

Appunti per Orale di Crittografia

Simone Ianniciello

A.A. 2020/2021

Contents

1	Domande per la prof	5
2	Esercizi	7
2.1	Correttezza RSA	7
2.2	Cifrario perfetto	7
2.2.1	Definizione	7
2.2.2	Enunciato di Shannon	8
2.2.3	Perfettezza di One-Time Pad	8
3		9
4	Il ruolo del caso	11
4.1	Il significato algoritmico della casualità	11
4.1.1	Esistenza di sequenze casuali di lunghezza n	11
4.2	Generatori di numeri pseudo-casuali	12
4.3	Algoritmi randomizzati	12
5	Cifrari storici	13
5.1	Cifrari a sostituzione monoalfabetica	13
5.2	Cifrari a sostituzione polialfabetica	13
5.3	Cifrari a trasposizione	14
6	Cifrari perfetti	15
6.1	Definizione	15
6.2	One-Time Pad	15
6.3	Generazione della chiave	16
7	Il cifrario simmetrico standard	19
7.1	Criteri di Shannon	19
7.2	Il cifrario DES	19
7.2.1	Attacchi	20
7.2.2	Variazioni del DES	21
7.3	AES	21
7.4	Cifrari a composizione di blocchi	21

8	Crittografia a chiave pubblica	23
8.1	Funzioni One-Way Trap-Door	23
8.2	Pregi e difetti dei cifrari a chiave pubblica	24
8.3	Il cifrario RSA	24
8.3.1	Correttezza di RSA	24
8.4	Attacchi all’RSA	25
8.5	Diffie-Hellman per lo scambio di chiavi	26
8.6	Curve ellittiche su campi finiti	26
8.7	Diffie-Hellman su curve ellittiche	27
8.8	Algoritmo di Koblitz	27
8.9	Algoritmo di ElGamal su curve ellittiche	27
8.9.1	Cifratura	27
8.9.2	Decifrazione	27
8.10	Sicurezza della crittografia su curve ellittiche	28
9	La firma digitale	29
9.1	Funzioni hash one-way	29
9.2	Identificazione	30
9.2.1	Identificazione tramite RSA	30
9.3	Autenticazione	30
9.4	Firma digitale	31
9.4.1	Messaggio cifrato e firmato in hash	31
9.5	La Certification Authority	31
9.5.1	Messaggio cifrato, firmato in hash, e certificato	31
9.6	Il protocollo SSL	31
9.7	Protocolli Zero-Knowledge	31
9.7.1	Fiat-Shamir	32

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

Esercizi

2.1 Correttezza RSA

- $p, q \mid m$
 - Eulero: $a^{\Phi(n)} \equiv 1 \pmod n$
 - $e \times d \equiv 1 \pmod{\Phi(n) = 1 + r\Phi(n)}$
 - $m^{ed} \pmod n \equiv m \times m^{r\Phi(n)} \pmod n \equiv m \times (m^{\Phi(n)})^r \pmod n$
 $\stackrel{\text{Eul}}{\equiv} m \times 1^r \pmod n \equiv m \pmod n$
- $p \mid m \wedge q \nmid m$
 - $m \equiv m^r \equiv 0 \pmod p \implies (m^r - m) \equiv 0 \pmod p$
 - Eulero: $a^{\Phi(q)} \equiv 1 \pmod q$
 - $m^{ed} \pmod q \equiv m^{1+r\Phi n} \pmod q \equiv m \times m^{r(p-1)(q-1)} \pmod q$
 $\equiv m \times (m^{\Phi(q)})^{r\Phi(p)} \pmod q \equiv m \pmod q$
 - $(m^{ed} - m) \equiv 0 \pmod q \implies (m^{ed} - m) \mid q$
 - $(m^{ed} - m) \equiv 0 \pmod n \implies m^{ed} \equiv m \pmod n$

2.2 Cifrario perfetto

2.2.1 Definizione

- Un cifrario si dice perfetto se non é possibile inferire alcuna informazione sul messaggio originale, dato il crittogramma associato
- $\forall m \in \text{Msg}, c \in \text{Critto}, \Pr(M = m) = \Pr(M = m \mid C = c)$
- La conoscenza complessiva di un crittoanalista non cambia dopo aver letto il crittogramma in transito sul canale

2.2.2 Enunciato di Shannon

- Dati M , l'insieme dei messaggi possibili e K , l'insieme delle chiavi
- Per Shannon : $|K| \geq |M|$
- Poniamo per assurdo che $|K| < |M|$
- Fissato un crittogramma $c \mid \mathcal{Pr}(C = c) > 0$, esso corrisponde a $s \leq |K|$ messaggi (non necessariamente distinti) in M
- Dato che $s \leq |K| < |M|$ allora necessariamente esiste almeno un messaggio $m \mid \mathcal{Pr}(M = m) > 0$ non ottenibile da c
- Quindi $\mathcal{Pr}(M = m \mid C = c) = 0 \neq \mathcal{Pr}(M = m)$

2.2.3 Perfettezza di One-Time Pad

- $\mathcal{Pr}(M = m \mid C = c) = \frac{\mathcal{Pr}(M = m, C = c)}{\mathcal{Pr}(C = c)}$
- Per definizione di XOR, fissato m , chiavi diverse corrispondono a crittogrammi diversi
- Perciò $\mathcal{Pr}(C = c) = (1/2)^n$ é costante
- Quindi gli eventi $(C = c) \wedge (M = m)$ sono indipendenti
- Ne risulta che $\mathcal{Pr}(M = m \mid C = c) = \frac{\mathcal{Pr}(M = m) \times \cancel{\mathcal{Pr}(C = c)}}{\cancel{\mathcal{Pr}(C = c)}}$

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 é casuale se
 - $\mathcal{K}(h) \geq |h| - \lceil \log_2(h) \rceil$

4.1.1 Esistenza di sequenze casuali di lunghezza n

- n : Lunghezza della rappresentazione binaria
- $S = 2^n$: Tutte le sequenze binarie lunghe n
- Chiamiamo T l'insieme delle sequenze non casuali all'interno di S
- $N = 2^{n^{\lceil \log_2 n \rceil - 1}}$: Sequenze più corte di $n - \lceil \log_2 n \rceil$
- N Contiene tutte anche tutti i programmi che generano le sequenze T
- $T \leq N < S$
- $T/S < 2^{-\lceil \log_2 n \rceil}$ Tende a 0 al crescere di n

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 $z = \frac{N-1}{2^w}$
 - y : $y \in [2, N-1]$
 - P_1 : $\text{mcd}(N, y) = 1$
 - P_2 : $(y^z \bmod N = 1) \vee (\exists i, 0 \leq i \leq w-1 : y^{2^i z} \bmod N = -1)$
 - $(P_1 \wedge P_2) = \text{false}$: N é sicuramente composto
 - $(P_1 \wedge P_2) = \text{true}$: N é primo con probabilità $p \geq \frac{3}{4}$

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 può 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 é uguale a $h! - 1$
- Permutazione di colonne
 - $k = \langle c, r, \pi \rangle$
 - π é 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 é scritto in una tabella $q \times q$
 - La chiave é 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 é $G = 2^{q^2/2}$

Chapter 6

Cifrari perfetti

6.1 Definizione

- $\mathcal{Pr}(M = m)$: Probabilità che m sia il messaggio che deve essere inviato
- $\mathcal{Pr}(M = m \mid C = c)$: Probabilità che il messaggio originale fosse m dato il crittogramma c in transito
- Un cifrario é perfetto se :
 - $\forall m \in \text{Msg}, c \in \text{Critto} : \mathcal{Pr}(M = m \mid C = c) = \mathcal{Pr}(M = m)$
 - Cioè m e c sono totalmente scorrelati tra loro perciò 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 così, dato un crittogramma c , esisterebbe un messaggio m' non generabile tramite la decrittazione di c con tutte le chiavi del sistema
 - * $\mathcal{Pr}(M = m' \mid C = c) = 0 \neq \mathcal{Pr}(M = m)$

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 $k = k_1 k_2 \dots k_n : n \geq |m|$, $\Pr(k_i = 0) = \Pr(k_i = 1) = 1/2$
- Cifratura : Dati $m = m_1 m_2 \dots m_n$ e $k = k_1 k_2 \dots k_n \implies c = c_1 c_2 \dots c_n$ 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
 - $\Pr(M = m \mid C = c) = \frac{\Pr(M=m, C=c)}{\Pr(C=c)}$
 - Per definizione di XOR chiavi diverse generano crittogrammi diversi. Quindi fissato m abbiamo $\Pr(C = c) = (1/2)^n$ quindi gli eventi $\{M = m\}$ e $\{C = c\}$ sono indipendenti
 - $\Pr(M = m, C = c) = \Pr(M = m) \times \Pr(C = c)$
 - $\Pr(M = m \mid C = c) = \frac{\Pr(M=m) \times \Pr(C=c)}{\Pr(C=c)} = \Pr(M = m)$

6.3 Generazione della chiave

- Per definizione di XOR la chiave non può essere riutilizzata perchè 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 realtà dei fatti, non tutti i 2^n messaggi sono messaggi possibili; Questo perchè 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 é circa α^n con $\alpha < 2$ variabile a seconda della lingua

Chapter 7

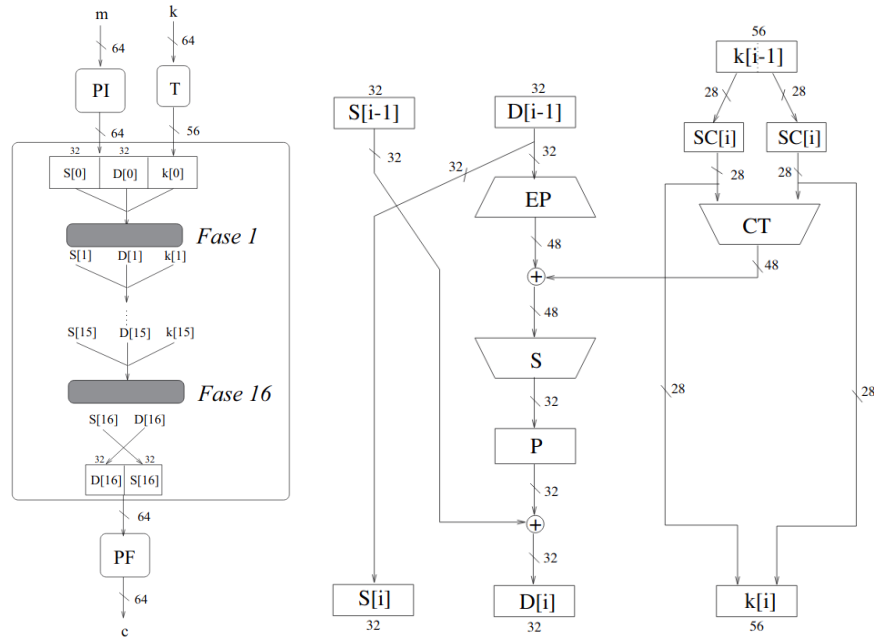
Il cifrario simmetrico standard

7.1 Criteri di Shannon

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

7.2 Il cifrario DES

- Il messaggio viene diviso in blocchi da 64 bit
- La chiave é composta da 8 Byte in cui ogni ottavo bit é di parit 
- La cifratura e decifrazione del messaggio avviene attraverso 16 round
- Dalla chiave vengono create 16 sottochiavi $k[0], k[1], \dots, k[15]$ 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

- Proprietà 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 può 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 è 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 $\langle m, c \rangle$ con m scelti dal crittoanalista
 - Si possono quindi analizzare i crittogrammi ottenuti per assegnare probabilità diverse a varie chiavi
 - La chiave originale dovrebbe quindi emergere come quella con probabilità più 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** : Ciò 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 è 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 è trasformato tramite una S-Box
 - **Op2** : La matrice ottenuta è 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 può affidare ai cifrari a composizione di blocchi

- Il messaggio viene diviso in blocchi $m = m_1m_2 \dots m_s$ 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 però $\gcd(x, s) = 1$ 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 \mathbb{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 $e : e < \Phi(n) \wedge \text{mcd}(e, \Phi(n)) = 1$
 - Si calcola $d = e^{-1} \bmod \Phi(n)$ Tramite **Euclide Esteso**
 - $k[\text{pub}] = \langle e, n \rangle$
 - $k[\text{prv}] = \langle d \rangle$
- Per cifrare un messaggio m esso deve essere codificato come un intero $m < n$
- m può essere diviso in blocchi di lunghezza $\lfloor \log_2(m) \rfloor$
- $c = \mathcal{C}(m, k[\text{pub}]) = m^e \bmod n$
- $m = \mathcal{D}(c, k[\text{prv}]) = c^d \bmod n = m^{ed} \bmod n = m$

8.3.1 Correttezza di RSA

Si distinguono due casi

- p e q non dividono m
 - $\text{mcd}(m, n) = 1$
 - 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 $z : z^2 - n = w^2$
- $\text{mcd}((p-1), (q-1))$ deve essere piccolo
 (Si scelgono quindi p e $q : \text{mcd}(\frac{p-1}{2}, \frac{q-1}{2}) = 1$)
- $e = 1+\Phi(n)/k$ con $m \mid k$ e $\text{mcd}(m, n) = 1$
 $c = m^e \bmod n = m \times (m^{\Phi(n)})^{1/k} \bmod n = m \bmod n$
- 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*
 - * $n = n_1 \times n_2 \times \dots \times n_e$

- * $m' < n$
- * $m' \equiv m^e \pmod n$
- $m^e \pmod n < n$
- $m' = m^e$: 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 più utenti
 - Date due chiavi $\langle e_1, n \rangle, \langle e_2, n \rangle : \text{mcd}(e_1, e_2) = 1$
 - 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
 - $m = m^{e_1 r + e_2 s} = (c_1^r \times c_2^s) \pmod n = ((c_1^{-1})^{-r} \times c_2^s) \pmod n$
 - Tramite **Euclide Esteso** si calcola $c^{-1} \pmod n$
 - Si può quindi calcolare m in tempo polinomiale
- RSA ha gli stessi problemi (e le stesse soluzioni) del DES per la periodicità 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 \mathbb{Z}_p^*
- A sceglie $x < p$ casuale e calcola $X = g^x \pmod p$ e spedisce X a B
- B fa lo stesso ($Y = g^y \pmod p$)
- Entrambi calcolano $k[\text{session}] = Y^x \pmod p = X^y \pmod p = g^{xy} \pmod p$
- Se un crittoanalista intercetta X (o Y) dovrebbe calcolarsi il logaritmo discreto per risalire a x (o y)
- L'unico attacco possibile è MITM

8.6 Curve ellittiche su campi finiti

- $E_p(a, b) = \{(x, y) \in \mathbb{Z}_p^2 \mid y^2 \pmod p = (x^3 + ax + b) \pmod p\} \cup \{O\}$
- 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 $P_A = n_AB$ e invia P_A a C
- C fa lo stesso ($P_C = n_CB$)
- Entrambi calcolano $S = n_AP_C = n_CP_A = n_An_CB$
- $k = x_S \bmod 2^{256}$

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 probabilità 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 funzionalità importanti

- Identificazione : Un sistema deve essere in grado di accertare l'identità di un utente che richiede di accedere ai suoi servizi
- Autenticazione : Il destinatario di un messaggio deve essere in grado di accertare l'identità del mittente e l'integrità 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 facoltà di ripudiare il messaggio
 - Il destinatario deve poter accertare l'identità del mittente e l'integrità del crittogramma
 - Il destinatario non deve poter sostenere che $m' \neq m$ è il messaggio inviatogli

9.1 Funzioni hash one-way

- Una funzione hash $f : X \rightarrow Y$ è definita per $n = |X| \gg m = |Y|$
- La differenza di cardinalità tra X e Y implica che esiste una partizione di X in sottoinsiemi disgiunti $X_1, \dots, X_m \mid \forall i \in [1, m]$ tutti gli elementi in X_i hanno come immagine uno stesso elemento in Y
- Una funzione hash one-way deve soddisfare tre caratteristiche
 - Per ogni $x \in X$ è computazionalmente facile calcolare $f(x)$
 - Per la maggior parte degli $y \in Y$ è computazionalmente difficile determinare x tale che $f(x) = y$ (one way)
 - È 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'identità 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 può essere descritto attraverso una funzione $\mathcal{A}(m, k)$ che genera un'informazione (detta MAC) utile a garantire la provenienza e l'integrità di m
- Se non è richiesta la confidenzialità, 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 $\langle \text{utente}, \text{chiave pubblica} \rangle$
- 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 $\langle \text{cert}_U, c, f \rangle$
- U verifica cert_U con la sua copia della chiave pubblica della CA
- Procede come nel protocollo senza certificato

9.6 Il protocollo SSL

9.7 Protocolli Zero-Knowledge

- Nei protocolli Zero-Knowledge due entit  (Prover P e Verifier V) non si fidano l'uno dell'altro
- Il prover dovr  essere in grado di dimostrare al Verifier di essere in possesso di una facolt  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*

9.7.1 Fiat-Shamir

- Obiettivo : Autenticazione (senza CA)
- P sceglie $n = pq, s < n$, e calcola $t = s^2 \bmod n$
- Rende nota $\langle n, t \rangle$ e mantiene segreti $\langle p, q, s \rangle$
- Iterazioni:
 1. V chiede a P di iniziare un'iterazione
 2. P genera $r < n$, calcola $u = r^2 \bmod n$, e comunica u
 3. V genera un bit casuale e e lo comunica
 4. P calcola $z = rs^e \bmod n$ e lo comunica
 5. V controlla che $x = z^2 \bmod n == ut^e \bmod n$
 6. Se il controllo va a buon fine si ripete dal passo 1, altrimenti si blocca
- Completezza
 - $x = z^2 \bmod n = (rs^e)^2 \bmod n$
 - $x = ut^e \bmod n = (r^2)(s^2)^e \bmod n$
 - Se si ha lo stesso valore di e in entrambe le equazioni, esse risultano identiche
- Correttezza
 - Se P é disonesto, deve prevedere il valore di e prima che gli venga comunicato e inviare $u = r^2/t^e \bmod n$ e $z = r$