Teoria dei Codici e Crittografia

Dario Balboni

1 Introduzione ai Codici

2 Codici Lineari

3 Codici Ciclici

4 Codici BCH e RS

5 Codici di Goppa

6 Introduzione alla Crittografia

7 Test di Primalità

8 Fattorizzazione su $\mathbb{Z}$

9 Problemi con il Logaritmo Discreto

In questa sezione g indica la base del logaritmo e b l'elemento da trovare tale che $a = g^b$. L'ordine del gruppo viene indicato con n.

9.1 Baby-Step Giant-Step

Detto $m = \lceil \sqrt{n} \rceil$ costruiamo una tabella di (j, g^j) per $j = 1, \dots, m$. A questo punto per calcolare il logaritmo discreto di a calcoliamo ag^{-im} per $i = 1, \dots, m$ e controlliamo se esso è uguale ad un qualche g^j. Se ciò succede abbiamo che $ag^{-im} = g^j$ e quindi $a = g^{j+im}$.

9.2 ρ di Pollard per il logaritmo discreto

Si basa su un metodo lepre-tartaruga: dividiamo G in tre insiemi G_0, G_1, G_2 tali che $1 \notin G_1$. Definiamo quindi $f(x) = \begin{cases} ax & \text{se } x \in G_0 \\ x^2 & \text{se } x \in G^1 \\ gx & \text{se } x \in G_2 \end{cases}$ che in un algoritmo vero scriveremmo come due successioni sugli esponenti di a e di g.

Se troviamo una collisione $a^\gamma g^\beta = a^{\gamma'} g^{\beta'}$ allora si ha $b = (\gamma - \gamma')^{-1}(\beta' - \beta) \mod n$.

9.3 Pohlig-Hellman

Particolarmente efficiente se l'ordine del gruppo si fattorizza con primi piccoli. Scriviamo $n = \prod_{i=1}^r p_i^{e_i}$ e $b = \log_g a$. Vogliamo prima di tutto determinare $b_i \equiv b \mod p_i^{e_i}$ per poi rimontare la soluzione con il teorema cinese del resto.

Ogni intero b_i viene ottenuto calcolando le cifre l_j per $j = 0, \ldots, e_i - 1$ della sua espansione \$p$_i$\$-aria nel seguente modo: al passo j (posto $q = p_i$ e $e = e_i$) si calcola $\gamma = g^{l_0 + l_1 q + \ldots + l_{j-1} q^{j-1}}$ e si nota che (scrivendo $b = b_i + kq^e$) $(g^{n/q^{j+1}})^{kq^e} = 1$. Da ciò segue che $\tilde{a} = (a\gamma^{-1})^{n/q^{j+1}} = \tilde{g}_j^l$ e quindi si può usare un altro algoritmo per calcolare $b_i = \log_{\tilde{g}} \tilde{a}$.

9.4 Basi di fattori

Si scelgono un piccolo numero di elementi "irriducibili", che vengono chiamati base di fattori. Ad esempio si possono prendere i primi piccoli $\mathcal{B} = \{p_1, \ldots, p_h\}$. A questo punto cerchiamo degli r_i tali che g^{r_i} si riesca a scrivere con elementi della base: $g^{r_i} = \prod_{j=1}^{h} p_j^{t_{ij}}$. In questo modo otteniamo delle relazioni lineari $r_i = \sum_{j=1}^{h} t_{ij} x_j$ dove le incognite x_j sono i logaritmi di p_j. Quando abbiamo abbastanza relazioni risolviamo il sistema lineare (ricordando di usare il teorema cinese per evitare di incappare in zero-divisori).

Noti gli x_j possiamo prendere una potenza a caso s e controllare se ag^s si può scrivere nella base di fattori $ag^s = \prod_j p_j^{t_j}$. Se sì possiamo ricavare $b = (\sum_j t_j x_j) - s$.

10 Principali crittosistemi a chiave pubblica

10.1 Diffie-Hellman

È più che altro un protocollo di scambio di chiavi.

- Alice e Bob scelgono di comune accordo un primo p e un generatore g di $\mathbb{Z}_p^*$.
- Alice sceglie un numero segreto a e Bob un numero segreto b.
- Alice invia a Bob $A = g^a \mod p$, Bob invia ad Alice $B = g^b \mod p$.
- Alice calcola $B^a = g^{ab} \mod p$ e Bob calcola $A^b = g^{ab} \mod p$.

In questo modi essi ottengono un segreto comune. Un eventuale terzo che potesse ascoltare la loro conversazione imparerebbe solo p, g, g^a e g^b ed avrebbe bisogno di un metodo efficiente per calcolare g^{ab} dati g^a e g^b, che al giorno d'oggi non è noto (ed il meglio che si possa fare è il logaritmo discreto di uno dei due).

10.2 Elgamal

Protocollo di cifratura asimmetrica.

- Alice sceglie un primo p ed un generatore $g \in \mathbb{Z}_p^*$. Successivamente sceglie a e calcola $A = g^a \mod p$. La chiave pubblica è (p, g, A) mentre quella privata è a.
- Bob che vuole mandare un messaggio m ad Alice, sceglie un intero b e calcola $B = g^b \mod p$. Quindi calcola la chiave di cifratura $K = A^b = g^{ab}$, cifra il messaggio calcolando $m' = Km$ ed invia ad alice (B, m').
- Per decifrare, Alice calcola la chiave K come B^a, quindi recupera il messaggio calcolando $K^{-1}m' = m$.

10.3 RSA

Protocollo di cifratura asimmetrica. Denotiamo nel seguito con $\phi(n)$ la funzione di Eulero di n.

- Alice genera la sua coppia di chiavi: sceglie opportunamente due numeri primi p e q e ne fa il prodotto $n = pq$. Calcola infine $\phi(n) = (p-1)(q-1)$ e sceglie un numero e tale che $1 < e < \phi(n)$ e $\gcd(e, \phi(n)) = 1$. La coppia (n, e) è la chiave pubblica di Alice. Infine essa calcola d tale che $de \equiv 1 \mod \phi(n)$, che le servirà per decifrare. La sua chiave privata è formata da (p, q, d).
- Per mandare un messaggio m (compreso tra 0 e n e coprimo con n) ad Alice, Bob calcola $c = m^e \mod n$ ed invia c ad Alice.
- Per recuperare il messaggio Alice deve solamente calcolare $c^d = m^{ed} = m \mod n$.

La sicurezza di RSA discende dal fatto che per trovare d è necessario conoscere $\phi(n)$, ma questo è dimostrabilmente tanto difficile quanto fattorizzare n. Questo sistema si basa quindi **effettivamente** sulla difficoltà della fattorizzazione.

11 Curve Ellittiche

11.1 Equazione di Weierstrass

Dato K un campo, un'equazione della forma $Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$ dove $a_1, \ldots, a_6 \in K$ viene detta equazione di Weierstrass ed identifica una curva ellittica.

11.2 Legge di Gruppo

Si può definire una legge di gruppo sulle cubiche. Ne scriviamo solo le formule: per calcolare $R = (x_3, y_3)$ somma di $P = (x_1, y_1)$ e $Q = (x_2, y_2)$ si ha $\begin{cases} x_3 = m^3 + a_1m - a_2 - x_1 - x_2 \\ y_3 = -(m + a_1)x_3 - q - a_3 \end{cases}$ dove $y = mx + q$ è la retta passante per P e Q (tangente se $P = Q$) con $m = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{se } P \neq Q \\ \frac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3} & \text{se } P = Q \end{cases}$ con $q = y_1 - mx_1$.

11.3 Teorema di Hasse

Data E una curva ellittica definita su $\mathbb{F}_q$ si ha la stima $\left|E - (q+1)\right| \leq 2\sqrt{q}$

11.4 Contare il numero di punti

Prendiamo un punto P e ne calcoliamo l'ordine con un metodo lepre-tartaruga sulla successione $P_i = i \cdot P$. Speriamo di trovare, nell'intervallo fornito dal Teorema di Hasse, un solo multiplo dell'ordine trovato. Eventualmente possiamo calcolare più ordini e verificare che cadano nella stima di Hasse solo i multi dei loro mcm.

11.5 Problema del logaritmo discreto

Il problema del logaritmo discreto è definibile su una curva ellittica come: dati $P, Q \in E$ determinare in più piccolo $k \in \mathbb{Z}$ tale che $Q = k \cdot P$. Si può quindi adattare lo scambio Diffie-Hellman alle curve ellittiche.

11.6 Scegliere una curva ellittica

Un metodo per scegliere una curva ellittica contentente un punto P è: prima scegliere il punto $P = (x, y) \in (\mathbb{F}_q)^2$, scegliere $a \in \mathbb{F}_q$ e porre poi $b = y^2 - x^3 - ax$.

11.7 Goldwasser-Kilian

È un test di primalità simile al test di Pocklington. Sia n un intero positivo, $a, b \in \mathbb{Z}_n$ e sia $E = \{(x, y) \in (\mathbb{Z}_n)^2 \mid y^2 \equiv x^3 + ax + b \mod n\} \cup \{O\}$ dove O è un simbolo che denota il "punto all'infinito". Sia inoltre m un intero. Supponiamo che esista un primo q che divide m e tale che $q > (n^{1/4} + 1)^2$. Se esiste $P \in E$ tale che $m \cdot P = O$ e $\frac{m}{q}P \neq O$ allora n è primo.

Le formule per la somma di punti prevedono anche delle divisioni. L'algoritmo potrebbe quindi doversi fermare se non possiamo dividere, ma in questo caso avremmo trovato che n non è primo (e ne avremmo addirittura trovato un divisore).

11.8 Algoritmo di fattorizzazione di Lenstra

È un algoritmo di fattorizzazione, simile all'algoritmo $p-1$ di Pollard. Si basa sull'osservazione che il calcolo di $k \cdot P$ richiede la divisione tra classi di resto modulo n, che può essere compiuta con l'algoritmo euclideo esteso se $\gcd(n, v) = 1$. Se $\gcd(n, v) = n$ comunque non ci sono problemi perché l'algoritmo restituisce il punto all'infinito della cubica, mentre se $\gcd(n, v) \neq 1, n$ abbiamo trovato un divisore di n.

1. Scegliamo un'equazione del tipo $y^2 = x^3 + ax + b$ in $\mathbb{Z}_n$ ed un punto P
2. Calcoliamo $eP \in E$, dove e è prodotto di molti numeri piccoli (prodotto di potenze di primi piccoli, oppure $B!$ per qualche B piccolo. In questo modo si può calcolare efficientemente).
3. Si possono presentare tre eventualità:
   - Se siamo riusciti a compiere tutte le operazioni, proviamo qualche altra curva e/o punto di partenza
   - Se abbiamo trovato $k \cdot P = O$ in qualche fase, ricominciamo da capo (poiché O è elemento neutro non ci sposteremo da esso).
   - Se ad un certo punto abbiamo $\gcd(v, n) \neq 1, n$, abbiamo trovato un fattore non banale di n.

12 Altri crittosistemi

13 Lezioni del Maestro

Disclaimer: Le parole del Maestro sono a volte di difficile decifrazione, e comunque invitano sempre ad una riflessione personale piuttosto che ad un bieco nozionismo. Pertanto siete pregati di non prendere con assoluta certezza quanto scritto di seguito che serve principalmente ad ispirare delle piacevoli conversazioni con i vostri amici.

13.1 Assunzioni per la sicurezza in crittografia (modelli)

13.1.1 P $\neq$ NP

Si assume sempre che P $\neq$ NP, dove si suppone che i problemi in P siano quelli efficientemente risolubili, mentre quelli NP-hard o NP-completi siano impossibili da risolvere.

Alcuni problemi che si pensavano essere strettamente in NP si sono poi rivelati essere in P. Ad esempio PRIMES (problema decisionale: dato n naturale è primo?):

- Algoritmo Miller-Rabin $\implies$ BPP
- AKS (2009) $\implies$ P

13.1.2 Scenari per la cifratura

Vedere questa risposta di Stack Overflow per una spiegazione concisa e soddisfacente, della quale ciò che segue è una brutta copia.

1. Indistinguibilità Dati due oggetti di cui uno è la codifica di un messaggio e l'altro è una successione casuale di bit i due sono indistinguibili: non c'è un algoritmo che permetta di dire chi è l'uno e chi è l'altro.

   Questa nozione viene spesso considerata sotto ipotesi aggiuntive (CPA, CCA, CCA2) nel setting di un gioco tra un challenger ed un attaccante nel quale l'attaccante ha diritto a consultare alcuni oracoli e il suo scopo è di rompere il sistema crittografico. Denoteremo con λ il parametro di sicurezza del crittosistema, con $(K_E, K_D) = KG(\lambda)$ la procedura di generazione della coppia chiave pubblica (di cifratura) e chiave privata (di decifratura). Gli algoritmi di cifratura E e D si suppongono essere noti a tutte le parti (così come KG) ma possono essere non deterministici (nonostante ciò verranno scritti come funzioni). È garantito che si riesca sempre a decifrare un messaggio cifrato: $D(K_D, E(K_E, M)) = M$.

   Si ha indistinguibilità quando, nei protocolli sotto esposti, la probabilità dell'avversario di vincere il gioco è minore di $\frac{1}{2} + \varepsilon$ dove ε è una funzione negligibile nel parametro di sicurezza λ.

2. IND-CPA: Indistinguibilità sotto Chosen Plaintext Attack **Descrizione**: L'avversario genera due parole di eguale lunghezza. Il challenger decide, casualmente, di cifrarne uno dei due. L'avversario deve quindi indovinare quale dei due è stato cifrato.

   Algoritmo

   (a) Challenger: istanzia la coppia di chiavi $(K_E, K_D) = KG(\lambda)$.

   (b) Avversario: sceglie m_0, m_1 due messaggi della stessa lunghezza e li manda al challenger. Può compiere altre operazioni in tempo polinomiale che includano chiamate all'oracolo di cifratura $E(K_E, -)$.

   (c) Challenger: sceglie $b \in \{0, 1\}$ casualmente, calcola $C = E(K_E, m_b)$ e manda C all'avversario.

   (d) Avversario: esegue altre operazioni in tempo polinomiale che includano chiamate all'oracolo di cifratura. Successivamente manda in output $g \in \{0, 1\}$.

   (e) Se $g = b$ l'avversario vince.

   Osservazioni: Questo modello è troppo debole, perché assume una sola interazione tra l'avversario e il challenger.

3. IND-CCA: Indistinguibilità sotto Chosen Ciphertext Attack **Descrizione**: Lo scenario è come il precedente ma l'avversario può chiamare oracoli di cifratura o decifratura **prima** di spedire il messaggio.

   Algoritmo

   (a) Challenger: istanzia la coppia di chiavi $(K_E, K_D) = KG(\lambda)$.

   (b) Avversario: sceglie m_0, m_1 due messaggi della stessa lunghezza e compie operazioni in tempo polinomiale includendo chiamate agli oracoli di cifratura $E(K_E, -)$ e di decifratura $D(K_D, -)$. Successivamente spedisce entrambi i messaggi al challenger.

   (c) Challenger: sceglie $b \in \{0, 1\}$ casualmente, calcola $C = E(K_E, m_b)$ e manda C all'avversario.

   (d) Avversario: esegue altre operazioni in tempo polinomiale **senza poter chiamare nuovamente gli oracoli**. Manda in output $g \in \{0, 1\}$.

   (e) Se $g = b$ l'avversario vince.

Osservazioni: Questo modello è più sicuro perché prevede la possibilità di interazioni ripetute. Ciò significa che la sicurezza non si indebolisce con il tempo.

4. IND-CCA2: Indistinguibilità sotto Adaptive Chosen Ciphertext Attack **Descrizione**: Oltre alle capacità in IND-CCA, all'avversario è concesso consultare gli oracoli dopo aver ricevuto C, ma non può spedire C stesso agli oracoli.

   Algoritmo: come sopra ma (d) viene sostituito dalla possibilità di eseguire operazioni in tempo polinomiale con chiamate ad entrambi gli oracoli esclusa la decifratura di C.

   Osservazioni: La necessità di IND-CCA2 suggerisce che la possibilità di utilizzare l'oracolo di decifratura dopo aver conosciuto il testo cifrato può dare parecchio vantaggio in alcuni schemi, visto che le richieste all'oracolo possono essere scelte in base allo specifico testo cifrato.

13.2 Possibili attacchi a Crittosistemi

13.2.1 Insicurezza di RSA

RSA come spiegato nei libri è insicuro e non soddisfa IND-CPA per via della parziale omomorficità: se so crittare a e b allora so anche crittare $a \cdot b$. Inoltre se a viene sempre cifrato nello stesso modo è possibile sapere se un messaggio cifrato contiene a oppure no. Per questo è necessario aggiungere del padding e qualche informazione casuale al messaggio trasmesso per evitare questo tipo di attacchi.

Nell'RSA standard lo zero e l'uno vengono sempre codificati come sé stessi e questa è un'altra debolezza.

Inoltre chiave pubblica e chiave privata **non sono simmetriche**: se l'esponente privato è piccolo ($< \sqrt{n}$) esso può essere riconosciuto facilmente.

13.2.2 Attacco di prossimità all'implementazione RSA con TCR

Supposizione: Chi decifra il messaggio (e quindi conosce p e q) potrebbe voler velocizzare i conti ed esponenziare il messaggio modulo i due primi per poi ricomporre il risultato con il Teorema Cinese.

Tipo di attacco: L'attacco è basato sulla prossimità al computer ricevente: vi è un microfono che ascolta il computer che fa i calcoli. Potendo scegliere il messaggio in chiaro (chosen plaintext) si riusciva a scoprire che bit ci fosse in una certa posizione ascoltando solo il rumore che fa il computer durante una decifrazione. Con pochi passaggi si riusciva a ricavare completamente la chiave privata (p e q).

Soluzione: Basta non usare il TCR. In questo modo chi ascolta può imparare n (che comunque già conosce) ma non p e q.

13.2.3 Attacchi algoritmici a scambi Diffie-Hellman

Alcuni metodi di rottura di Diffie-Hellman non sono completamente esponenziali: vanno come $O(2^{\sqrt{n}})$ o $O(2^{\sqrt[3]{n}})$ (General Number Field Sieve). Oltretutto esistono attacchi basati sui computer quantistici (Fattorizzazione di Shor) che possono rompere questi sistemi in tempo polinomiale.

13.3 Funzioni di Hashing

Vogliamo trasformare una stringa di lunghezza arbitraria in una stringa di lunghezza fissata (hash) in modo che sia difficilmente contraffattibile, ovvero che sia possibilmente iniettiva. Non essendo ciò possibile si chiede che possa resistere ad un preimage attack.

13.3.1 Preimage Attack

Sia h la funzione di hashing (nota) ed x un messaggio (non noto). Sapendo $h(x)$ deve essere computazionalmente impossibile trovare un messaggio x' tale che $h(x') = h(x)$.

13.3.2 Derivazione da un crittosistema

Si può derivare una funzione di hashing da un crittosistema seppur in maniera non efficiente: si divide il messaggio M a blocchi $b_0, \ldots, b_k$. L'algoritmo specifica un blocco di partenza c_0 fissato per tutti. Si procede ora induttivamente per ottenere c_{i+1} si usa b_i per cifrare c_i. L'hash cercato è quindi c_{k+1}.

13.4 Algoritmi di Firma

13.4.1 Derivazione da un crittosistema ed una funzione di hashing

Mostro di saper cifrare un hash derivato dal messaggio originario. In questo modo il ricevente (sotto opportune ipotesi di difficoltà di collisioni e di sicurezza del crittosistema) può aspettarsi che sia stato io a mandare il messaggio.

Attenzione che normalmente non si possono usare le stesse chiavi per cifratura e firma perché si indeboliscono a vicenda visto che la firma - concettualmente - equivale ad una decodifica di messaggi arbitrari.

13.5 Merkle-Hellman

Crittosistema basato su Subset Sum. Funzionamento: dato un insieme di numeri $a_1, \ldots, a_n \in \mathbb{N}$ e $c_1, \ldots, c_n \in \{0,1\}$ codifico il messaggio $(c_i)_i$ inviando $A = \sum_i c_i a_i \in \mathbb{N}$. È dimostrato che dato $\{a_i\}_i$ e A, trovare c_i è un problema NP-hard (ciò non significa che una certa istanza non possa essere molto semplice da rompere). Inoltre, affinché esso possa essere utilizzato crittograficamente, è necessario che (avendo a disposizione dei dati in più) sia possibile decifrarlo rapidamente. Inoltre la soluzione deve essere unica.

Se gli a_i sono supercrescenti, ovvero $a_{i+1} > \sum_{k=1}^{i} a_k$, dato $A = \sum_i c_i a_i$ è molto semplice trovare i c_i. Idea: posso prendere m e $d < m$ scelto casualmente (ma vicino ad m per mascherare anche i numeri piccoli) e considerare $b_i = da_i \mod m$ e pubblicare come base $\{b_i\}_i$. Quando ricevo $\sum_i c_i b_i$ moltiplico per l'inverso di d ed ottengo $\sum_i c_i a_i \mod m$ da cui recupero il messaggio originario.

13.6 Reticoli Interi

Sono interessanti perché per ora sono gli unici tipi di crittosistemi classici che ancora resistono ai computer quantistici.

Determinante di una matrice quadrata: $\det A = \sqrt{|\det (A^t \cdot A)|}$.

13.6.1 Teorema di Minkowski

Sia S un insieme convesso, $S \subseteq \mathrm{Span}_\Lambda$ e simmetrico ($x \in S - x \in S$). Se $\mu(S) > 2^n \cdot \det \Lambda$ allora $S \cap \Lambda$ è non vuoto e contiene un $x \neq 0$.

13.6.2 Shortest Vector Problem

Dato un reticolo trovare il vettore non nullo più corto.

13.6.3 Closes Vector Problem

Dato un reticolo ed un vettore si chiede di trovare il vettore del reticolo più vicino al vettore dato.

13.6.4 Basi Ridotte

Vorremmo avere una descrizione del nostro reticolo con basi fatte da vettori "corti". Data una coppia di vettori a e b in $\mathbb{R}^2$ consideriamo $a+b$ e $a-b$. Diciamo allora che una base di un reticolo in $\mathbb{R}^2$ è ridotta se $||a||, ||b|| \leq ||a+b||, ||a-b||$.

Nel caso una delle disuguaglianze non valga si può sostituire uno dei due vettori con quello più corto trovato.

Se siamo in dimensione 2, l'algoritmo termina sicuramente restituendo una base ridotta per il reticolo. Viene quindi data una definizione di base $\$\delta\$$-ridotta in dimensione arbitraria per permettere all'algoritmo LLL di terminare.

Una base $a_1, \ldots, a_n$ si dice $\$\delta\$$-ridotta (con $\frac{1}{4} < \delta < 1$) se valgono le condizioni $||a_i|| \leq ||a_i \pm a_j||$ ed inoltre si ha $\delta\pi_i(a_i) < \pi_i(a_{i+1})$ dove π_i è "fare Gram-Schmidt fino al punto i-esimo".

13.6.5 Algoritmo LLL (Lenstra-Lenstra-Lovaz)

Permette di trovare una base $\$\delta\$$-ridotta di un reticolo qualunque, fissato δ a priori, in tempo polinomiale.

Funziona nel "modo ovvio":

1. Si controlla se tutte le condizioni sono soddisfatte, nel qual caso ci si ferma
2. Si esegue un passo di riduzione con Gram-Schmidt approssimato
3. Se $\delta\pi_i(a_i) \geq \pi_i(a_{i+1})$ si scambiano a_i ed a_{i+1}

La parte furba di tutto è mostrare che l'algoritmo termina in tempo polinomiale, ma questo l'hanno già fatto Lenstra, Lenstra e Lovaz.

13.6.6 Fattorizzazione di Polinomi a coefficienti interi

13.7 Applicazioni Crittografiche dei Reticoli

13.7.1 Merkle-Hellman

Si può rompere Merkle-Hellman utilizzando LLL considerando la matrice opportuna codificandolo come problema di Shortest Vector. (Tralascio la matrice perché non ho voglia di scriverla)

13.7.2 Goldreich-Goldwasser-Halewi

13.7.3 Fiat-Shamir

13.7.4 Ferigle-Fiat-Shamir

13.7.5 Digital Signature Standard

13.7.6 NTRU