k_i$
- Decifrazione : Identico alla cifratura ma con c e m invertiti
  - Il cifrario One-Time Pad si considera perfetto se
  - *Tutti i messaggi hanno la stessa lunghezza n* : Altrimenti la lunghezza del crittogramma darebbe informazioni utili al crittoanalista
  - *Tutte le sequenze di n bit sono messaggi possibili* : Non propriamente vero
- Dimostrazione di perfezione di One-Time Pad
  - $\mathcal{P}r(M = m|C = c) = \frac{\mathcal{P}r(M=m, C=c)}{\mathcal{P}r(C=c)}$
  - Per definizione di `XOR` chiavi diverse generano crittogrammi diversi. Quindi fissato m abbiamo $\mathcal{P}r(C = c) = (1/2)^n$ quindi gli eventi $\{M = m\}$ e $\{C = c\}$ sono indipendenti
  - $\mathcal{P}r(M = m, C = c) = \mathcal{P}r(M = m) \times \mathcal{P}r(C = c)$
  - $\mathcal{P}r(M = m|C = c) = \frac{\mathcal{P}r(M=m) \times \cancel{\mathcal{P}r(C=c)}}{\cancel{\mathcal{P}r(C=c)}} = \mathcal{P}r(M = m)$

6.3 Generazione della chiave

- Per definizione di `XOR` la chiave non puo' essere riutilizzata perche' dati
  - $c' = m' \oplus k$
  - $c'' = m'' \oplus k$
  - $\forall i \in [0, n] \implies c'_i \oplus c''_i = m'_i \oplus m''_i$: Cio significa che due porzioni identiche di m' e m'' corrispondono a un tratto di $c'_i \oplus c''_i$ di tutti zeri
- Metodi di generazione
  - **Generatore pubblico, seme privato** : I due partner devono scambiarsi soltanto il seme ma espone il cifrario ad un attacco esauriente sui semi possibili
  - **Generatore e seme privati** : Richiede che i partner si accordino su un generatore di loro definizione per evitare un attacco esauriente sui generatori conosciuti

- Approssimazioni e considerazioni
  - Nella realta' dei fatti, non tutti i 2^n messaggi sono messaggi possibili; Questo perche' si assume che il messaggio sia codificato in linguaggio naturale e deve seguire determinate regole.
  - Nei linguaggi naturali il numero di messaggi validi di lunghezza n e' circa α^n con $\alpha < 2$ variabile a seconda della lingua

Chapter 7

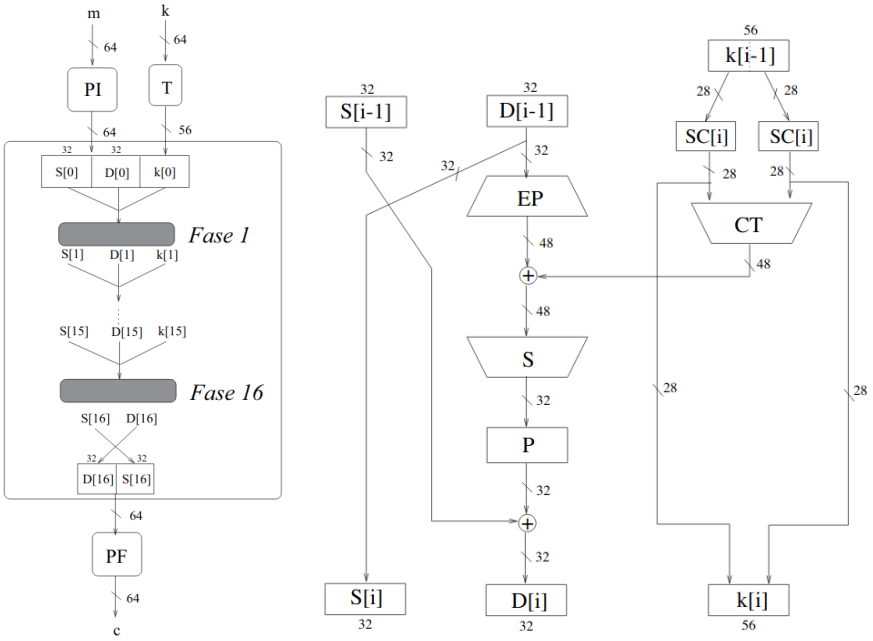
Il cifrario simmetrico standard

7.1 Criteri di Shannon

- **Diffusione** : Permutazioni ed espansioni del messaggio
- **Confusione** : Combinazione e compressione dei bit del messaggio e e della chiave

7.2 Il cifrario DES

- Il messaggio viene diviso in blocchi da 64 bit
- La chiave e' composta da 8 Byte in cui ogni ottavo bit e' di parita'
- La cifratura e decifrazione del messaggio avviene attraverso 16 round
- Dalla chiave vengono create 16 sottochiavi $k[0], k[1], \ldots, k[15]$ dette chiavi locali
- La decifrazione si esegue invertendo l'ordine delle chiavi locali
- Tutte le operazioni del DES sono lineari ad eccezione della S-Box
- Tutte le espansioni e compressioni del DES sono progettate in modo e maniera da garantire l'utilizzo di tutti i bit della chiave



7.2.1 Attacchi

- Proprieta' del complemento
  - $\mathcal{C}_{DES}(m,k) = c \implies \mathcal{C}_{DES}(\overline{m},\overline{k}) = \overline{c}$
  - Procurandosi alcune coppie $\langle m, c_1\rangle$ e $\langle \overline{m}, c_2\rangle$ si puo' effettivamente dimezzare il numero di chiavi da testare per un attacco esaustivo
  - Infatti se testando una chiave k risulta $\mathcal{C}_{DES}(m,k) = c_1$ allora k probabilmente e' la chiave
  - Se invece risulta che $\mathcal{C}_{DES}(m,k) = \overline{c_2}$ allora la chiave potrebbe essere $\overline{k}$

- Crittoanalisi differenziale
  - Si supponga di riuscire ad ottenere 2^{47} coppie $\langle m, c$ con m scelti dal crittoanalista
  - Si possono quindi analizzare i crittogrammi ottenuti per assegnare probabilitá diverse a varie chiavi
  - La chiave originale dovrebbe quindi emergere come quella con probabilitá piú alta

- Crittoanalisi lineare
  - Date 2^{43} coppie $\langle m, c\rangle$ si possono inferire alcuni bit della chiave tramite un'approssimazione lineare della funzione di cifratura e di ricavare gli altri tramite attacco esauriente

7.2.2 Variazioni del DES

- **Scelta indipendente delle sottochiavi** : Cio porta la lunghezza della chiave da 56 a 768 bit. In questo caso un attacco con analisi differenziale richiede 2^61 coppie messaggio crittogramma
- **3TDEA** : Si scelgono due chiavi k_1, k_2 e si procede come $\mathcal{C}_{DES}(\mathcal{D}_{DES}(\mathcal{C}_{DES}(c, k_1), k_2), k_3)$
- **2TDEA** : Come 3TDEA con $k_3 = k_1$

7.3 AES

- Cifrario simmetrico
- Il messaggio é diviso in blocchi da 128 bit
- Le chiavi sono da 128, 192 o 256 bit
- In base alla lunghezza della chiave il processo viene iterato 10 (128bit), 12 (192bit), o 14 (256bit) volte
- Le operazioni di ogni fase sono:
  - **Op1** : Ogni Byte della matrice B é trasformato tramite una S-Box
  - **Op2** : La matrice ottenuta é permutata tramite shift ciclici sulle righe
  - **Op3** : Le colonne risultanti vengono trasformate tramite una operazione algebrica
  - **Op4** : Ogni byte della matrice viene messo in `XOR` con un Byte della chiave locale per quella fase

7.4 Cifrari a composizione di blocchi

- Per la struttura dei cifrari appena descritti, blocchi identici del messaggio vengono trasformati in blocchi identici del crittogramma.
- Per ovviare a questo problema ci si puó affidare ai cifrari a composizione di blocchi

- Il messaggio viene diviso in blocchi $m = m_1 m_2 \dots m_s$ di b bit
- Se m_s contiene $r < b$ bit lo si completa aggiungendo la sequenza $10000\dots$ di lunghezza $b - r$, altrimenti si aggiunge un nuovo blocco $m_{s+1} = 10000$
- Sia c_0 una sequenza di b bit scelta in modo casuale
- Allora $c_i = \mathcal{C}(m_i \oplus c_{i-1}, k)$ e $m_i = c_{i-1} \oplus \mathcal{D}(c_i, k)$

Chapter 8

Crittografia a chiave pubblica

8.1 Funzioni One-Way Trap-Door

- Fattorizzazione
  - Calcolare Il prodotto $n = pq$ e polinomialmente facile
  - Ricavare p e q dato n invece richiede tempo esponenziale perchè bisogna *provarli tutti*
- Calcolo della radice in modulo
  - Calcolare $y = x^z \bmod s$ richiede tempo polinomiale grazie al sistema delle successive esponenziazioni
  - Se s non é primo calcolare $x = \sqrt[z]{y} \bmod s$ richiede tempo esponenziale
  - Se peró $mcd(x, s) = 1$ e si conosce $v = z^{-1} \bmod \Phi(s)$ si ha $y^v \bmod s = x^{zv} \bmod s = x^{1+k\Phi(s)} \bmod s = x \bmod s$
- Calcolo del logaritmo discreto
  - Dati x, y, s interi si richiede di trovare z tale che $y = x^z \bmod s$
  - La soluzione a questo problema esiste se s é primo e x é un generatore di $\mathcal{Z}_s^*$
  - Esiste un artificio piuttosto complesso per introdurre una trap-door in questa funzione

8.2 Pregi e difetti dei cifrari a chiave pubblica

- Pregi
  - Dati n utenti connessi a un sistema, il numero complessivo di chiavi e' $2n$ anzichè $\frac{n(n-1)}{2}$
  - Non é richiesto alcuno scambio segreto di chiavi
- Difetti
  - Il sistema é esposto ad attacchi di tipo *chosen plain-text*
  - Questi sistemi sono molto più lenti dei cifrari simmetrici

8.3 Il cifrario RSA

- Creazione delle chiavi
  - Si scelgono p e q primi molto grandi
  - $n = pq$
  - $\Phi(n) = (p-1)(q-1)$
  - Si sceglie e : $e < \Phi(n) \wedge mcd(e, \Phi(n)) = 1$
  - Si calcola $d = e^{-1} \bmod \Phi(n)$ Tramite `Euclide Esteso`
  - $k[pub] = \langle e, n \rangle$
  - $k[prv] = \langle d \rangle$
- Per cifrare un messaggio m esso deve essere codificato come un intero $m < n$
- m puó essere diviso in blocchi di lunghezza $\lfloor \log_2(m) \rfloor$
- $c = \mathcal{C}(m, k[pub]) = m^e \bmod n$
- $m = \mathcal{D}(c, k[prv]) = c^d \bmod n = m^{ed} \bmod n = m$

8.3.1 Correttezza di RSA

Si distinguono due casi

- p e q non dividono m
  - $mcd(m, n) = 1$
  - Eulero : $m^{\Phi(n)} \equiv 1 \bmod n$
  - $e \times d \equiv 1 \bmod \Phi(n) = 1 + r\Phi(n)$

  - $m^{ed} \bmod n = m^{1+r\Phi(n)} \bmod n = m \times (m^{\Phi(n)})^r \bmod n =$
    $= m \times 1^r \bmod n = m \bmod n$

- $m \mid p \wedge m \nmid q$
  - $m \equiv m^r \equiv 0 \bmod p, (m^r - m) \equiv 0 \bmod p$
  - $\Phi(q) = (q-1)$ perchè q é primo
  - $m^{\Phi(q)} \bmod q \equiv 1 \bmod q$
  - $m^{ed} \bmod q = m^{1+r\Phi(n)} \bmod q = m \times (m^{\Phi(q)})^{r(p-1)} \bmod q = m \bmod q$
  - $m^{ed} \equiv m \bmod q \implies (m^{ed} - m) \equiv 0 \bmod q : (m^{ed} - m) \mid q$
  - $(m^{ed} - m) \equiv 0 \bmod n \implies m^{ed} \bmod n = m \bmod n$

8.4 Attacchi all'RSA

- $|p - q|$ molto piccolo
  - Supponiamo $|p - q|$ piccolo
  - $p+q/2 \approx \sqrt{n}$
  - $\frac{(p+q)^2}{4} - n = \frac{(p-q)^2}{4}$
  - $(p+q/2)^2 \approx n = n_a$
  - $n_a - n = \frac{(p-q)^2}{2^2}$
  - $\frac{(p-q)^2}{2^2} > 0 \implies n_a > n$
  - $w = \frac{(p-q)}{2}$
  - Bisogna trovare $z : z^2 - n = w^2$

- $mcd((p-1),(q-1))$ deve essere piccolo
  (Si scelgono quindi p e $q : mcd(\frac{p-1}{2}, \frac{q-1}{2}) = 1$)

- $e = 1+\Phi(n)/k$ con $m \mid k$ e $mcd(m,n) = 1$
  $c = m^e \bmod n = m \times (m^{\Phi(n)})^{1/k} \bmod n = m \bmod n$

- e molto piccolo
  - Poniamo che e utenti condividano lo stesso valore di e e che ricevano tutti lo stesso messaggio
  - $c_i = m^e \bmod n_i$
  - Si assume che n_i siano tutti coprimi tra loro
  - Per *Teorema Cinese del Resto*
    * $n = n_1 \times n_2 \times \cdots \times n_e$

    * $m' < n$
    * $m' \equiv m^e \bmod n$
  - $m^{e\ (\equiv m')} < n$
  - $m' = m^e$: Perchè m' e m^e sono minori di n
  - $m = \sqrt[e]{m'}$
  - Per mantenere valori di e piccoli senza compromettere la sicurezza basta aggiungere una sequenza di bit diversa alla fine di ogni messaggio (*padding*)

- n uguale per piú utenti
  - Date due chiavi $\langle e_1, n\rangle, \langle e_2, n\rangle : mcd(e_1, e_2) = 1$
  - Tramite `Euclide Esteso` si possono calcolare r e s tali che $e_1 r + e_2 s = 1$ (Fissiamo $r < 0$)
  - Poniamo adesso che si intercettino due crittogrammi c_1, c_2 relativi allo stesso messaggio m diretti ai due utenti attaccati
  - $m = m^{e_1 r + e_2 s} = (c_1^r \times c_2^s) \bmod n = ((c_1^{-1})^{-r} \times c_2^s) \bmod n$
  - Tramite `Euclide Esteso` si calcola $c^{-1} \bmod n$
  - Si puó quindi calcola m in tempo polinomiale

- RSA ha gli stessi problemi (e le stesse soluzioni) del DES per la periodicitá dei blocchi

8.5 Diffie-Hellman per lo scambio di chiavi

- A e B si accordano su un primo p e un generatore g di $\mathcal{Z}_p^*$
- A sceglie $x < p$ casuale e calcola $X = g^x \bmod p$ e spedisce X a B
- B fa lo stesso ($Y = g^y \bmod p$)
- Entrambi calcolano $k[session] = Y^x \bmod p = X^y \bmod p = = g^{xy} \bmod p$
- Se un crittoanalista intercetta X (o Y) dovrebbe calcolarsi il logaritmo discreto per risalire a x (o y)
- L'unico attacco possibile é `MITM`

8.6 Curve ellittiche su campi finiti

- $E_p(a, b) = \{(x, y) \in \mathcal{Z}_p^2 \mid y^2 \bmod p = (x^3 + ax + b) \bmod p\} \cup \{O\}$
- La curva presenta una simmetria rispetto alla retta $y = p/2$

8.7 Diffie-Hellman su curve ellittiche

- A e C si accordano su una curva definita su campo finito, e su un punto B di ordine n molto grande
- n é il piú piccolo intero tale che $nB = O$
- A sceglie un intero $n_A < n$ e genera $P_A = n_A B$ e invia P_A a C
- C fa lo stesso ($P_C = n_C B$)
- Entrambi calcolano $S = n_A P_C = n_c P_A = n_A n_C B$
- $k = x_S \bmod 2^{256}$

8.8 Algoritmo di Koblitz

Questo algoritmo serve a trasformare un messaggio $m < p$ in un punto P_m della curva $E_p(a, b)$

- Si fissa $h \mid (m + 1)h < p$
- Si prova ogni $i \in [0, h)$ si calcola $x_i = mh + i$
- Se esiste la radice quadrata di $y_i^2 = x_i^3 + ax_i + b$ allora si sceglie $P_m = (x_i, y_i)$, altrimenti si itera la i

Questo algoritmo ha probabilitá di successo pari a $1 - 1/2^h$

8.9 Algoritmo di ElGamal su curve ellittiche

8.9.1 Cifratura

- M sceglie r casuale
- $V = rB$
- $W = P_m + rP_D$ dove P_D é la chiave pubblica di D
- Invia $\langle V, W \rangle$

8.9.2 Decifrazione

- Calcola $P_m = W - n_D V = P_m + rP_D - n_D(rB) =$
  $= P_m + \cancel{r(n_D B)} - \cancel{n_D r B}$

8.10 Sicurezza della crittografia su curve ellittiche

- Per calcolare il logaritmo discreto in algebra modulare e la fattorizzazione degli interi esiste un algoritmo sub-esponenziale chiamato *index calculus*
- Questo algoritmo richiede in media $O(2^{\sqrt{b \log b}})$ operazioni per chiavi di b bit
- Per il problema del logaritmo discreto su curve ellittiche invece non esiste un algoritmo del genere
- L'algoritmo più efficiente conosciuto ad oggi (*Pollard* ρ) richiede in media $O(2^{b/2})$ operazioni quindi é pienamente esponenziale

TDEA, AES (bit della chiave)	RSA e DH (bit del modulo)	ECC (bit dell'ordine)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

Chapter 9

La firma digitale

Ai protocolli crittografici sono richieste tre funzionalitá importanti

- Identificazione : Un sistema deve essere in grado di accertare l'identitá di un utente che richiede di accedere ai suoi servizi
- Autenticazione : Il destinatario di un messaggio deve essere in grado di accertare l'identitá del mittente e l'integritá di un crittogramma ricevuto
- Firma digitale : Come una firma manuale, deve possedere tre caratteristiche (accertabili anche da una terza parte facente da giudice)
  - Il mittente non ha facoltá di ripudiare il messaggio
  - Il destinatario deve poter accertare l'identitá del mittente e l'integritá del crittogramma
  - Il destinatario non deve poter sostenere che $m' \neq m$ é il messaggio inviatogli

9.1 Funzioni hash one-way

- Una funzione hash $f : X \to Y$ é definita per $n = |X| \gg m = |Y|$
- La differenza di cardinalitá tra X e Y implica che esiste una partizione di X in sottoinsiemi disgiunti $X_1, \ldots, X_m \mid \forall i \in [1, m]$ tutti gli elementi in X_i hanno come immagine uno stesso elemento in Y
- Una funzion hash one-way deve soddisfare tre caratteristiche
  - Per ogni $x \in X$ é computazionalmente facile calcolare $f(x)$
  - Per la maggior parte degli $y \in Y$ é computazionalmente difficile determinare x tale che $f(x) = y$ (one way)
  - É computazionalmente difficile determinare una coppia x', x'' in X tale che $f(x') = f(x'')$ (claw free)

9.2 Identificazione

- Le password in un database vengono salvate sotto forma di hash con il seguente meccanismo
- Durante la fase di registrazione un utente sceglie una password P_u
- Il sistema genera un seme S_u e calcola un hash $Q_u = f(P_u S_u)$ e memorizza $\langle u, S_u, Q_u \rangle$
- Quando un utente prova ad effettuare l'accesso, il sistema recupera il suo seme e calcola Q'_u
- Se $Q'_u = Q_u$ l'identificazione ha avuto successo
- Alla password viene aggiunto un seme perchè altrimenti password uguali genererebbero hash uguali

9.2.1 Identificazione tramite RSA

- Il sistema S genera un valore casuale $r < n$ e lo invia in chiaro a U
- U applica la sua chiave privata r calcolando $f = r^d \bmod n$
- S verifica l'identitá di U applicando la chiave pubblica di U stesso verificando che $f^e \bmod n = r$

Questo protocollo però richiede che U si fidi di S. Infatti se S non fosse *chi dice di essere*, potrebbe richiedere a U di applicare la propria chiave privata a messaggi opportunamente creati per inferire informazioni su d

9.3 Autenticazione

- Il meccanismo di autenticazione puó essere descritto attraverso una funzione $\mathcal{A}(m, k)$ che genera un'informazione (detta MAC) utile a garantire la provenienza e l'integritá di m
- Se non é richiesta la confidenzialitá, `Mitt` spedisce la coppia $\langle m, \mathcal{A}(m, k) \rangle$
- Altrimenti spedisce la coppia $\langle \mathcal{C}(m, k'), \mathcal{A}(m, k) \rangle$
- Si può generare un MAC tramite una funzione hash one-way concatenando il messaggio e la chiave
- $\mathcal{A}(m, k) = h(mk)$

9.4 Firma digitale

9.4.1 Messaggio cifrato e firmato in hash

- Il mittente U calcola $f = \mathcal{D}(h(m), k_U[prv])$ e $c = \mathcal{C}(m, k_V[pub])$
- Spedisce la tripla $\langle U, c, f \rangle$ a V
- V calcola $m = \mathcal{D}(c, k_V)[prv]$ e $h(m) = \mathcal{C}(f, k_U[pub])$
- Se $h(m)$ é uguale al valore ottenibile ricalcolando la funzione hash sul messaggio, la firma é valida

9.5 La Certification Authority

- Le `CA` autenticano le associazioni $\langle$ `utente, chiave pubblica` $\rangle$
- Un certificato contiene la chiave pubblica e una lista di informazioni relative al suo proprietario, tutto opportunamente firmato dalla `CA`
- Se `U` vuole comunicare con `V`, richiede cert_V alla `CA`

9.5.1 Messaggio cifrato, firmato in hash, e certificato

- `U` calcola f e c come nel protocollo senza certificato
- Spedisce la tripla $\langle cert_U, c, f \rangle$
- `U` verifica $cert_U$ con la sua copia della chiave pubblica della `CA`
- Procede cone nel protocollo senza certificato

9.6 Il protocollo SSL

9.7 Protocolli Zero-Knowledge

- Nei protocolli Zero-Knowledge due entitá (Prover P e Verifier V) non si fidano l'uno dell'altro
- Il prover dovrá essere in grado di dimostrare al Verifier di essere in possesso di una facoltá particolare senza comunicargli alcuna informazione su di essa
- Questi protocolli si basano su una serie di iterazioni, dopo ognuna delle quali in Verifier aumenta la sua confidenza che P sia *chi dice di essere*