UNIVERSITA' DEGLI STUDI DI PALERMO

FACOLTA' DI SCIENZE MM.FF.NN.

CORSO DI LAUREA IN MATEMATICA PER L'INFORMATICA E LA COMUNICAZIONE SCIENTIFICA

LE CURVE ELLITTICHE IN CRITTOGRAFIA

Tesi di laurea di:

Battaglia Ilenia

ANNO ACCADEMICO 2005-2006

Indice

Prefazione

*La nascita della crittografia, l'arte di nascondere i messaggi, risale ad epoca Antica. La crittografia, dopo aver svolto vari ruoli diversi a secondo dei momenti storici, oggi risulta indispensabile in moltissimi ambienti come, ad esempio, quello commerciale e privato. L'introduzione della crittografia basata su Curve Ellittiche (**ECC**, Elliptic Curve Cryptography) è abbastanza recente, ma si è imposta, comunque, come alternativa ai sistemi crittografici a chiave pubblica ampiamente utilizzati come l'RSA e il DSA. Al momento, però, non esistono algoritmi sufficientemente veloci che risolvano il problema matematico sul quale si basa.*

L' ECC offre lo stesso grado di sicurezza dei sistemi tradizionali (RSA, DSA, Diffie-Hellman) utilizzando chiavi di dimensione inferiore.

Obiettivi e Organizzazione della Tesi

Verrà presentata la Crittografia basata sulle Curve Ellittiche, analizzando sia gli aspetti teorici che i protocolli utilizzati e soprattutto confrontando questa teoria con le moderne soluzioni a chiave pubblica.

Il primo capitolo si apre con una breve esposizione di alcuni

momenti significativi della storia della crittografia e di alcuni concetti di base per affrontare un completo studio su tale argomento; presenta i principali sistemi asimmetrici ed introduce brevemente l'ECC. Il secondo capitolo descrive le Curve Ellittiche e la sua aritmetica, comprese le formule relative alle regole di addizione sulle Curve Ellittiche. Il terzo affronta lo studio del problema matematico sul quale si basa l'ECC, ossia l'Elliptic Curve Discrete Logarithm Problem (ECDLP). Il quarto riguarda il confronto tra l'ECC con altri sistemi a chiave pubblica e le relative conclusioni a cui si giunge. L'ultima parte consiste di un appendice riguardante le possibili applicazioni delle EC in crittografia e i principali algoritmi che si basano sullo studio di tali Curve.

Al fine di permettere una maggiore comprensione ed una migliore leggibilità, ho mantenuto alcuni termini nella lingua originale ed ho utilizzato, inoltre, i seguenti acronimi:

AES	*Advanced Encryption Standard*
AGM	*Arithmetic–Geometric–Mean*
DES	*Data Encryption Standard*
DH	*Diffie–Hellman*
DHP	*Diffie–Hellman Problem*
DL	*Discrete Logarithm*
DLP	*Discrete Logarithm Problem*

DSA	*Digital Signature Algorithm*
EC	*Elliptic Curve*
ECC	*Elliptic Curve Cryptography*
ECDH	*Elliptic Curve Diffie–Hellman*
ECDHP	*Elliptic Curve Diffie–Hellman Problem*
ECDLP	*Elliptic Curve Discrete Logarithm Problem*
ECIES	*Elliptic Curve Integrated Encryption Scheme*
ECMQV	*Elliptic Curve Menezes–Qu–Vanstone*
ECDSA	*Elliptic Curve Digital Signature Algorithm*
GMR	*Goldwasser–Micali–Rivest*
IFP	*Integer Factorization Problem*
RSA	*Rivest–Shamir–Adleman*
SEA	*Schoof–Elkies–Atkin*
SST	*Satoh–Skjrnaa–Taguchi*

Infine vorrei ringraziare il dott. G. Falcone per il suo prezioso contributo all'elaborazione del lavoro svolto.

Capitolo 1

INTRODUZIONE ALLA CRITTOGRAFIA

1.1 Concetti di base

Il termine Crittografia deriva dalla lingua greca ($\kappa\rho\upsilon\pi\tau\acute{o}\varsigma$, nascosto e $\gamma\rho\alpha\varphi\epsilon\iota\varsigma$, scrivere) ed è l'arte di nascondere i messaggi. Lo scopo della Crittografia è quindi quello di inventare codici oscuri; la sicurezza dei metodi usati è cresciuta col passare del tempo.

In questo contesto un ruolo molto importante è ricoperto dalla *chiave* del codice. Agli inizi della storia della crittografia venivano usati metodi in cui la chiave non giocava un ruolo essenziale; oggi, invece, ogni codice deve poter avere moltissime chiavi. La matematica è apparsa in maniera esplicita in crittografia solo dagli anni '40 in poi; essa serve sia per rompere che per costruire codici.

Il modello fondamentale della Crittografia prevede un *mittente* Alice, un *destinatario* Bob ed un *nemico* X. Il mittente vuole inviare un messaggio al destinatario in modo segreto; quindi, il nemico non deve avere alcuna possibilità di venire a conoscenza del messaggio.

Due metodi per ottenere ciò sono la crittografia e la *steganografia* (parola proveniente dal greco che significa "scrittura coperta").

Questo secondo modo consiste nel nascondere il messaggio all'interno di un altro messaggio o di una immagine così che il nemico non possa notare la sua trasmissione.

Nel caso della crittografia, il nemico nota la trasmissione del messaggio criptato, ma non ha modo di conoscere il messaggio originario; il mittente trasforma il *testo in chiaro* M in un *testo cifrato* C e trasmette C. Solo il destinatario deve essere in grado di riottenere M da C, quindi nasce l'esigenza di utilizzare una chiave segreta, nota solo al destinatario. La modifica del testo in chiaro è detta *cifratura*, il procedimento inverso è detto *decifratura*.

Il generico schema crittografico è quello della figura seguente:

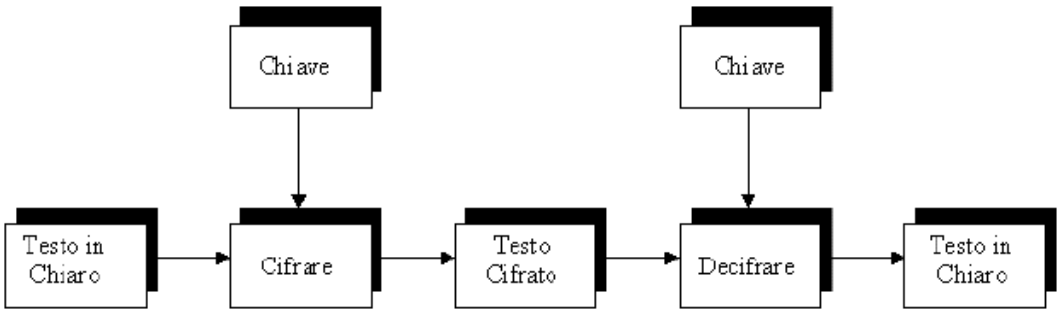

Figura 1.1: Generico schema crittografico

La Crittoanalisi, l'arte di rompere i codici segreti, assume nel nostro contesto un valore positivo, poiché permette di valutare la sicurezza offerta da un codice. Precisamente, la crittoanalisi mira a scoprire il valore dalla chiave utilizzata. La crittografia e la crittoanalisi formano insieme ciò che viene chiamata Crittologia.
I *codici a trasposizione* ed i *codici a sostituzione* costituiscono due metodi crittografici usati nell'antichità che sviluppano i prototipi di due tipi di cifrari.
Un esempio del primo tipo è la *scitala lacedemonica*, usata a Sparta circa 2500 anni fa. La scitala era un cilindro usato per trasmettere un messaggio in modo segreto. Il metodo consisteva nell'avvolgere intorno alla scitala un nastro, sul quale veniva scritto il messaggio in righe longitudinali. Finita l'operazione si svolgeva il nastro, che veniva mandato al destinatario. È chiaro che sul nastro le lettere che componevano le parole del messaggio risultavano permutate. Il destinatario era in possesso di un cilindro identico a quello usato; riavvolgendo il nastro su di esso, il messaggio si ricomponeva, così da essere letto. La chiave di questo codice è data dal diametro del cilindro. Un codice basato su questo principio si chiama a *trasposizione*.
Codici basati su un diverso principio sono quelli a sostituzione; in essi ogni lettera del testo in chiaro viene trasformata in un'altra lettera. Il prototipo storico di questi codici è il *Codice di Cesare*, basato sull'uso di un *alfabeto in chiaro* ed un *alfabeto segreto*.

Una qualsiasi lettera del testo in chiaro viene cifrata nella lettera dell'alfabeto segreto. L'alfabeto segreto usato da Cesare si ottiene shiftando circolarmente le lettere dell'alfabeto in chiaro di tre posizioni a sinistra. Si può notare facilmente che il Codice di Cesare è soggetto ad una facile crittoanalisi. Infatti, poiché ogni lettera viene trasformata sempre allo stesso modo, il cifrato conserva la frequenza con cui compare ogni data lettera. Schemi di questo tipo conservano ormai un mero valore enigmistico.
Gli algoritmi crittografici possono essere divisi in due classi principali: gli algoritmi a chiave simmetrica e quelli a chiave pubblica (o asimmetrica).

1.2 Crittografia a chiave simmetrica

Gli algoritmi di crittografia a chiave simmetrica sono in questo modo denominati perché utilizzano la stessa chiave sia per la cifratura che per la decifratura. Questo tipo di algoritmi storicamente sono nati per prima e fanno uso di tecniche di trasposizione e di sostituzione.
La Figura 1.2 rappresenta una comunicazione caratterizzata dall'utilizzo di una chiave simmetrica trasmessa attraverso un canale sicuro, che in realtà è molto difficile da realizzare[1].

[1]Si osservi che un tale canale potrebbe essere utilizzato direttamente per la trasmissione del messaggio.

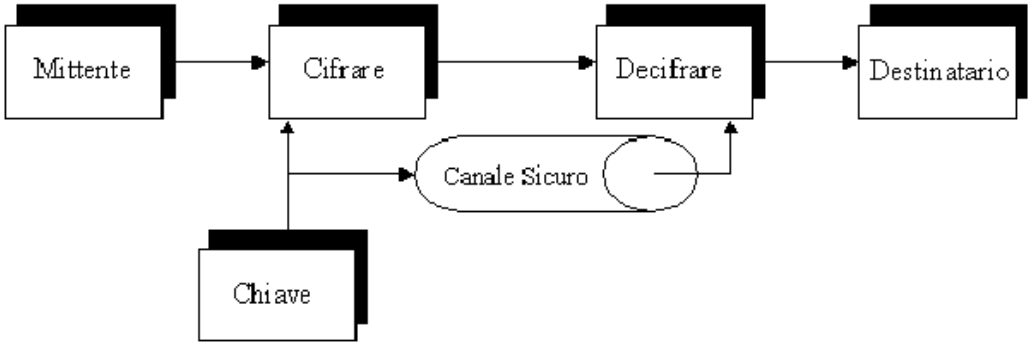

Figura 1.2: Generico schema crittografico a chiave simmetrica

Una delle caratteristiche più significative dei sistemi a chiave simmetrica è la veloce e facile implementazione. L'aspetto negativo riguarda il cosiddetto *Problema della Distribuzione delle chiavi*: se si hanno N entità che devono comunicare tra loro in modo sicuro attraverso un sistema crittografico a chiave simmetrica si deve fare in modo che ogni entità possegga $N-1$ chiavi differenti, una per ognuna delle altre entità; ciò significa che occorre essere in grado di generare e conservare $\frac{N(N-1)}{2}$ chiavi diverse. Naturalmente, per valori molto grandi di N la gestione del sistema diventa troppo complessa.

Un esempio di crittografia a chiave simmetrica è il *Codice Vernam*. In tale codice ogni dato è una stringa x definita sull'alfabeto $\mathbb{Z}_2 = \{0, 1\}$. I due interlocutori posseggono una medesima chiave k, una stringa definita sullo stesso alfabeto e di lunghezza pari a quella del messaggio che deve essere trasmesso. Il mittente trasmette $f(x) = x + k$, il destinatario, ricevuto il messaggio crip-

tato, dovrà solamente calcolare $f(x)+k = x$ visto che ci troviamo in $\mathbb{Z}_2$. Se un nemico intercetta il messaggio $f(x)$, ma non conosce k, non può risalire al messaggio x, perché

$$\forall \ y \ \exists \ k \ \text{ tale che } \ f(x) = x + k = y$$

quindi tale codice è inattaccabile.

1.3 Crittografia a chiave pubblica

Viste le difficoltà dovute alla distribuzione delle chiavi dei cifrari simmetrici, caratterizzati da un'unica chiave per criptare e decriptare, che deve essere protetta e allo stesso tempo distribuita agli utenti del sistema, nel 1976 i ricercatori Diffie ed Hellman proposero, nell'oggi celebre articolo *"New Directions in Cryptography"* [4], il concetto di crittografia a chiave pubblica.
Un sistema a chiave pubblica prevede che la chiave di decifratura non sia facilmente derivabile da quella di cifratura. Nella proposta di Diffie–Hellman l'algoritmo di cifratura $E(\cdot)$ e quello di decifratura $D(\cdot)$ devono soddisfare due condizioni:

1. $D(E(M)) = M$;
2. deve essere difficile dedurre $D(\cdot)$ da $E(\cdot)$.

Questo metodo funziona nel seguente modo: Alice, che vuole ricevere messaggi segreti da Bob, trova due funzioni E e D, che sod-

disfano le precedenti condizioni, parametrizzati dalla chiave per criptare e da quella per decriptare. La chiave di cifratura viene resa pubblica mentre quella di decifratura viene mantenuta segreta da Alice. Se Bob vuole spedire un messaggio M ad Alice, si procura la sua chiave pubblica, calcola $C = E(M)$, che è il messaggio cifrato, e glielo invia. A questo punto Alice, utilizzando la sua chiave privata, calcolerà $M = D(C)$.

La crittografia a chiave pubblica, quindi, richiede che l'utente possegga due chiavi: una *pubblica* usata per criptare i messaggi da chiunque voglia comunicare con lui ed una *privata* per decifrare i messaggi ricevuti. Queste due chiavi devono essere tra loro correlate, ma deve essere difficile poter calcolare la chiave privata da quella pubblica. Questa *difficoltà* si basa sulla intrattabilità di alcuni problemi matematici.

I problemi matematici maggiormente utilizzati per implementare sistemi crittografici a chiave pubblica sono:

Problema della fattorizzazione degli interi (IFP): dato un numero intero $n = pq$, con p e q numeri primi molto grandi, trovare p e q. Su tale problema si basa il sistema *RSA*;

Problema del Logaritmo Discreto (DLP): calcolare il logaritmo x di un elemento $b = a^x$ su un campo finito. Su tale problema si basa l'algoritmo di firma digitale *DSA* e il sistema di *ElGamal*;

Problema del Logaritmo Discreto su Curve Ellittiche: è una forma generalizzata del *DLP* sui punti di una curva ellittica. Su tale problema si basa l'analogo su Curva Ellittica di *ElGamal*.

La caratteristica dei sistemi crittografici a chiave pubblica è che sono molto più lenti di quelli a chiave simmetrica ma, allo stesso tempo, risolvono il Problema della Distribuzione delle Chiavi. Proprio per questo si preferisce l'utilizzo di un sistema che accomuni entrambi: gli schemi a chiave pubblica vengono utilizzati per lo scambio delle chiavi simmetriche (in genere molto più brevi del messaggio), la crittografia simmetrica viene utilizzata per cifrare i messaggi.

1.3.1 RSA e problema della fattorizzazione intera

Il crittosistema *RSA* deve il nome ai suoi inventori R. Rivest, A. Shamir e L. Adleman. Venne introdotto nel 1977 ed è uno dei sistemi a chiave pubblica maggiormente utilizzati per cifrare e per fornire la firma digitale. Prima di descriverlo diamo la seguente

1.3.1 Definizione. Il *problema della fattorizzazione intera* (IFP) è il seguente: dato un intero positivo n, trovare la sua fattorizzazione prima, ossia trovare i primi $p_1, p_2, .., p_k$ tali che

$$n = p_1^{e_1} p_2^{e_2} .. p_k^{e_k}$$

dove $e_i \geq 1$. □

L'*algoritmo RSA* si basa sul seguente problema:
dato un intero positivo n, prodotto di due numeri primi distinti p e q, un intero positivo e (*esponente di cifratura*) tale che $MCD(e, \phi) = 1$ (dove $\phi = (p-1)(q-1)$), ed un intero c, si trovi quell'unico intero m tale che $m^e \equiv c \ (mod\ n)$.
Il primo passaggio dell'RSA è la generazione della coppia di chiavi, che avviene secondo il seguente Algoritmo 1.3.2. La chiave pubblica consiste nella coppia di interi (n, e), mentre quella privata è l'intero d, chiamato *esponente di decifratura*, tale che $ed \equiv 1 \ (mod\ \phi)$.

1.3.2 Algoritmo. Generazione delle chiavi RSA

1. Selezionare due numeri primi grandi, distinti e casuali p e q;
2. Calcolare $n = pq$ e $\phi = (p-1)(q-1)$;
3. Selezionare un intero casuale e, $1 \leq e \leq \phi$, tale che $MCD(e, \phi) = 1$;
4. Usando l'Algoritmo esteso di Euclide[2], calcolare l'unico intero d, $1 \leq d \leq \phi$ tale che $ed \equiv 1 (mod\ \phi)$;

[2]Dati due interi non negativi a, b con $a \geq b$, determinare $d = MCD(a, b)$ e gli interi x e y che soddisfano $ax + by = d$.

(a) Se $b = 0$ allora $d = a$, $x = 1$, $y = 0$ e restituire (d, x, y).

(b) Consideriamo $x_2 = 1$, $x_1 = 0$, $y_2 = 0$, $y_1 = 1$.

(c) Mentre $b > 0$ allora:

- $q = \lfloor a/b \rfloor$, $r = a - qb$, $x = x_2 - qx_1$, $y = y_2 - qy_1$.
- $a = b$, $b = r$, $x_2 = x_1$, $x_1 = x$, $y_2 = y_1$ e $y_1 = y$.

5. La chiave pubblica è (n, e); quella privata è d. □

Immaginiamo adesso che Bob voglia cifrare un messaggio da inviare ad Alice. L'algoritmo per farlo è il seguente:

1.3.3 Algoritmo. Cifratura e decifratura RSA

Cifratura. Bob svolge le seguenti azioni:

1. Ottiene la chiave pubblica di Alice (n, e);
2. Rappresenta il messaggio come un intero m nell'intervallo $[0, n-1]$;
3. Calcola $c \equiv m^e (mod\ n)$;
4. Invia c ad Alice.

Decifratura. Per riottenere m da c, Alice svolge la seguente azione:

1. Usa la propria chiave privata d per riottenere $m \equiv c^d\ (mod\ n)$. □

Il meccanismo di *RSA* si basa sull'uguaglianza $m^{ed} \equiv m(mod\ n)$; ciò è conseguenza del fatto che, per costruzione, vale $ed = k\phi + 1$, per qualche k intero. Se $m \not\equiv 0(mod\ n)$, applicando il Teorema di Eulero[3] si ha:

$$m^{k\phi+1} \equiv (m^{\phi})^k m(mod\ n) \equiv (1)^k m(mod\ n) \equiv m(mod\ n),$$

(d) Considerare $d = a$, $x = x_2$, $y = y_2$ e restituire (d, x, y).

[3]Preso un intero $n \geq 2$, se $a \in \mathbb{Z}_n^*$ allora $a^{\varphi(n)} \equiv 1(mod\ n)$, dove φ è la funzione totiente di Eulero.

relazione banalmente verificata nel caso in cui $m \equiv 0(mod\ n)$.

1.3.4 Esempio

Generazione chiavi. Alice sceglie i primi $p = 13$ $q = 31$, e calcola $n = pq = 403$ e $\phi = (p-1)(q-1) = 360$. Successivamente sceglie $e = 7$ in modo che risulti $1 \leq e \leq 360$ e $MCD(e, \phi) = 1$. Usando l'Algoritmo esteso di Euclide trova d tale che $ed \equiv 1\ (mod\ \phi)$. Cioè considera il fatto che $1 = ed + k\phi$ e, quindi, attraverso la seguente tabella

q	r	x	y	a	b	x_2	x_1	y_2	y_1
–	–	–	–	360	7	1	0	0	1
51	3	1	−51	7	3	0	1	1	−51
2	1	−2	103	3	1	1	−2	−51	103
3	0	7	−360	1	0	−2	7	103	−360

determina $d = 103$. La chiave pubblica di Alice è $(n = 403, e = 7)$, mentre la chiave privata è $d = 103$.

Cifratura. Per cifrare un messaggio $m = 42$, Bob calcola

$$c = m^e\ (mod\ n) = 42^7\ (mod\ 403) = 354,$$

e lo invia ad Alice.

Decifratura. Per decifrare c, Alice calcola

$$m = c^d\ (mod\ n) = 354^{103}\ (mod\ 403) = 42.$$

□

Adesso vedremo come utilizzare gli stessi parametri $(n = pq, e)$ e d dell'algoritmo RSA per *firmare* un messaggio.
Supponiamo che Alice voglia firmare un messaggio m da inviare a Bob; lo schema di Firma e successiva Verifica può essere riassunto attraverso l'algoritmo seguente. Per fare questo Alice deve utilizzare una funzione $H(\cdot)$ detta *funzione hash* e come chiave pubblica e privata rispettivamente (n, e) e d. La funzione H è di pubblico dominio e serve soltanto a ridurre le dimensioni del messaggio, se queste fossero troppo grandi. In caso contrario, nulla vieta di assumere $H(m) = m$.

1.3.5 Algoritmo. Firma e verifica RSA

1. Alice genera la firma s del messaggio m:
   - Calcola $h = H(m)$;
   - Calcola $s = h^d (mod\ n)$;
   - s è la firma di Alice del messaggio m, che verrà trasmessa assieme al messaggio;
2. Bob riceve la coppia (s, m) e verifica la firma di Alice:
   - Calcola $h = H(m)$;
   - Calcola $h' = s^e (mod\ n)$;
   - Se $h = h'$ la firma è *accettata*, altrimenti è *rifiutata*. □

La verifica della firma si basa dunque anch'essa sul fatto che $h^{ed} \equiv h(mod\ n)$.

1.3.6 Esempio

Generazione Firma. Consideriamo i valori utilizzati nell'esempio (1.3.4) e supponiamo $H(m) = m$.
Per generare la sua firma, Alice deve calcolare

$$s = h^d\ (mod\ n) = 42^{103}\ (mod\ 403) = 393.$$

A questo punto Alice può trasmette la sua firma con il relativo messaggio: $(393, 42)$.

Verifica Firma. Ricevuta la coppia $(393, 42)$, per verificarne la validità Bob calcola rispettivamente

$$h = H(m) = m \ \text{ e } \ h' = s^e\ (mod\ n) = 393^7\ (mod\ 403) = 42.$$

Essendo $h = h'$, Bob accetta s come firma di Alice del messaggio m. □

1.3.7 Osservazione. La sicurezza dell'RSA è tutta basata sull'intrattabilità del problema di determinare p e q da n, dal momento che, se il nemico viene a conoscenza di p e q, egli può, allo stesso modo di Bob, calcolare ϕ e quindi d.

Un metodo molto utilizzato per risolvere il problema della fattorizzazione è il seguente:

Algoritmo ρ di Pollard per RSA.

Sia $f : S \longrightarrow S$ una funzione random, dove S è un insieme finito di cardinalità n. Sia x_0 un elemento casuale dell'insieme S; consideriamo la sequenza $x_0,\ x_1,\ x_2\ \ldots$ definita in modo che $x_{i+1} = f(x_i)$ per $i \geq 0$.

Essendo S un insieme finito, per il *Principio dei cassetti*[4], troveremo due indici distinti i e j tali che $x_i = x_j$, cioè avremo una *collisione.*

Una tecnica utilizzata per la determinazione di una collisione è il cosidetto *ciclo di Floyd.* L'idea di Floyd consiste nel calcolare le coppie (x_i, x_{2i}) con $i = 1, 2, \cdots$ fino ad ottenere $x_i = x_{2i}$.

Sia p un numero primo tale che p/n. L'algoritmo ρ di Pollard per la fattorizzazione considera la sequenza di interi $x_0,\ x_1,\ \cdots$ definita nel modo seguente: $x_0 = 2$, $x_{i+1} = f(x_i) = x_i^2 + 1\ (mod\ p)$ per $i \geq 0$. Utilizzando il ciclo di Floyd si determinano x_i e x_{2i} tale che $x_i \equiv x_{2i}\ (mod\ p)$. Essendo che p/n e $p/(x_{2i} - x_i)$ allora avremo che $MCD(x_{2i} - x_i, n) > 1$.

L'algoritmo completo è il seguente:

1.3.8 Algoritmo. ρ di Pollard per la fattorizzazione degli interi

Dato un numero intero non primo n, determinare un fattore d di n.

[4]Se n oggetti sono inseriti in m cassetti, e $n > m$, allora almeno un cassetto deve contenere più di un oggetto.

1. Siano $a = 2$ e $b = 2$;

2. Per $i = 1, 2, \ldots$ svolgere le seguenti azioni:

   - Calcolare $a = a^2 + 1 \ (mod\ n)$ e $b = a^2 + 1 \ (mod\ n)$;
   - Calcolare $d = MCD(|a - b|, n)$;
   - Se $1 < d < n$, d è un fattore di n. □

1.3.9 Esempio

Sia $n = 1763$.

La seguente tabella mostra i valori di a, b e d ottenuti seguendo i vari passaggi del secondo punto dell'algoritmo 1.3.8.

a	b	d
5	26	1
26	677	1
677	1713	1
1713	738	1
738	1641	43

Abbiamo trovato che uno dei due fattori di $n = 1763$ è 43, quindi l'altro sarà $1763/43 = 41$. □

La fattorizzazione degli interi è uno dei problemi più antichi della matematica, infatti, molte tecniche utilizzate nei moderni algorit-

mi di fattorizzazione risalgono agli antichi Greci (come ad esempio l'algoritmo di Euclide per calcolare il MCD e il Crivello di Eratostene). Anche i grandi matematici del '700 e dell'800 (Fermat, Eulero e Legendre) hanno dato il loro contributo. I progressi maggiori però si sono avuti negli ultimi trent'anni dopo l'introduzione della crittografia a chiave pubblica.

Vi sono due tipi di algoritmi per risolvere l'IFP, gli algoritmi *special–purpose* e quelli *general–purpose*. I primi sfruttano particolari caratteristiche del numero n da fattorizzare, mentre i secondi dipendono dalla dimensione di n.

Un esempio di algoritmo special–purpose è costituito dal metodo di fattorizzazione basato sulle curve ellittiche ideato da H. Lenstra nel 1985. Tale algoritmo dipende dalle dimensione dei fattori primi di n.

L'idea di base degli algoritmi general–purpose è quella della *"differenza dei quadrati"* e consiste nel trovare due interi x e y tali che

1. $x^2 \equiv y^2 (mod\ n)$;
2. $x \neq \pm y (mod\ n)$;

dove n è il numero da fattorizzare.

La prima può essere riscritta come $(x-y)(x+y) \equiv 0\ (mod\ n)$ (ossia $n \mid (x-y)(x+y)$) e la seconda implica che $(x-y)$ e $(x+y)$ non sono divisibili per n; ciò significa che il $MCD(x-y, n)$ e il

$MCD(x+y, n)$ sono due fattori non banali di n.
Il funzionamento di questo secondo tipo di algoritmi si basa sull'idea di trovare un sistema lineare di equazioni la cui soluzione porti ad equazioni del primo tipo, verificando le condizioni del secondo tipo.

1.3.2 Sistemi basati sul Logaritmo Discreto

La sicurezza di molti sistemi crittografici si basa sull'intrattabilità del Problema del Logaritmo Discreto.
Particolari esempi di sistemi di questo tipo sono il protocollo di Diffie–Hellmann per la condivisione di una chiave segreta, il crittosistema ElGamal e l'algoritmo di firma digitale DSA.
Innanzitutto diamo la seguente definizione:

1.3.10 Definizione. Sia G un gruppo ciclico finito di ordine n. Sia α un generatore di G e $\beta \in G$. Il *Logaritmo Discreto* (DL) di β in base α, indicato con $\log_\alpha \beta$, è l'unico intero x, $0 \leq x \leq n-1$, tale che $\alpha^x = \beta$. □

Gruppi di particolare interesse in crittografia per il Problema del Logaritmo Discreto sono i gruppi moltiplicativi $\mathbb{Z}_p^*$ degli interi modulo p, con p primo. Quindi:

1.3.11 Definizione. Il *Problema del Logaritmo Discreto* (DLP)

è il seguente: dato un numero primo p, un generatore α di $\mathbb{Z}_p^*$, trovare l'intero x, $0 \leq x \leq p-2$, tale che $\alpha^x \equiv \beta \ (mod\ p)$. □

L'utilizzo del DLP in crittografia deriva dalla difficoltà di calcolare il logaritmo discreto.
I più importanti algoritmi che si basano su questo problema sono:

Protocollo Diffie–Hellman.

Il protocollo Diffie–Hellman venne presentato, nello stesso articolo che introdusse la Crittografia a Chiave Pubblica, da Diffie ed Hellman nel 1976 [4]. Esso è un protocollo che consente ad Alice e a Bob di condividere una chiave segreta K, senza la necessità di un canale sicuro. Tale protocollo funziona in questo modo:

1.3.12 Protocollo Diffie–Hellman.

1. Vengono fissati un numero primo p ed un generatore α di $\mathbb{Z}_p^*$;
2. Alice seleziona un intero casuale $1 < x < p-1$ e calcola $X = \alpha^x (mod\ p)$;
3. Alice invia X a Bob;
4. Bob seleziona un intero casuale $1 < y < p-1$ e calcola $Y = \alpha^y (mod\ p)$;
5. Bob invia Y ad Alice;
6. Alice riceve Y e calcola $K_X = Y^x (mod\ p) = (\alpha^y)^x (mod\ p)$;

7. Bob riceve X e calcola $K_Y = X^y (mod\ p) = (\alpha^x)^y (mod\ p)$;

8. Alice e Bob condividono la chiave $K = K_X = K_Y$. □

1.3.13 Esempio

Sia $p = 257$ e sia $\alpha = 3$ un generatore di $\mathbb{Z}^*_{257}$.

Alice considera l'intero casuale $x = 15$ e calcola $X = \alpha^x\ (mod\ p) = 3^{15}\ (mod\ 257) = 83$ da inviare a Bob.

Una volta ricevuto X, Bob sceglie l'intero casuale $y = 8$ e calcola $Y = \alpha^y\ (mod\ p) = 3^8\ (mod\ 257) = 136$ e lo invia ad Alice.

Adesso, ricevuti rispettivamente Y e X, condividono la chiave comune $K = K_X = K_Y$. Infatti:

$$K_X = Y^x\ (mod\ p) = 136^{15}\ (mod\ 257) = 17$$

e

$$K_Y = X^y\ (mod\ p) = 83^8\ (mod\ 257) = 17.$$

□

1.3.14 Osservazione. Si osservi che per determinare la chiave segreta K, il nemico non deve necessariamente risolvere il DLP ma il seguente problema:

1.3.15 Definizione. Il *Problema di Diffie–Hellman* (DHP) è il seguente: dato un numero primo p, un generatore α di Z_p^* e gli elementi $\alpha^a (mod\ p)$ e $\alpha^b (mod\ p)$ trovare $\alpha^{ab} (mod\ p)$. □

Sebbene sia facile dimostrare che la risoluzione del DLP in $\mathbb{Z}_p^*$ implichi la risoluzione del DHP, non è ancora stato provato che valga il contrario. □

Crittosistema di ElGamal.

La sicurezza di tale sistema si basa sull'intrattabilità del Problema del Logaritmo Discreto. Questo algoritmo costituisce il primo crittosistema basato sul DLP. Tale schema viene utilizzato anche per implementare uno schema di firma digitale.
Gli algoritmi seguenti descrivono rispettivamente la generazione delle chiavi e lo schema di Cifratura-Decifratura del sistema ElGamal.

1.3.16 Algoritmo. Generazione delle chiavi per ElGamal
Ogni entità crea una chiave pubblica e una corrispondente chiave privata. Alice svolgerà le seguenti azioni:

1. Sceglie un grande numero primo casuale p e un generatore α del gruppo moltiplicativo $\mathbb{Z}_p^*$ degli interi modulo p;

2. Seleziona un intero casuale a tale che $1 \leq a \leq p-2$ e calcola $\alpha^a (mod\ p)$;

3. La chiave pubblica di Alice è (p, α, α^a) mentre quella privata è a. □

1.3.17 Algoritmo. Schema ElGamal

Bob cripta un messaggio m per Alice, che lo decripterà.

1. *Cifratura.* Bob esegue le seguenti azioni:
   - Ottiene la chiave pubblica di Alice (p, α, α^a);
   - Rappresenta il messaggio come un intero m compreso tra 0 e $p-1$;
   - Seleziona un intero casuale k, $1 \leq k \leq p-2$;
   - Calcola $\gamma = \alpha^k (mod\ p)$ e $\delta = m(\alpha^a)^k (mod\ p)$;
   - Invia $c = (\gamma, \delta)$ ad Alice.
2. *Decifratura.* Per ottenere m da c, Alice esegue le seguenti azioni:
   - Usa la chiave privata a per calcolare $\gamma^{p-1-a} (mod\ \ p)$ (osserviamo che $\gamma^{p-1-a} = \gamma^{-a} = \alpha^{-ak}$);
   - Otteniamo m calcolando $(\gamma^{-a})\delta (mod\ p)$; □

La decifratura di tale algoritmo si basa sull'osservazione che

$$(\gamma^{-a})\delta \equiv \alpha^{-ak} m \alpha^{ak} \equiv m (mod\ p).$$

Il processo di cifratura è non–deterministico in quanto il messaggio cifrato c dipende dal messaggio in chiaro m e dal valore

casuale k, scelto da Bob. Questo significa che uno stesso messaggio può essere cifrato in vari modi differenti a secondo del valore attribuito a k.

1.3.18 Esempio

Generazione delle chiavi. Alice seleziona il primo $p = 2357$ e un generatore $\alpha = 2$ di $\mathbb{Z}^*_{2357}$. Seleziona la chiave privata $a = 1751$ e calcola

$$\alpha^a \ (mod\ p) = 2^{1751} \ (mod\ 2357) = 1185.$$

La chiave pubblica di Alice è: $(p = 2357, \alpha = 2, \alpha^a = 1185)$.

Cifratura. Per cifrare il messaggio $m = 2035$, Bob seleziona un intero casuale $k = 1520$ e calcola

$$\gamma = 2^{1520} \ (mod\ 2357) = 1430$$

e

$$\delta = 2035 \cdot 1185^{1520} \ (mod\ 2357) = 697.$$

Bob invia $\gamma = 1430$ e $\delta = 697$ ad Alice.

Decifratura. Per decifrare, Alice ricava

$$\gamma^{p-1-a} = 1430^{605} \ (mod\ 2357) = 872,$$

ed infine m calcolando

$$m = 872 \cdot 697 \ (mod\ 2357) = 2035.$$

□

Allo stesso modo, per un dato messaggio m, vi sono un gran numero di firme digitali.

L'algoritmo che segue descrive il processo con il quale Alice usa la propria chiave privata a per firmare il messaggio m e il modo in cui Bob verifica tale firma utilizzando la chiave pubblica (p, α, α^a) di Alice.

Come nel caso dello schema RSA, entrambi possono usare una funzione hash $H(\cdot)$ di pubblico dominio nel caso in cui il messaggio sia troppo lungo.

1.3.19 Algoritmo. Firma e verifica ElGamal.

1. Alice genera la firma (r, s) del messaggio m.

   - Seleziona un intero casuale $1 \leq k \leq p-2$ con $MCD(k, p-1) = 1$;
   - Calcola $r = \alpha^k (mod\ p)$;
   - Calcola $s = k^{-1}[H(m) - ar](mod\ p-1)$;
   - La coppia (r, s) è la firma di Alice del messaggio m;

2. Bob verifica la firma di Alice (r, s) sul messaggio m

   - Verifica che $1 \leq r \leq p-1$, altrimenti rifiuta la firma;
   - Calcola $v_1 = \alpha^{ar} r^s (mod\ p)$;
   - Calcola $v_2 = \alpha^{H(m)} (mod\ p)$;
   - Accetta la firma se e solo se $v_1 = v_2$ □

La verifica della firma si basa sulle seguenti considerazioni:
se s è stato veramente generato da Alice allora

$$ks \equiv (H(m) - ar)(mod\ (p-1)),$$

che equivale a scrivere $H(m) \equiv ar + ks(mod\ (p-1))$. Ciò significa che $\alpha^{H(m)} \equiv \alpha^{ar+ks} \equiv (\alpha^a)^r r^s (mod\ p)$ e quindi $v_1 = v_2$.

1.3.20 Esempio

Generazione Firma. Consideriamo i valori utilizzati nell'esempio 1.3.18 e supponiamo $H(m) = m$.
Per generare la sua firma, Alice seleziona un intero casuale k tale che $MCD(k, p-1) = 1$, ad esempio $k = 5$ e calcola

$$r = \alpha^k\ (mod\ p) = 2^5\ (mod\ 2357) = 32$$

e

$$s = k^{-1} \cdot (H(m) - ar)\ (mod\ p-1) = 5^{-1} \cdot 191\ (mod\ 2356) =$$

$$= (-471) \cdot (-53997)\ (mod\ 2356) = 1923.$$

Osserviamo infatti che: se $a \in \mathbb{Z}_n$. Allora a è invertibile se e solo se $MCD(a, n) = 1$. Per esempio $4 \in \mathbb{Z}_9$ è invertibile, $4^{-1} = 7$ perché $4 \cdot 7 \equiv 1\ (mod\ 9)$.
A questo punto Alice può trasmette la sua firma: $(32, 1923)$.

Verifica Firma. Ricevuta la coppia $(32, 1923)$, per verificarne la validità Bob calcola rispettivamente

$$v_1 = \alpha^{ar} \cdot r^s\ (mod\ p) = 2^{1751 \cdot 32} \cdot 32^{1923}\ (mod\ 2357) =$$

$$= 2^{1844} \cdot 32^{1923} \ (mod\ 2357) = 2213 \cdot 1661 \ (mod\ 2357) = 1230$$

$$v_2 = \alpha^{H(m)} \ (mod\ p) = 2^{2035} \ (mod\ 2357) = 1230.$$

Essendo $v_1 = v_2$, Bob accetta (r, s) come firma di Alice del messaggio m. □

Algoritmo DSA (Digital Signature Algorithm).
Il DSA è un nuovo schema di firma digitale che rappresenta una variante dello schema di firma ElGamal. Gli algoritmi seguenti indicano come avviene la generazione della coppia di chiavi e lo schema di firma/verifica DSA.

1.3.21 Algoritmo. Generazione delle chiavi per DSA

1. Selezionare un grande numero primo q;
2. Selezionare un numero primo p tale che $q|p-1$;
3. Selezionare un elemento $\alpha \in \mathbb{Z}_p^*$ di ordine q;
4. Selezionare un intero casuale $1 \leq x \leq q-1$;
5. Calcolare $y = \alpha^x (mod\ p)$;
6. La Chiave Pubblica è (p, q, α, y); quella Privata x. □

1.3.22 Esempio

Generazione delle chiavi. Alice seleziona i primi $p = 107$ e

$q = 53$ $(q/p-1)$ ed $\alpha = 25 \in \mathbb{Z}_{107}^*$ di ordine 53. Seleziona l'intero casuale $x = 32$ e calcola

$$y = \alpha^x \ (mod\ p) = 25^{32} \ (mod\ 107) = 41.$$

La chiave pubblica di Alice è $(p = 107, q = 53, \alpha = 25, y = 41)$, mentre quella privata è $x = 32$. □

Nel seguente algoritmo si ipotizza che Alice firmi, con la propria chiave privata, un messaggio m e che Bob successivamente verifichi tale firma, usando la chiave pubblica di Alice. Entrambi utilizzano la funzione hash $H(m)$.

1.3.23 Algoritmo. Firma e verifica DSA

1. Alice genera la firma (r, s) del messaggio m
   - Seleziona un intero casuale $1 \leq k \leq q-1$;
   - Calcola $r = (\alpha^k (mod\ p))(mod\ q)$;
   - Calcola $k^{-1}(mod\ q)$;
   - Calcola $s = k^{-1}[H(m) + xr](mod\ q)$;
   - La coppia (r, s) è la firma di Alice del messaggio m;
2. Bob verifica la firma di Alice (r, s) sul messaggio m
   - Verifica che $1 \leq r \leq q-1$, altrimenti rifiuta la firma;
   - Calcola $w = s^{-1}(mod\ q)$;

- Calcola $u_1 = wH(m)(mod\ q)$ e $u_2 = rw(mod\ q)$;
- Calcola $v = (\alpha^{u_1} y^{u_2}(mod\ p))(mod\ q)$;
- Accetta la firma se e solo se $v = r$. □

I vari passi di tale algoritmo ci permettono di dimostrare come il meccanismo firma/verifica DSA funzioni correttamente: se (r, s) è la firma leggittima di Alice sul messaggio m allora vale $H(m) \equiv -xr + ks(mod\ q)$ e quindi, moltiplicando per w, $wH(m) + xrw \equiv k(mod\ q)$. L'ultima congruenza può essere riscritta come $u_1 + xu_2 \equiv k(mod\ q)$, pertanto se si eleva ad entrambi i membri si ottiene $(\alpha^{u_1} y^{u_2}(mod\ p))(mod\ q) = \alpha^k(mod\ p)$, cioè $v = r$.

1.3.24 Esempio

Generazione Firma. Consideriamo i valori utilizzati nell'esempio 1.3.22 e supponiamo $H(m) = m = 23$.
Per generare la sua firma, Alice seleziona l'intero casuale $k = 19$ e calcola

$$r = (\alpha^k\ (mod\ p))(mod\ q) = (25^{19}\ (mod\ 107))(mod\ 53) = 35,$$

$$k^{-1}\ (mod\ q) = 19^{-1}\ (mod\ 53) = -39\ (mod\ 53) = 14$$

e

$$s = k^{-1}(H(m) + xr)\ (mod\ q) = 14 \cdot (23 + 32 \cdot 35)\ (mod\ 53) =$$

$$= 14 \cdot 30\ (mod\ 53) = 49$$

A questo punto Alice può trasmette la sua firma: $(35, 49)$.

Verifica Firma. Ricevuta la coppia $(35, 49)$, per verificarne la validità Bob calcola rispettivamente

$$w = s^{-1}\ (mod\ q) = 49^{-1}\ (mod\ 53) = -40\ (mod\ 53) = 13,$$

$$u_1 = w \cdot H(m)\ (mod\ q) = 13 \cdot 23\ (mod\ 53) = 34,$$

$$u_2 = r \cdot w\ (mod\ q) = 35 \cdot 13\ (mod\ 53) = 31$$

ed infine

$$v = (\alpha^{u_1} \cdot y^{u_2}\ (mod\ 107))(mod\ q) = (25^{34} \cdot 41^{31}\ (mod\ 107))(mod$$

$$53) = (52 \cdot 48\ (mod\ 107))(mod\ 53) = 35\ (mod\ 53) = 35$$

Essendo $v = r$, Bob accetta (r, s) come firma di Alice del messaggio m. □

Algoritmi per risolvere il DLP.

Se nell'IFP la dimensione dell'input era quella dell'intero n da fattorizzare, nel DLP la dimensione dell'input dipende dal numero di punti N del gruppo G considerato. Se ad esempio $G = \mathbb{Z}_p^*$, con p primo, si ha che $N = p - 1$.

Come per l'IFP, anche per il DLP vi sono due tipi di algoritmi di risoluzione. Gli algoritmi *special–purpose* sfruttano particolari caratteristiche del numero N, mentre il running time[5] degli algoritmi *general–purpose* dipende solo dalle dimensioni di N.

[5]Il *running time* di un algoritmo, con un particolare input, è uguale al numero di passi elementari da eseguire.

I più veloci algoritmi di tipo *general–purpose* per risolvere il DLP si basano sul metodo *index–calculus*.

1.3.25 Algoritmo index–calculus per il logaritmo discreto in un gruppo ciclico

Dato un generatore α di un gruppo ciclico G di ordine n ed un elemento $\beta \in G$ determinare il logaritmo discreto $y = \log_\alpha \beta$.

1. Scegliere un sottoinsieme $S = \{p_1, p_2, \cdots, p_t\}$ di G tale che un significativo numero di elementi di G possano essere espressi come prodotto di elementi di S.

2. Determinare relazioni lineari che portino a logaritmi di elementi di S nel modo seguente:

   - Scegliere un intero casuale k, $0 \leq k \leq n-1$ e calcolare α^k.
   - Scrivere α^k come prodotto di elementi di S:
   $$\alpha^k = \prod_{i=1}^{t} {p_i}^{c_i}, \quad c_i \geq 0.$$
   Se ciò è possibile, calcolare i logaritmi determinando
   $$k \equiv \sum_{i=1}^{t} c_i \log_\alpha p_i \ \ mod\ n.$$
   - Ripetere il punto 2 fino a quando non si ottengono $t + c$ relazioni lineari (c è un piccolo intero positivo tale che il sistema di $t + c$ equazioni ha un'unica soluzione con grande probabilità).

3. Lavorando modulo n, risolvere il sistema lineare delle $t + c$ equazioni (con t non noto) determinate nel punto 2 per ottenere i valori di $\log_\alpha p_i$, $1 \leq i \leq t$.

4. Calcolare y:

   - Selezionare un intero casuale k, $0 \leq k \leq n-1$ e calcolare $\beta \cdot \alpha^k$.
   - Scrivere $\beta \cdot \alpha^k$ come prodotto di elementi di S:

$$\beta \cdot \alpha^k = \prod_{i=1}^{t} {p_i}^{d_i}, \quad d_i \geq 0.$$

   Se ciò non è possibile, ripetere il punto 4. Altrimenti determinare $y = \log_\alpha \beta = -k + \sum_{i=1}^{t} d_i \log_\alpha p_i \ \ mod\ n$; e restituire y. □

1.3.26 Esempio

Sia $p = 229$. L'elemento $\alpha = 6$ è un generatore di $\mathbb{Z}_{229}$ di ordine $n = 228$. Consideriamo $\beta = 13$. Allora $\log_6 13$ è calcolato come segue:

1. Consideriamo come base i primi 5 primi: $S = \{2, 3, 5, 7, 11\}$.

2. Consideriamo le seguenti 6 selazioni:

$$6^{100} \ \ mod\ 229 = 180 = 2^2 \cdot 3^2 \cdot 5$$

$$6^{18} \ \ mod\ 229 = 176 = 2^4 \cdot 11$$

$$6^{12} \ \ mod\ 229 = 165 = 3 \cdot 5 \cdot 11$$

$$6^{62} \ \ mod\ 229 = 154 = 2 \cdot 7 \cdot 11$$

$$6^{143} \ \ mod\ 229 = 198 = 2 \cdot 3^2 \cdot 11$$

$$6^{206} \ \ mod\ 229 = 210 = 2 \cdot 3 \cdot 5 \cdot 7.$$

Queste relazioni possono essere riscritte, considerando i logaritmi degli elementi di S, nel modo seguente:

$$100 \equiv 2\log_6 2 + 2\log_6 3 + \log_6 5 \ (mod\ 228)$$

$$18 \equiv 4\log_6 2 + \log_6 11 \ (mod\ 228)$$

$$12 \equiv \log_6 3 + \log_6 5 + \log_6 11 \ (mod\ 228)$$

$$62 \equiv \log_6 2 + \log_6 7 + \log_6 11 \ (mod\ 228)$$

$$143 \equiv \log_6 2 + 2\log_6 3 + \log_6 11 \ (mod\ 228)$$

$$206 \equiv \log_6 2 + \log_6 3 + \log_6 5 \log_6 7 \ (mod\ 228).$$

3. Risolvendo il sistema lineare di sei equazioni in cinque indeterminate (i logaritmi $x_i = \log_6 p_i$) troviamo le soluzioni $\log_6 2 = 21$, $\log_6 3 = 208$, $\log_6 5 = 98$, $\log_6 7 = 107$ e $\log_6 11 = 162$.

4. Consideriamo l'intero $k = 77$. Quindi $\beta \cdot \alpha^k = 13 \cdot 6^{77} \ \ mod\ 229 = 147 = 3 \cdot 7^2$. Da ciò segue che

$$\log_6 13 = (\log_6 3 + 2\log_6 7 - 77) \ \ mod\ 228 = 117.$$

□

Un altro modo di procedere per risolvere il problema del logaritmo discreto su un gruppo finito G di ordine n è quello di utilizzare il metodo Baby step/Giant step.
L'algoritmo afferma che:

Sia $m = \lceil \sqrt{n} \rceil$, dove n è l'ordine di α. Se $\beta = \alpha^x$, allora si può scrivere $x = im + j$, dove $0 \leq i, j < m$. Dunque, $\alpha^x = \alpha^{im}\alpha^j$, che implica $\beta(\alpha^{-m})^i = \alpha^j$.

Ciò permette di sviluppare il seguente algoritmo per la determinazione di x.

1.3.27 Algoritmo Baby step/Giant step per il logaritmo discreto

Dato un generatore α di un gruppo ciclico finito G di ordine n ed un elemento $\beta \in G$ determinare il logaritmo discreto $x = \log_\alpha \beta$.

1. Considerare $m = \lceil \sqrt{n} \rceil$;
2. Costruire la tabella (j, α^j), con $0 \leq j < m$;
3. Calcolare α^{-m} e considerare $\gamma = \beta$;
4. Per i da 0 a $m - 1$:
    - Costruire la tabella $(i, \gamma = \beta\alpha^{-mi})$;
    - verificare se γ si trova all'interno della seconda colonna della tabella costruita nel punto 2;

- Se $\gamma = \alpha^j$, si può calcolare $x = im + j$. □

1.3.28 Esempio

Sia $p = 113$. L'elemento $\alpha = 3$ è un generatore di $\mathbb{Z}_{113}^*$ di ordine $n = 112$. Consideriamo $\beta = 57$. Allora $\log_3 57$ è calcolato nel modo seguente:

1. Consideriamo $m = \lceil \sqrt{112} \rceil = 11$;

2. Costruiamo una tabella al variare di j per $0 \leq j < 11$:

j	0	1	2	3	4	5	6	7	8	9	10
$3^j \mod 113$	1	3	9	27	81	17	51	40	7	21	63

3. Usando l'algoritmo esteso di Euclide calcolare $\alpha^{-1} = 3^{-1} \mod 113 = 38$ e allora calcolare $\alpha^{-m} = 38^{11} \mod 113 = 58$.

4. Successivamente, considerando $\gamma = \beta\alpha^{-mi} \mod 113$ al variare di i, costruire la seguente tabella:

i	0	1	2	3	4	5	6	7	8	9
$\gamma = 57 \cdot 58^i \mod 113$	57	29	100	37	112	55	26	39	2	3

Abbiamo trovato $\beta\alpha^{-9m} = 3 = \alpha^1$, $\beta = \alpha^{100}$ e, quindi, $\log_3 57 = 100$. □

1.3.3 Crittografia su Curva Ellittica

Abbiamo visto che il Logaritmo Discreto è definibile su un qualsiasi gruppo ciclico. La crittografia basata su Curve Ellittiche (ECC) si basa sulla risoluzione del Logaritmo Discreto nel gruppo dei punti di una curva ellittica definita su un campo finito.
La definizione rigorosa di curva ellittica verrà data nel capitolo successivo, adesso possiamo dire che una *curva ellittica* $\mathcal{E}$ *definita sul campo finito* $GF(p^t)$, con p primo, è definita dall'equazione

$$y^2 = x^3 + ax + b \tag{1.1}$$

dove i coefficienti $a, b \in GF(p^t)$ verificano $4a^3 + 27b^2 \neq 0$. La coppia (x, y), con x, y nella chiusura algebrica[6] di $GF(p^t)$ che soddisfano l'equazione (1.1), è un punto della curva $\mathcal{E}$. L'insieme di tutti i punti di $\mathcal{E}$ in cui x, y appartengono a $GF(p^t)$ (detti *punti razionali su* $GF(p^t)$) si indica con $\mathcal{E}(GF(p^t))$.
Come si vedrà nel capitolo seguente, è possibile definire l'operazione di addizione $(+)$ fra due punti P e Q di una curva ellittica, utilizzando lo speciale *punto all'infinito* Θ come elemento identità. Si dimostra inoltre che $\mathcal{E}(GF(p^t))(+)$ forma un gruppo additivo abeliano. Ciò significa che è possibile definire il Logaritmo Discreto sui sottogruppi ciclici di $\mathcal{E}(GF(p^t))$.
Possiamo adesso illustrare l'idea di base dell'ECC, realizzando

[6] La *chiusura algebrica* di un campo K si indica con $\overline{K}$ ed è tale che: $\overline{K}$ è estensione algebrica di K e $\overline{K}$ è algebricamente chiusa.

un analogo del sistema ElGamal su una curva ellittica.

1.3.4 ElGamal su curve ellittiche

Sia $\mathcal{E}$ una curva ellittica definita su $GF(p^t)$ e sia P un punto in $\mathcal{E}(GF(p^t))$ con ordine primo n. Consideriamo il sottogruppo ciclico generato da P

$$\langle P \rangle = \{\Theta, P, 2P, ..., (n-1)P\}$$

L'equazione di $\mathcal{E}$, il punto P e i primi p e n sono conosciuti da tutti. La coppia di chiavi (Pubblica e Privata) viene generata in questo modo:

- la Chiave Privata è un intero d scelto a caso nell'intervallo $[1, n-1]$;
- la Chiave Pubblica è il punto $Q = dP$ appartenente a $\langle P \rangle$;

Il problema di determinare d conoscendo P e Q è proprio *l'Elliptic Curve Discrete Logarithm Problem* (ECDLP).
L'algoritmo di ElGamal su curva ellittica venne proposto da Koblitz nel 1986. A differenza di quello originale, il messaggio non viene rappresentato come un intero, ma come un punto M in $\mathcal{E}(GF(p^t))$ mediante una codifica di pubblico dominio. Il funzionamento è analogo a quello originario di ElGamal, con le dovute modifiche: il punto M viene cifrato sommando ad esso il punto kQ, dove k è un intero casuale e Q è la chiave pubblica del

destinatario. Il mittente quindi spedisce i punti kP ed $M + kQ$. Il destinatario, invece, utilizza la propria chiave privata d per recuperare da essi il messaggio M.

Infatti: $d(kP) = k(dP) = kQ$ e quindi $M + kQ - d(kP) = M + kQ - kQ = M$.

Il procedimento può essere rappresentato dai seguenti algoritmi:

1.3.29 Algoritmo. Cifratura ElGamal su EC

Partendo dal messaggio M e dalla chiave pubblica Q, per ottenere il testo cifrato (C_1, C_2) occorre:

1. Selezionare a caso $1 \leq k \leq n - 1$;
2. Calcolare $C_1 = kP$;
3. Calcolare $C_2 = M + kQ$;
4. Restituire (C_1, C_2). □

1.3.30 Algoritmo. Decifratura ElGamal su EC

Partendo dalla chiave privata d e dal cifrato (C_1, C_2) per determinare il messaggio m occorre:

1. Calcolare $M = C_2 - dC_1$;
2. Recuperare m da M e restituire m. □

1.3.31 Esempio

Cifratura. Sia $y^2 = x^3 + 2x + 4$ su $\mathbb{Z}_7$ l'equazione che definisce la curva ellittica $\mathcal{E}$. Supponiamo di voler cifrare il messaggio $m = 110 = 6$. Da tale numero ricaviamo il punto $M = (6, 1) \in \mathcal{E}$, infatti: se $x = 6$ allora $y^2 = 6^3 + 2 \cdot 6 + 4 \equiv 1 \ (mod\ 7)$, quindi $y = \pm 1$.

Scelto $k = 2$ e considerando la coppia $(P = (0, 2), Q = (6, 6))$ per cifrare M basta calcolare $C_1 = kP = 2(0, 2)$ e $C_2 = M + kQ = (6, 1) + 2(6, 6)$.

Iniziamo con il calcolare $C_1 = (x_1, y_1)$. Per la legge di gruppo, che svilupperemo in dettaglio nel paragrafo 2.1.3,

$$x_1 = \left(\frac{3 \cdot 0^2 + 2}{2 \cdot 2}\right)^2 - 2 \cdot 0 = \frac{2^2}{4^2} \equiv \frac{4}{2} \equiv 2$$

e

$$y_1 = \left(\frac{3 \cdot 0^2 + 2}{2 \cdot 2}\right)(0 - 2) - 2 \equiv \frac{2}{4}(-2) - 2 \equiv -1 - 2 = -3 = 4,$$

quindi $C_1 = (2, 4)$.

Adesso calcoliamo $kQ = 2(6, 6) = (x, y)$ per poi ricavare C_2. Osserviamo che

$$x = \left(\frac{3 \cdot 6^2 + 2}{2 \cdot 6}\right)^2 - 2 \cdot 6 \equiv \frac{5^2}{5^2} - 5 \equiv 1 - 5 \equiv -4 \equiv 3$$

e

$$y = \left(\frac{3 \cdot 6^2 + 2}{2 \cdot 6}\right)(6 - 3) - 6 \equiv 1 \cdot 3 - 6 \equiv 3 - 6 \equiv -3 \equiv 4,$$

quindi $kQ = (3, 4) \Rightarrow C_2 = (6, 1) + (3, 4) = (x_2, y_2)$, dove

$$x_2 = \left(\frac{4 - 1}{3 - 6}\right)^2 - 6 - 3 \equiv \frac{3^2}{(-3)^2} - 2 \equiv 1 - 2 \equiv -1 \equiv 6$$

e

$$y_2 = \left(\frac{4-1}{3-6}\right)(6-6) - 1 \equiv -1 \equiv 6.$$

Abbiamo ottenuto $C_2 = (6, 6)$.

Cifratura. Essendo $Q = 3P$, la chiave privata risulterà essere $d = 3$; quindi $M = C_2 - 3C_1 = C_2 + 3(-C_1)$. Osserviamo che $-C_1 = -(2, 4) = (2, -4) = (2, 3)$, quindi $3(-C_1) = 2(-C_1) + (-C_1)$.

Le coordinate di $2(-C_1) = 2(2, 3)$ sono rispettivamente:

$$x = \left(\frac{3 \cdot 2^2 + 2}{2 \cdot 3}\right)^2 - 2 \cdot 2 \equiv \frac{0^2}{6^2} - 4 \equiv -4 \equiv 3$$

e

$$y = \left(\frac{3 \cdot 2^2 + 2}{2 \cdot 3}\right)(2-3) - 3 \equiv -3 \equiv 4.$$

Allora $3(-C_1) = (2, 3) + (3, 4)$, cioé le sue coordinate saranno:

$$x = \left(\frac{4-3}{3-2}\right)^2 - 2 - 3 \equiv \frac{1^2}{1^2} - 5 \equiv 1 - 5 \equiv -4 \equiv 3$$

e

$$y = \frac{1}{1}(2-3) - 3 \equiv -1 - 3 \equiv -4 \equiv 3.$$

Possiamo concludere che $M = (6, 6) + (3, 3) = (x_M, y_M)$, dove

$$x_M = \left(\frac{3-6}{3-6}\right)^2 - 6 - 3 \equiv \frac{(-3)^2}{(-3)^2} - 2 \equiv 1 - 2 \equiv -1 \equiv 6$$

e

$$y_M = \left(\frac{-3}{-3}\right)(6-6) - 6 \equiv -6 \equiv 1.$$

Abbiamo ottenuto in questo modo $M = (6, 1)$ e quindi anche $m = 110$. □

La caratteristica dell'ECC è dunque quella dell'applicazione di un problema noto, il Logaritmo Discreto, in un particolare gruppo, quello dei punti di una curva ellittica. Questo comporterà notevoli vantaggi dal punto di vista della sicurezza.

Dopo questa breve introduzione all'ECC, il capitolo che segue fornisce una descrizione formale delle Curve Ellittiche, evidenziando le caratteristiche di interesse crittografico. Nel capitolo 3 viene analizzato l'ECDLP, in particolare vengono presentati i suoi metodi di risoluzione. Infine, il capitolo 4 sarà caratterizzato dal confronto tra la sicurezza dell'ECC e quella relativa ai problemi tradizionali dell'IFP e DLP.

Capitolo 2

CURVE ELLITTICHE

La Crittografia basata su Curve Ellittiche è una teoria molto recente. Tali curve sono state utilizzate per risolvere problemi di varia natura come ad esempio la fattorizzazione dei numeri interi.
In questo capitolo si darà una descrizione matematica delle Curve Ellittiche, verrà studiata la sua aritmetica (*legge di gruppo*, *multipli*) e verranno presentati alcuni concetti fondamentali come quello di *ordine di una curva*. Infine si evidenzieranno algoritmi che permetteranno di ricavare con più o meno semplicità tale ordine.

2.1 Introduzione

Consideriamo un campo generico K e la sua chiusura algebrica $\overline{K}$ e sia $K \leq F \leq \overline{K}$. Riassumiamo qui i concetti fondamentali di piano affine e proiettivo su K, adattando alcune definizioni più

generali alle nostre esigenze.

2.1.1 Coordinate proiettive

2.1.1 Definizione. Si definisce *piano affine* $\mathcal{A}^2(K)$ ogni insieme, i cui elementi diremo *punti*, che sia in corrispondenza biunivoca con lo spazio vettoriale K^2. Identificando $\mathcal{A}$ con K^2 per mezzo di tale corrispondenza biunivoca scriveremo direttamente:

$$\mathcal{A}^2(K) = \{(x, y) : x, y \in K\}.$$

Una coppia ordinata di punti (P, Q) è detta *vettore affine* ed è indicata con $\overrightarrow{PQ}$. Ad esso viene associato il vettore $\varphi(P, Q) = Q - P \in K^2$. □

La dimostrazione della seguente Proposizione è del tutto banale.

2.1.2 Proposizione. *In un piano affine valgono le seguenti proprietà:*

1) $\forall P \in \mathcal{A}, \forall\, \mathbf{v} \in K^2 \exists\, !Q \in \mathcal{A} : \varphi(P, Q) = \mathbf{v}$.

2) $\varphi(P, Q) + \varphi(Q, R) = \varphi(P, R)$ *(relazione triangolare)* □

Ad esempio $\mathbb{R}^2$ è *piano affine su* $\mathbb{R}$ e la coppia (x, y) è detta *punto affine reale*.

Definiamo adesso una relazione di equivalenza $\sim$ su $K^3 - \mathbf{0}$:

$$(X, Y, Z) \sim (X', Y', Z') \iff (X, Y, Z) = \lambda(X', Y', Z')$$

con $0 \neq \lambda \in K$.

La classe di equivalenza della terna (X, Y, Z) viene chiamata *punto proiettivo* e visto che dipende dai rapporti fra X, Y e Z si indica con $[X : Y : Z]$.

Ad esempio le terne $(4, 2, 6)$ e $(12, 6, 18)$ sono equivalenti (con $\lambda = 3$) e appartengono alla classe di equivalenza $[2 : 1 : 3]$.

2.1.3 Definizione. Si definisce *piano proiettivo* $\mathbf{P}^2(K)$ su K l'insieme, $(K^3 - \mathbf{0})/\sim$, delle classi di equivalenza rispetto $\sim$ ovvero l'insieme di tutti i punti proiettivi. Pertanto scriveremo

$$\mathbf{P}^2(K) = \{[X : Y : Z] : (X, Y, Z) \in (K^3 - \mathbf{0})\}.$$

Allo stesso modo si definisce la *retta proiettiva* $\mathbf{P}^1(K)$ l'insieme delle classi di equivalenza di coppie $(X, Y) \neq (0, 0)$ con $X, Y \in K$ rispetto alla relazione:

$$(X, Y) \sim (X', Y') \iff (X, Y) = \lambda(X', Y')$$

con $0 \neq \lambda \in K$. □

Consideriamo una classe di equivalenza generica di $\mathbf{P}^2(K)$.

Se $Z \neq 0$ il generico punto $[X : Y : Z]$ di $\mathbf{P}^2(K)$ è equivalente a $[X/Z : Y/Z : 1]$. Quindi se poniamo $X' = X/Z$ e $Y' = Y/Z$,

la classe equivalente può essere rappresentata univocamente da $[X' : Y' : 1]$. Le generiche terne di questo tipo sono in corrispondenza biunivoca con K^2 e dunque formano un piano affine $\mathcal{A}^2(K)$. Tali punti sono detti *punti finiti* di $\mathbf{P}^2(K)$. Per ottenere tutto il piano proiettivo, ai punti finiti vanno aggiunti quindi soltanto i punti con $Z = 0$.

Se $Z = 0$ le classi di equivalenza che si ottengono sono del tipo $[X : Y : 0]$ con X e Y non entrambi nulli: il loro insieme si identifica pertanto con la retta proiettiva $\mathbf{P}^1(K)$ e viene detto *retta all'infinito* e i suoi punti $[X : Y : 0]$ sono detti punti all'infinito. Possiamo concludere che $\mathbf{P}^2(K) = \mathcal{A}^2(K) \cup \mathbf{P}^1(K)$.

Quelle che sono state introdotte fino ad ora sono dette *coordinate proiettive standard*, mentre quelle utilizzate nell'ECC sono le *coordinate proiettive pesate* che sono definite nel modo seguente:
si definisce la relazione di equivalenza $\sim$ su $K^3 - \mathbf{0}$

$$(X, Y, Z) \sim (X', Y', Z') \iff X = \lambda^c X', \;\; Y = \lambda^d Y', \;\; Z = \lambda Z'$$

con $\lambda \in K$ e c, d due interi positivi.

Indichiamo con

$$\mathbf{P}^2[F] = \{\lambda[x, y; t] : F \ni \lambda \neq 0,\, F^3 \ni (x, y; t) \neq \mathbf{0}\}$$

il *piano proiettivo su F.*

2.1.4 Definizione Una *curva ellittica* $\mathcal{E}$, definita sul campo K,

è una curva proiettiva *liscia* per la quale esiste un riferimento proiettivo del piano $\mathbf{P}^2[K]$ rispetto al quale essa è definita dall'Equazione Generalizzata di Weierstrass:

$$y^2t + a_1xyt + a_3yt^2 = x^3 + a_2x^2t + a_4xt^2 + a_6t^3 \qquad (2.1)$$

dove $a_1, a_2, a_3, a_4, a_6 \in K$. □

Scelta $t = 0$ come retta all'infinito, i punti propri della curva sono i punti $[x, y; t]$ con $t \neq 0$ e, dividendo per t, si possono rappresentare con l'equazione non omogenea

$$F(x, y) = y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6 = 0.$$

La curva $\mathcal{E}$ possiede un solo punto all'infinito Θ, come si può vedere intersecando $\mathcal{E}$ con la retta all'infinito di equazione $t = 0$, perché si trova $x^3 = 0$ quindi $\Theta = [0, 1; 0]$.
Indichiamo con $\mathcal{E}(F)$ tutti i punti di $\mathbf{P}^2[F]$ che soddisfano l'equazione (2.1) e, in particolare consideriamo i seguenti insiemi:

$$\mathcal{E}(K) = \{(x, y) \in K \times K : F(x, y) = 0\} \cup \{\Theta\}$$

e

$$\mathcal{E}(\overline{K}) = \{(x, y) \in \overline{K} \times \overline{K} : F(x, y) = 0\} \cup \{\Theta\}.$$

Ricordiamo che con il termine *liscio* si vuole specificare che in $\mathcal{E}(K)$ non vi sono punti *singolari*, ossia punti su cui le derivate parziali di $F(x, y)$ si annullano simultaneamente. Per questo motivo, le curve *lisce* sono anche dette *non–singolari*.

Si definisce *discriminante* Δ della curva $\mathcal{E}$, definita dall'equazione (2.1), la quantità:

$$\Delta = -d_2^2 d_8 - 8d_4^3 - 27d_6^2 + 9d_2 d_4 d_6$$

dove

$$d_2 = a_1^2 + 4a_2, \qquad d_4 = 2a_4 + a_1 a_3,$$

$$d_6 = a_3^2 + 4a_6, \qquad d_8 = a_1^2 a_6 + 4a_2 a_6 - a_1 a_3 a_4 + a_2 a_3^2 - a_4^2.$$

Come vedremo il discriminante ci permette di stabilire se una curva è *non–singolare*, visto che, CNS affinché lo sia è che il discriminante sia diverso da zero.

Come abbiamo visto, ogni curva ellittica possiede un solo *punto all'infinito* Θ. Osserviamo che il punto all'infinito di $\mathcal{E}$ è $[0, 1; 0]$ cioè lo stesso dell'asse y, pertanto la retta passante per un punto P e per il *punto all'infinito* Θ è la retta parallela all'asse y e passante per P.

La curva $\mathcal{E}$ e l'asse y sono tangenti all'infinito, avendo il solo punto Θ come intersezione.

La seguente figura suggerisce come all'infinito le due estremità dell'asse y risultino tangenti alla curva nel punto all'infinito:

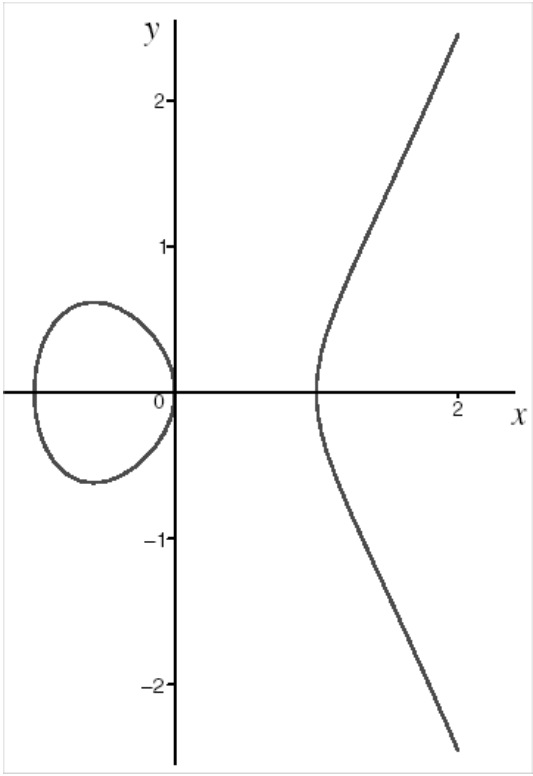

Figura 2.1: $\mathcal{E} : y^2 = x^3 - x$

2.1.2 Semplificazione delle equazioni di Weierstrass

In questa sezione vogliamo semplificare l'equazione di Weierstrass pur lasciandola della forma $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$.

Chiaramente non tutte le trasformazioni di variabili conservano tale forma, anzi si può dimostrare che se $\bar{x} = f(x, y)$ e $\bar{y} = g(x, y)$ è un cambiamento lineare di coordinate che conserva la forma normale di $\mathcal{E}$ allora $\bar{x} = u^2x + r$ e $\bar{y} = u^3y + u^2sx + t$ per opportuni $u,\ r,\ s,\ t$.

Diamo allora la seguente

2.1.5 Definizione. Due curve ellittiche $\mathcal{E}_1$ e $\mathcal{E}_2$ definite sul campo K dalle equazioni

$$\mathcal{E}_1 : \qquad y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

$$\mathcal{E}_2 : \qquad y^2 + \bar{a}_1xy + \bar{a}_3y = x^3 + \bar{a}_2x^2 + \bar{a}_4x + \bar{a}_6$$

sono *isomorfe* su K se esistono $u, r, s, t \in K$, con $u \neq 0$, tali che il cambiamento di variabili

$$(x, y) \longrightarrow (u^2x + r, u^3y + u^2sx + t)$$

trasforma l'equazione di $\mathcal{E}_1$ in quella di $\mathcal{E}_2$. Tale trasformazione è chiamata *trasformazione ammissibile di variabili.* □

Un'equazione di Weierstrass

$$\mathcal{E} : \; y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

definita su un campo K può essere semplificata, quindi, considerando un ammissibile cambiamento di variabili.
Considereremo separatamente i casi in cui K ha caratteristica diversa da 2 e 3, o caratteristica uguale a 2 o a 3.

$Char(K) \neq 2, 3$
In tale caso rientrano sia il campo dei numeri reali che i campi finiti $GF(p)$ con $p > 3$.
In questo caso possiamo considerare la trasformazione

$$(x, y) \longrightarrow \left(\frac{x - a_1^2 - 12a_2}{36}, \frac{y - 3a_1x}{216} - \frac{a_1^3 + 4a_1a_2 - 12a_3}{24} \right)$$

che trasforma l'equazione (2.1) in

$$y^2 = x^3 + ax + b$$

dove (v. [8]) $a = -\frac{c_4}{48}$ e $b = -\frac{c_6}{864}$ considerando rispettivamente $b_2 = a_1^2 + 4a_2$, $b_4 = a_1a_3 + 2a_4$, $b_6 = a_3^2 + 4a_6$, $b_8 = a_1^2a_6 - a_1a_3a_4 + 4a_2a_6 + a_2a_3^2 - a_4^2$ e $c_4 = b_2^2 - 24b_4$, $c_6 = -b_2^3 + 36b_2b_4 - 216b_6$; inoltre $\Delta = -4a^3 - 27b^2$.

Se si considera l'equazione implicita $F(x, y) = y^2 - f(x)$, dove $f(x) = x^3 + ax + b$, la condizione di *non–singolarità* equivale al fatto che non esiste alcun punto di $\mathcal{E}$ in cui le due derivate parziali

$$\frac{\partial F}{\partial x} = -f'(x), \quad \frac{\partial F}{\partial y} = 2y$$

si annullino contemporaneamente.

In ogni tale punto la retta tangente si calcola come segue: consideriamo l'equazione $y^2 = f(x)$, e calcoliamo il coefficiente angolare $\frac{dy}{dx}$ della retta tangente ad $\mathcal{E}$ nel punto (x, y). Derivando si ottiene $2y\frac{dy}{dx} = f'(x)$ e quindi

$$\frac{dy}{dx} = \frac{f'(x)}{2y} = \frac{-\partial F/\partial x}{\partial F/\partial y}.$$

L'ultima equazione ha senso se il denominatore è diverso da 0 mentre può essere interpretata come la pendenza di una retta verticale se il numeratore è diverso da 0 e il denominatore è nullo. Se entrambe le quantità si annullano, la pendenza della retta tangente in quel punto perde significato, ovvero in quel punto la curva non ammette tangente. Se entrambe le derivate parziali si annullano nel punto $P = (x_0, y_0)$ significa che $f'(x_0) = 2y_0 = 0$. Inoltre,

considerato che $y^2 = f(x)$, si ha che $f(x_0) = 0$.

$\grave{E}$ ben noto che un polinomio $f(x)$ ammette una radice multipla α se e solo se α è radice di $f(x)$ e di $f'(x)$. Quindi l'annullamento delle derivate parziali in $P = (x_0, y_0)$ implica che x_0 sia una radice multipla di $f(x)$.

Dalla teoria dei risultanti è noto che due polinomi hanno un fattore comune di grado positivo se e solo se il loro risultante è zero. Poiché il risultante di due polinomi $f(x) = a_0 + \cdots + a_m x^m$ e $g(x) = b_0 + \cdots + b_n x^n$ è per definizione il determinante

$$R(f,g) = \begin{vmatrix} a_0 & a_1 & \cdots & a_m & 0 & 0 & 0 & \cdots & 0 \\ 0 & a_0 & a_1 & \cdots & a_m & 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \cdots & \cdots & a_0 & a_1 & \cdots & a_m \\ b_0 & b_1 & \cdots & b_n & 0 & 0 & 0 & \cdots & 0 \\ 0 & b_0 & b_1 & \cdots & b_n & 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \cdots & \cdots & b_0 & b_1 & \cdots & b_n \end{vmatrix}$$

abbiamo

$$R(f,f') = \begin{vmatrix} b & a & 0 & 1 & 0 \\ 0 & b & a & 0 & 1 \\ a & 0 & 3 & 0 & 0 \\ 0 & a & 0 & 3 & 0 \\ 0 & 0 & a & 0 & 3 \end{vmatrix} = 4a^3 + 27b^2 = \Delta$$

è nullo se e solo se esiste una radice multipla x_0 di $f(x)$.

Riassumiamo quanto detto nella seguente

2.1.6 Proposizione. A meno di un opportuno cambiamento di variabili, una curva ellittica $\mathcal{E}$ sul campo K, avente caratteristica diversa da 2 e 3, è definita dall'Equazione di Weierstrass

$$y^2 = x^3 + ax + b \tag{2.2}$$

dove $a, b \in K$ sono tali che $4a^3 + 27b^2 \neq 0$. □

Facciamo un esempio considerando come campo K l'insieme dei numeri reali.
La seguente figura rappresenta la curva ellittica $y^2 = x^3 - 3x + 3$. Come si può intuire dal grafico, tale curva ammette in tutti i punti una retta tangente. Infatti possiamo facilmente osservare che risulta $\Delta = 108 - 243 \neq 0$.

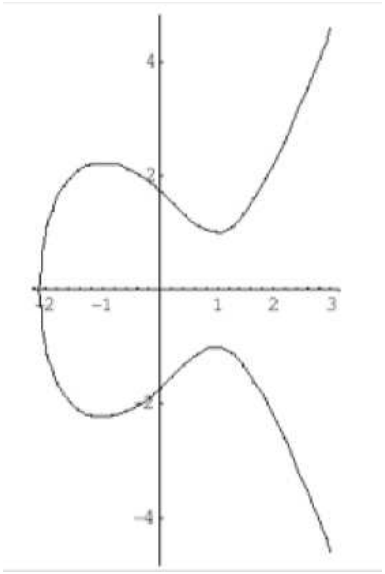

Figura 2.2: Curva ellittica: $y^2 = x^3 - 3x + 3$

La figura 2.3 rappresenta la curva $y^2 = x^3 - 3x + 2 = (x-1)^2(x+2)$ la quale presenta una singolarità, un *nodo*, nel punto $(1, 0)$, come si può ossevare anche dal fatto che $\Delta = 108 - 108 = 0$.

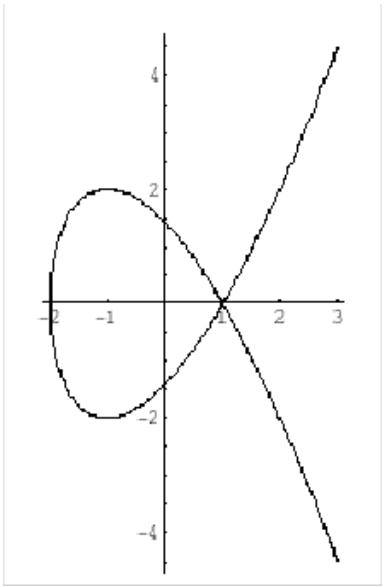

Figura 2.3: Curva: $y^2 = x^3 - 3x + 2 = (x-1)^2(x+2)$

Nella figura 2.4, invece, è rappresentata la curva $y^2 = x^3$ che nel punto $(0, 0)$ presenta un punto singolare di tipo *cuspide*. Infatti $\Delta = 0 - 0$.

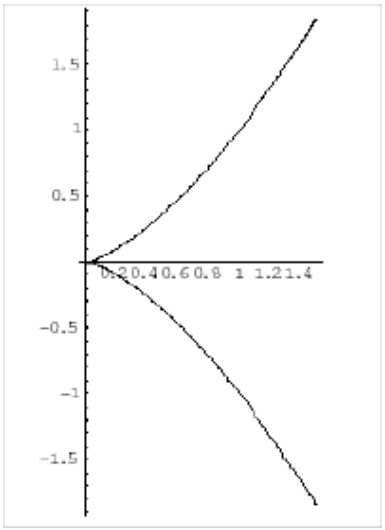

Figura 2.4: Curva: $y^2 = x^3$

$Char(K) = 2$

Se la caratteristica del campo è uguale a 2 allora ci sono due casi da considerare nell'equazione di Weierstrass $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$.

$a_1 \neq 0$

In questo caso si considera la trasformazione

$$(x, y) \longrightarrow \left(a_1^2 x + \frac{a_3}{a_1},\ \ a_1^3 y + \frac{a_1^2 a_4 + a_3^2}{a_1^3} \right)$$

che trasforma la (2.1) in:

$$y^2 + xy = x^3 + ax^2 + b \tag{2.3}$$

dove $a, b \in K$. Tale curva viene detta *non–supersingolare*[1] e ha discriminante $\Delta = b$.

$a_1 = 0$

La trasformazione ammissibile è

$$(x, y) \longrightarrow (x + a_2, y)$$

che trasforma la curva $\mathcal{E}$ in

$$y^2 + cy = x^3 + ax + b$$

dove $a, b, c \in K$.

Tale curva è detta *supersingolare* e ha discriminante $\Delta = c^4$.

$Char(K) = 3$

Se la caratteristica del campo è 3, dobbiamo distinguere due casi.

$a_1^2 \neq -a_2$

La trasformazione ammissibile è

$$(x, y) \longrightarrow \left(x + \frac{d_4}{d_2}, y + a_1 x + a_1 \frac{d_4}{d_2} + a_3 \right),$$

dove $d_2 = a_1^2 + a_2$ e $d_4 = a_4 - a_1 a_3$, che trasforma la curva $\mathcal{E}$ nella curva

$$y^2 = x^3 + ax^2 + b$$

[1]Sia n la caratteristica del campo K. Una curva ellittica $\mathcal{E}$ definita su K è *supersingolare* se n divide t, dove t è la *traccia* . Se n non divide t, $\mathcal{E}$ è detta *non–supersingolare*. (Per il concetto di traccia vedere par. 2.3)

dove $a, b \in K$. Tale curva è detta *non–supersingolare* e ha discriminante $\Delta = -a^3 b$.

$a_1^2 = -a_2$

La trasformazione ammissibile

$$(x, y) \longrightarrow (x, y + a_1 x + a_3)$$

trasforma la curva $\mathcal{E}$ in

$$y^2 = x^3 + ax + b$$

dove $a, b \in K$. Questa curva è detta *supersingolare* ed ha discriminante $\Delta = -a^3$.

2.1.3 La legge di Gruppo

Una caratteristica dell'insieme dei punti di una curva ellittica è che si può definire una struttura algebrica su di esso.
Tale struttura è data da un'operazione di *addizione*, rispetto alla quale essa è un gruppo abeliano. Proprio questo gruppo viene utilizzato per costruire i crittosistemi basati su curve ellittiche.
La regola d'addizione è detta *regola tangente-corda* e, nel caso reale, può essere spiegata intuitivamente in modo geometrico. Si considerano due punti distinti della curva $\mathcal{E}$, $P = (x_1, y_1)$ e $Q = (x_2, y_2)$. La somma $R = P + Q$ si ottiene nel seguente modo: si traccia la retta passante per P e Q. Tale retta interseca

la curva in un terzo punto R'. Basta individuare il punto simmetrico a R' rispetto all'asse delle ascisse (ovvero cambiare il segno della coordinata y) per ottenere R, come mostra la figura 2.5.

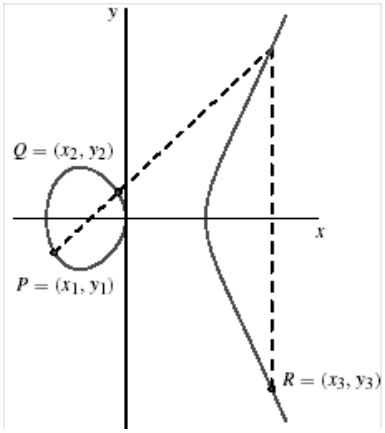

Figura 2.5: Addizione: $P + Q = R$

La somma $P + P$, invece, segue il seguente procedimento: si traccia la retta tangente alla curva nel punto P. Tale retta interseca la curva in un secondo punto R'. Si determina il simmetrico di R' rispetto all'asse delle x e si ottiene così $R = P + P$.

2.1.7 Nota. Le *involuzioni*, cioè i punti P tali che $P + P = \Theta$, coincidono con le intersezioni di $\mathcal{E}$ con l'asse x. □

La figura 2.6 rappresenta proprio questo procedimento.

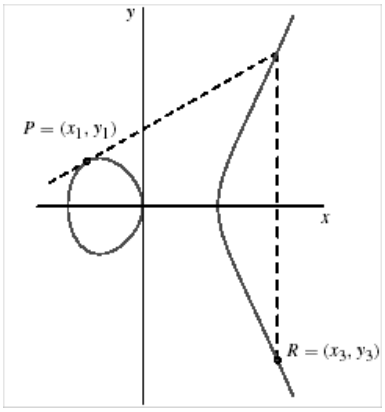

Figura 2.6: Addizione: $P + P = R$

Le formule algebriche per la *Legge di Gruppo* possono essere derivate dalla rappresentazione geometrica appena mostrata. Consideriamo due casi:

Legge di gruppo per curve su campi di caratteristica diversa da 2 e 3

Consideriamo l'equazione $y^2 = x^3 + ax + b$. Distinguamo tre casi:

- Consideriamo prima la somma di due punti distinti P e Q rispettivamente di coordinate cartesiane (x_1, y_1) e (x_2, y_2). Analizziamo prima il caso in cui $x_1 \neq x_2$. Sia L la retta passante per questi due punti di coefficiente angolare

$$m = \frac{y_2 - y_1}{x_2 - x_1}.$$

  In questo caso l'equazione della retta L è $y = m(x - x_1) + y_1$. Per determinare le due intersezioni, quindi, bisogna risolvere l'equazione cubica $(m(x - x_1) + y_1)^2 = x^3 + ax + b$ che può

essere scritta come

$$0 = x^3 - m^2x^2 + (2m^2x_1^2 - 2m^2y_1 + a)x + c. \qquad (2.4)$$

Di tale cubica però conosciamo già le due radici x_1 e x_2, quindi se fattorizziamo il polinomio (2.4) si ottiene

$$x^3 - m^2x^2 + (2m^2x_1 - 2m^2y_1 + a)x + c = (x - x_1)(x - x_2)(x - t) =$$

$$= x^3 - (x_1 + x_2 + t)x^2 + ..$$

dove t è la terza radice da calcolare. Si ha che $t = m^2 - x_1 - x_2$ e quindi, considerando la figura 2.5, R' ha coordinate

$$x = m^2 - x_1 - x_2$$

$$y = m(x - x_1) + y_1$$

Infine se si riflette rispetto all'asse delle ascisse si ottiene il punto $R = (x_3, y_3) = P + Q$, dove

$$x_3 = m^2 - x_1 - x_2$$

$$y_3 = m(x_1 - x_3) - y_1$$

Se invece $x_1 = x_2$ (e quindi $y_1 = -y_2$), la retta L è verticale e quindi interseca la curva nel punto all'infinito Θ. Pertanto si ottiene $P + Q = \Theta$, ossia $Q = -P$.

- Vediamo adesso il caso in cui $P = Q = (x_1, y_1)$. In questo caso la retta L è tangente alla curva nel punto P. La pendenza m di tale retta si ottiene con la derivazione implicita:

$$\frac{dy}{dx} = \frac{\partial F/\partial x}{\partial F/\partial y}$$

$$2y\frac{dy}{dx} = 3x^2 + a, \text{ quindi } \; m = \left(\frac{dy}{dx}\right)_{(x_1,y_1)} = \frac{3x_1^2 + a}{2y_1}.$$

Se $y_1 = 0$, la retta L è verticale e si definisce $P + P = \Theta$.

Se, invece $y_1 \neq 0$, l'equazione della retta L è $y = m(x - x_1) + y_1$. Procedendo come nel caso precedente otteniamo un'equazione cubica di cui conosciamo la sola radice x_1 che è doppia. Ripetendo i calcoli precedenti si ottiene il punto $R = (x_3, y_3) = P + P$, dove

$$x_3 = m^2 - 2x_1$$

$$y_3 = m(x_1 - x_3) - y_1$$

- Infine consideriamo il caso in cui $Q = \Theta$.

  La retta passante per P e Θ è verticale ed interseca la curva nel punto P', che è proprio il simmetrico di P rispetto all'asse x. Pertanto quando si riflette il punto P' rispetto all'asse delle ascisse, per ottenere il punto somma $P + \Theta$, si ricava nuovamente P. Quindi per ogni P appartenente alla curva si ha

  $$P + \Theta = P.$$

  Concludendo, valgono le seguenti proprietà:

  1. **Identità**: $P + \Theta = P$, per ogni P appartenente alla curva;
  2. **Opposto**: Dato il punto $P = (x, y)$, allora $(x, y) + (x, -y) = \Theta$. Il punto $(x, -y)$ viene indicato con $-P$

ed è chiamato *opposto* di P; inoltre, se P appartiene alla curva anche P' è un punto di essa;

3. **Addizione**: Siano $P = (x_1, y_1)$ e $Q = (x_2, y_2)$ punti appartenenti alla curva, con $P \neq \pm Q$. Allora $R = P + Q = (x_3, y_3)$ dove

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \text{ e } \; y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1;$$

4. **Raddoppio**: Sia $P = (x_1, y_1)$ appartenente alla curva e $P \neq -P$. Allora $R = P + P = (x_3, y_3)$, dove

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1 \quad \text{e} \quad y_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1.$$

Legge di gruppo per curve non-supersingolari su campi di caratteristica 2 o 3

Consideriamo la curva non-supersingolare: $y^2 + xy = x^3 + ax^2 + b$

Valgono le seguenti proprietà:

1. **Identità**: $P + \Theta = P$, per ogni P appartenente alla curva;

2. **Opposto**: Dato il punto $P = (x, y) \in \mathcal{E}$, il punto $P' = (x, x+y)$ appartiene anch'esso a $\mathcal{E}$. Inoltre $(x, y)+(x, x+y) = \Theta$. Pertanto il punto $(x, x + y)$ viene indicato con $-P$ ed è chiamato *opposto* di P;

3. **Addizione**: Siano $P = (x_1, y_1)$ e $Q = (x_2, y_2)$ punti appartenenti alla curva, con $P \neq \pm Q$. Allora $R = P + Q = (x_3, y_3)$

dove

$$x_3 = \left(\lambda^2 + \lambda + x_1 + x_2 + a\right) \text{ e } y_3 = \lambda(x_1 + x_3) + x_3 + y_1$$

con $\lambda = (y_1 + y_2)/(x_1 + x_2)$;

4. **Raddoppio**: Sia $P = (x_1, y_1)$ appartenente alla curva e $P \neq -P$. Allora $R = P + P = (x_3, y_3)$, dove

$$x_3 = \lambda^2 + \lambda + a = x_1^2 + \frac{b}{x_1^2} \quad \text{e} \quad y_3 = (x_1^2 + \lambda x_3 + x_3)$$

con $\lambda = x_1 + y_1/x_1$.

Legge di gruppo per curve supersingolari

Considerando la curva supersingolare: $y^2 + cy = x^3 + ax + b$ abbiamo invece:

1. **Identità**: $P + \Theta = P$, per ogni P appartenente alla curva;

2. **Opposto**: Dato il punto $P = (x, y) \in \mathcal{E}$, il punto $P' = (x, x+c)$ è anch'esso un punto di $\mathcal{E}$ e risulta $(x, y)+(x, x+c) = \Theta$. Pertanto P' viene indicato con $-P$ ed è chiamato *opposto* di P;

3. **Addizione**: Siano $P = (x_1, y_1)$ e $Q = (x_2, y_2)$ punti appartenenti alla curva, con $P \neq \pm Q$. Allora $R = P + Q = (x_3, y_3)$ dove

$$x_3 = \left(\frac{y_2 + y_1}{x_2 + x_1}\right)^2 + x_1 + x_2 \text{ e } y_3 = \left(\frac{y_2 + y_1}{x_2 + x_1}\right)(x_1 + x_3) + y_1 + c;$$

4. **Raddoppio**: Sia $P = (x_1, y_1)$ appartenente alla curva e $P \neq -P$. Allora $R = P + P = (x_3, y_3)$, dove

$$x_3 = \left(\frac{x_1^2 + a}{c}\right)^2 \quad \text{e} \quad y_3 = \left(\frac{x_1^2 + a}{c}\right)(x_1 + x_3) + y_1 + c.$$

Ottenute queste espressioni possiamo enunciare il seguente:

2.1.8 Teorema. L'addizione di punti di una curva ellittica $\mathcal{E}$ definisce su di essa la struttura di gruppo abeliano. □

2.2 Multipli

Questo paragrafo si occuperà dei metodi per calcolare kP, dove k è un intero e P è un punto sulla curva ellittica $\mathcal{E}$ definita su un campo $GF(p^t)$.

Se k è un intero positivo, si indica con kP la somma

$$Q = kP = \underbrace{P + P + ... + P}_{k \text{ volte}}.$$

Se invece k è negativo, kP rappresenta la somma $-k(-P)$.

Se $k = 0$ si pone $0P = \Theta$.

Si osservi che per calcolare il multiplo kP è sufficiente applicare la formula del raddoppio, ricordando che se $\sum_{i=0}^{\lceil log_2 k \rceil} \epsilon_i 2^i$ è l'espansione in base 2 di k (e quindi $\epsilon_i = 0, 1$) allora $kP = (\sum \epsilon_i 2^i)P = \sum \epsilon_i (2^i P)$.

Esistono inoltre altri algoritmi per la determinazione dei multipli

che sfruttano le proprietà delle curve ellittiche per i quali rimandiamo a [16].
I protocolli dell'ECC si basano proprio sulla determinazione dei multipli e sulla difficoltà di calcolare k anche se sono noti P e kP. Questo è, infatti, il problema del Logaritmo Discreto su Curva Ellittica (ECDLP) che verrà presentato nel capitolo successivo.

2.3 Ordine di una curva ellittica

Un parametro molto importante e difficile da determinare è il numero di punti razionali su un dato campo di una generica curva ellittica definita su un campo finito $GF(p^t)$. Tale grandezza si indica con $\#\mathcal{E}(GF(p^t))$ ed è detta *ordine della curva* $\mathcal{E}$ *sul campo finito* $GF(p^t)$.

2.3.1 Esempio

Consideriamo ad esempio la curva $\mathcal{E}$ sul campo finito $GF(5)$ descritta dall'equazione $y^2 = x^3 + x + 1$. Il numero di punti si ottiene in questo modo: si elencano tutti i possibili valori che può assumere x, si calcolano i valori $x^3 + x + 1 \pmod 5$ e successivamente si determina, se esiste, la radice quadrata.

x	$x^3 + x + 1$ (mod5)	y	Punti
0	1	± 1	$(0,1), (0,4)$
1	3		
2	1	± 1	$(2,1), (2,4)$
3	1	± 1	$(3,1), (3,4)$
4	4	± 2	$(4,2), (4,3)$

Considerando anche Θ, si ottiene che $\#\mathcal{E}(GF(5)) = 9$. □

Naturalmente, per valori grandi di q, questo metodo non è conveniente. È chiaro che, se $q = p^t$, si può affermare che vi sono al più $2q + 1$ punti. Infatti, si possono assegnare ad x tutti i valori del campo $GF(q)$ e per ognuno di essi vi sono al più due y corrispondenti. Se in più si aggiunge il punto all'infinito Θ, si ottiene proprio $2q + 1$.
Il seguente Teorema, comunque, fornisce dei limiti più precisi per $\#\mathcal{E}$.

2.3.2 Teorema (Hasse).

Sia $\mathcal{E}$ una curva ellittica definita sul campo finito $GF(q)$. Allora l'ordine di $\mathcal{E}$, $\#\mathcal{E}(GF(q))$, soddisfa:

$$q + 1 - 2\sqrt{q} \leq \#\mathcal{E}(GF(q)) \leq q + 1 + 2\sqrt{q}.$$

□

L'intervallo $[q + 1 - 2\sqrt{q}, q + 1 + 2\sqrt{q}]$ è chiamato *intervallo di*

Hasse. Una formulazione alternativa del teorema di Hasse è la seguente: se $\mathcal{E}$ è definita su $GF(q)$, allora

$$\#\mathcal{E}(GF(q)) = q + 1 - t$$

dove $|t| \leq 2\sqrt{q}$, dove t è detta *traccia* di $\mathcal{E}$ su $GF(q)$.
Anche se questo teorema ci permette di restringere l'intervallo in cui trovare l'esatto valore di $\#\mathcal{E}(GF(q))$, i crittosistemi basati su Curva Ellittica richiedono la conoscenza esatta di tale parametro. Esistono vari teoremi che permettono di determinare $\#\mathcal{E}$.

2.3.1 Metodo ingenuo

Consideriamo una curva ellittica $y^2 = x^3 + ax + b$ definita su $GF(p)$, con $p > 3$; per x che varia tra tutti i valori possibili in $GF(p)$ l'equazione $y^2 = x^3 + ax + b$ avrà rispettivamente:

- 2 soluzioni se $x^3 + ax + b$ è un residuo quadratico[2] non nullo in $GF(p)$;
- 1 soluzione se $x^3 + ax + b$ è divisibile per p;
- 0 soluzioni se $x^3 + ax + b$ non è un residuo quadratico in $GF(p)$.

Ha un buon funzionamento per campi piccoli mentre è inutilizzabile per grandi valori di p.

[2]Siano dati il primo p e $x \in \mathbb{Z}_p$. Allora x è un *residuo quadratico modulo p* se esiste un $y \in \mathbb{Z}_p$ tale che $y^2 \equiv x(mod\ p)$.

2.3.2 Algoritmo di Schoof

Nel 1985 Schoof presentò il primo algoritmo *polynomial-time* per calcolare $\#\mathcal{E}(GF(p))$ per un'arbitraria curva ellittica $\mathcal{E}$. L'algoritmo calcola $\#\mathcal{E}(GF(p))$ mod l, per alcuni valori primi di l, e utilizza successivamente il Teorema Cinese del Resto[3].
Essendo inefficiente per valori di q di interesse pratico, venne successivamente ottimizzato da Atkin e Elkies, ottenendo il cosiddetto *algoritmo Schoof-Elkies-Atkin* (SEA).
Tale algoritmo è basato sulla seguente procedura:
sia $\mathcal{E}$ la curva ellittica data da $y^2 = x^3 + ax + b$. Dal teorema di Hasse si ha che:

$$\#\mathcal{E}(GF(q)) = q + 1 - t, \text{ con } |t| \leq 2\sqrt{q}.$$

Sia $S = \{3, 5, \cdots\}$ un insieme di numeri primi diversi da 2 e da p tali che

$$\prod_{s \in S} s \geq 4\sqrt{q}$$

Si calcola t $(mod\ s)$ per ogni $s \in S$, quindi dal Teorema Cinese del Resto si determina t $(mod \prod s)$. Poiché $|t| \leq 2\sqrt{q}$ questo ci

[3]Siano $n_1, n_2, \cdots, n_k$ interi coprimi tra loro tali che $N = n_1 \times n_2 \cdots \times n_k$. Siano inoltre $a_1, a_2, \cdots, a_k$ interi qualunque. Allora il seguente sistema di congruenze:

$$\begin{cases} x \equiv a_1 (mod\ n_1) \\ x \equiv a_2 (mod\ n_2) \\ \cdots \\ x \equiv a_k (mod\ n_k) \end{cases}$$

ammette un'unica soluzione modulo N pari a $x_0 = \sum_{i=1}^{k} a_i M_i y_i$ dove $\forall i = 1, 2, \cdots k$, $M_i = \frac{N}{n_i}$ e y_i è tale che $M_i y_i \equiv 1 (mod\ n_i)$.

consente di determinare univocamente t.

Qui ci limitiamo a dare un esempio di come sia possibile utilizzare la formula di Hasse

$$\#\mathcal{E} = q + 1 - t \quad \text{con} \quad |t| \leq 2\sqrt{q}$$

e la conoscenza di alcuni sottogruppi di torsione di $\mathcal{E}$ per determinare $\#\mathcal{E}$.

Sullo stesso principio si basa l'Algoritmo di Schoof [20] che permette di calcolare $\#\mathcal{E}$ a partire dalle equazioni $\psi_n = 0$ di Lang [14] che determinano i punti di $\#\mathcal{E}$ di ordine n.

2.3.3 Esempio.

Consideriamo la curva $\mathcal{E}$ sul campo finito $GF(25)$ descritta dall'equazione $y^2 = x^3 + x + 1$.

Per il teorema di Hasse:

$$\#\mathcal{E} = q + 1 - t = 25 + 1 - t \quad \text{con} \quad |t| \leq 2\sqrt{q} = 2 \cdot \sqrt{25} = 10.$$

Abbiamo così ottenuto l'intervallo di Hasse: $16 \leq 26 - t \leq 36$.

Consideriamo il punto $(0, 1) \in \mathcal{E}(GF(25))$. Poiché $(0, 1) \in \mathcal{E}(GF(5))$ avremo che $\langle (0, 1) \rangle \subseteq \mathcal{E}(GF(5))$ e, come abbiamo visto nell'esempio 2.3.1, l'ordine di $(0, 1)$ in $\mathcal{E}(GF(5))$ è 9.

Tale risultato ci permette di restringere ulteriormente l'intervallo di possibili valori per $\#\mathcal{E}(GF(25))$, infatti, l'ordine di tale curva dovrà essere un multiplo di 9. Se fattorizziamo in numeri primi i

valori appartenenti all'intervallo di Hasse $I = [16, 36]$, otteniamo che gli unici valori possibili sono tre:

$$18 = 2 \cdot 3^2$$
$$27 = 3^3$$
$$36 = 2^2 \cdot 3^2.$$

Per avere ordine 18 o 36, la curva $\mathcal{E}(GF(25))$ deve possedere un'involuzione (e dunque un punto $P = (x, 0)$ sull'asse delle ascisse, cfr. Nota 2.1.7), quindi, si deve verificare che l'equazione $x^3 + x + 1 = 0$ ha radici in $GF(25)$. Ma ciò è falso, infatti:

1. Non ha radici in $GF(5)$, come mostra un calcolo diretto;

2. Poiché $[GF(25) : GF(5)] = 2$, se esistesse $\beta \in GF(25)$ tale che $\beta^3 + \beta + 1 = 0$, allora sarebbe $x^3 + x + 1 = g(x)(x - a)$, dove $g(x)$ è un polinomio di secondo grado ed $a \in GF(5)$ sarebbe un'ulteriore radice del polinomio $x^3 + x + 1 = 0$. Ma ciò è assurdo per quanto visto prima. Quindi la curva $\mathcal{E}(GF(5))$ non possiede punti di ordinata nulla.

Possiamo quindi concludere che $\mathcal{E}(GF(25))$ non possiede elementi di ordine 2 e quindi, per il teorema di Hasse, il suo ordine è 27. Per verificare l'esattezza del risultato mediante il metodo ingenuo e per confrontarne la praticità si veda l'Appendice B. □

Tra i vari lavori apparsi dal 1999, vogliamo menzionare l'articolo di Satoh, Skjrnaa e Taguchi [19] in cui gli autori hanno proposto

delle modifiche al metodo di Schoof per calcolare il numero di punti su campi finiti con caratteristica piccola. Tali metodi risultano essere molto veloci nel caso di campi di caratteristica 2 e permettono di determinare curve ellittiche di interesse crittografico definite su $GF(2^{163})$.

Capitolo 3

LOGARITMO DISCRETO SU CURVE ELLITTICHE

3.1 Introduzione

Come abbiamo già accennato, la sicurezza dell'ECC si basa sulla difficoltà di risolvere il problema del Logaritmo Discreto su curva ellittica (ECDLP).

3.1.1 Definizione. Il *problema del Logaritmo Discreto su curva ellittica* (ECDLP) è: data una curva ellittica $\mathcal{E}$ definita su un campo finito $GF(q)$, un punto $P \in \mathcal{E}(GF(q))$ di ordine n[1] e un punto $Q \in \langle P \rangle$, trovare l'intero $k \in \{0, \ldots, n-1\}$ tale che $Q = kP$. L'intero k è chiamato *logaritmo discreto di Q in base P* e si indica con $k = \log_P Q$. □

[1]Data una curva ellittica $\mathcal{E}/K$ e il punto $P \in \mathcal{E}(K)$, si definisce *ordine di P* il più piccolo intero positivo n, se esiste, tale che $nP = \Theta$. Se tale numero non esiste si dice che il punto P ha *ordine infinito*.

I parametri delle curve ellittiche utilizzate per i crittosistemi, quindi, devono essere scelti accuratamente in modo da resistere ai noti attacchi al ECDLP.

L'attacco più semplice consiste nella *ricerca esaustiva*, con la quale viene calcolata la sequenza $P, 2P, 3P, \cdots$ fino a che non si ottiene Q. Si può facilmente osservare che per contrastare tale attacco basta considerare n molto grande.

Per risolvere l'ECDLP vi sono due tipi di algoritmi: gli *special–purpose* e i *general–purpose*. Nel paragrafo successivo vengono presentati i principali attacchi che caratterizzano l'ECDLP.

3.2 Metodo Pohling–Hellman

L'algoritmo *Pohling–Hellman* riduce il calcolo di $k = log_P Q$ al calcolo del logaritmo discreto nei sottogruppi di $\langle P \rangle$ di ordine pari ai fattori primi di n, dove n è l'ordine di P.

Sia quindi $n = p_1^{e_1} p_2^{e_2} \dots p_r^{e_r}$ la fattorizzazione prima di n. La strategia seguita dall'algoritmo è quella di calcolare $k_i = k \ (mod \ p_i^{e_i})$ per ogni $1 \leq i \leq r$ e risolvere il sistema di congruenze

$$k \equiv k_i (mod \ p_i^{e_i}).$$

Il Teorema Cinese del Resto garantisce l'unicità della soluzione $k \ \ mod \ n$.

Vediamo ora come il calcolo di ogni k_i si riduce al calcolo di e_i

logaritmi discreti nel sottogruppo di $\langle P \rangle$ di ordine p_i.

Per semplificare la notazione utilizzeremo e e p per indicare i generici e_i e p_i. Scriviamo k in base p:

$k = z_0 + z_1 p + z_2 p^2 + \cdots + z_{e-1} p^{e-1} + z_e p^e + \cdots$ che può essere scritto come

$$k = z_0 + z_1 p + z_2 p^2 + \cdots + z_{e-1} p^{e-1} + p^e (z_e + \cdots)$$

dove $0 \leq z_i \leq p-1$. Quindi $k \pmod{p^e} = z_0 + z_1 p + z_2 p^2 + \cdots + z_{e-1} p^{e-1} \pmod{p^e}$.

Vediamo ora come calcolare i coefficienti $z_0 \cdots z_{e-1}$.

Definiamo $P_0 = \frac{n}{p} P$ e $Q_0 = \frac{n}{p} Q$. L'ordine di P_0, quindi, è p.

Esplicitiamo $Q = kP$ ricordando che l'ordine di P è n ($nP = \Theta$).

Otteniamo:

$$Q_0 = \frac{n}{p} Q = \frac{n}{p}[(z_0 + z_1 p + \cdots) P] = \frac{n}{p} z_0 P + (z_1 + \cdots) nP = z_0 P_0.$$

Pertanto z_0 si ottiene risolvendo un ECDLP in $\langle P_0 \rangle$.

Successivamente si calcola $Q_1 = \frac{n}{p^2}(Q - z_0 P)$.

Esplicitando ancora Q, si ha:

$$Q_1 = \frac{n}{p^2}(kP - z_0 P) = (k - z_0)(\frac{n}{p^2} P) = [z_1 + p(z_2 + \cdots)](\frac{n}{p} P) =$$

$$= z_1 \frac{n}{p} P + (z_2 + \cdots) nP = z_1 P_0.$$

Anche z_1, quindi, si ottiene risolvendo un ECDLP in $\langle P_0 \rangle$.

Il procedimento continua in questo modo per i rimanenti coefficienti, pertanto il generico z_t si ottiene calcolando $Q_t = \frac{n}{p^{t+1}}(Q - z_0 P - z_1 pP - \cdots - z_{t-1} p^{t-1} P)$ e risolvendo $Q_t = z_t P_0$.

Si intuisce che per resistere all'attacco *Pohling–Hellman* i sottogruppi generati dai fattori primi di n devono essere sufficientemente grandi e quindi l'ordine n di P deve essere divisibile per un grande primo.

3.2.1 Esempio.

Consideriamo la curva ellittica $\mathcal{E}$ definita su $GF(7919)$ dall'equazione:

$$\mathcal{E} : y^2 = x^3 + 1001x + 75.$$

Scelto $P = (4023, 6036) \in \mathcal{E}(GF_{7919})$, risulta che l'ordine di P è:

$$n = 7889 = 7^3 \cdot 23.$$

Consideriamo $Q = (4135, 3169) \in \langle P \rangle$. Vogliamo determinare $k = \log_P Q$.

- Determiniamo $k_1 = k \ \ mod\ 7^3$. Scriviamo $k_1 = z_0 + z_1 7 + z_2 7^2$ e calcoliamo

  $$P_0 = 7^2 \cdot 23P = (7801, 2071)$$

  $$Q_0 = 7^2 \cdot 23Q = (7801, 2071)$$

  e troviamo che $Q_0 = P_0$; quindi $z_0 = 1$. Adesso calcoliamo

  $$Q_1 = 7 \cdot 23(Q - P) = (7285, 14)$$

  e troviamo che $Q_1 = 3P_0$; quindi $z_1 = 3$.

  Infine, calcoliamo

  $$Q_2 = 23(Q - P - 3 \cdot 7P) = (7285, 7905)$$

e troviamo che $Q_2 = 4P_0$, quindi $z_2 = 4$. Allora $k_1 = 1 + 3 \cdot 7 + 4 \cdot 7^2 = 218$.

- Adesso determiniamo $k_2 = k \ \ mod\ 23$. Calcoliamo

$$P_0 = 7^3 P = (7190, 7003)$$

$$Q_0 = 7^3 Q = (2599, 759)$$

e troviamo che $Q_0 = 10P_0$, quindi $k_2 = 10$.

- Infine risolvendo la coppia di congruenze

$$k \equiv 218\ (mod\ 7^3)$$
$$k \equiv 10\ (mod\ 23)$$

otteniamo $k = 4334$. □

3.3 Metodo ρ di Pollard

L'idea su cui si basa il funzionamento di questo algoritmo è quella di trovare due coppie distinte (c', d') e (c'', d'') di interi modulo n tali che

$$c'P + d'Q = c''P + d''Q.$$

In questo caso avremo

$$(c' - c'')P = (d'' - d')Q = (d'' - d')kP$$

in modo che

$$(c' - c'') \equiv (d'' - d')k \bmod n.$$

Quindi, $k = \log_P Q$ può essere ottenuto calcolando

$$k = (c' - c'')(d'' - d')^{-1} \bmod n.$$

Si tratta cioè di considerare le triple $(c, d, cP+dQ)$, facendo variare $0 \leq c, d \leq n-1$ finché un punto $cP + dQ$ non si ottiene una seconda volta, cioè quando si verifica una collisione.
Per un risultato di statistica noto come il *Paradosso del compleanno*[2] se la scelta di c e d è casuale, ci aspettiamo di incontrare una collisione dopo un numero di iterazioni approssimabile con $\sqrt{\pi \frac{n}{2}}$.
Affinché la scelta degli interi c e d sia (pseudo)casuale si definisce una *funzione iterativa* $f : \langle P \rangle \longrightarrow \langle P \rangle$ nel seguente modo.
Sia $\{S_1, S_2, \cdots, S_L\}$ una casuale partizione di $\langle P \rangle$. Scriveremo $H(X) = j$ se $X \in S_j$ e chiameremo H *funzione di partizione.*
Inoltre sceglieremo due interi $0 \leq a_j, b_j \leq n-1$ per ogni $1 \leq j \leq L$. Allora $f : \langle P \rangle \longrightarrow \langle P \rangle$ è definita in modo che

$$f(X) = X + a_j P + b_j Q \text{ dove } j = H(X).$$

Osserviamo che se $X = cP + dQ$ allora $f(X) = \overline{X} = \bar{c}P + \bar{d}Q$ dove $\bar{c} = c + a_j \bmod n$ e $\bar{d} = d + b_j \bmod n$.

[2]Paradosso del compleanno: il numero minimo di persone che bisogna considerare affinché la probabilità che due di esse compiano gli anni nello stesso giorno sia pari a $1/2$, è 24 (valore che approssima $\sqrt{\pi \frac{365}{2}}$). Più in generale, ogniqualvolta si selezionano a caso degli elementi da un insieme di dimensione n, è sufficiente selezionarne $O(\sqrt{n})$ per avere una probabilità di $1/2$ di selezionare lo stesso elemento due volte.

Consideriamo adesso un punto $X_0 \in \langle P \rangle$ e determiniamo una sequenza $\{X_i\}_{i \geq 0}$ di punti tale che $X_i = f(X_{i-1})$ per $i \geq 1$.
Essendo l'insieme $\langle P \rangle$ finito, ad un certo punto si avrà un indice t tale che $X_t = X_{t+s}$ con $s \geq 1$.
Ciò significa che la sequenza X_i diventa periodica di periodo s, ossia

$$X_i = X_{i-s}.$$

La rappresentazione grafica di tale processo ricorda la lettera greca ρ da cui il nome di tale algoritmo (come si osserva dalla figura seguente).

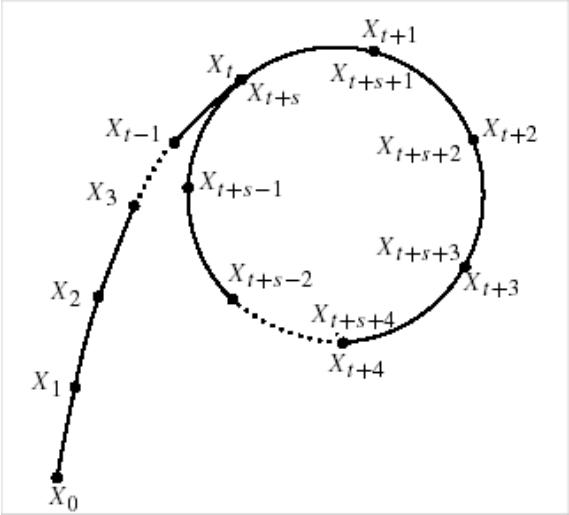

Figura 3.1: Rappresenta la sequenza X_i in un algoritmo ρ di Pollard

Il valore t è chiamato *lunghezza della coda*, mentre s è detto *lunghezza del ciclo*.
Se la funzione f è sufficientemente casuale, ci si aspetta una collisione in un tempo $O(\sqrt{n})$.

Un vantaggio non trascurabile dell'algoritmo ρ *di Pollard* è quello di utilizzare il *ciclo di Floyd*.
Il vantaggio di tale ciclo è il non dover trascrivere e/o conservare tutte le coppie. Il numero i di coppie da calcolare prima che $X_i = X_{2i}$ è tale che $t \leq i \leq t + s$.
In tale intervallo infatti esiste sicuramente un i tale che $X_i = X_{2i}$.
L'Algoritmo di Pollard completo è il seguente:

3.3.1 Algoritmo ρ di Pollard per l'ECDLP

Dato un punto $P \in \mathcal{E}(GF(q))$ di ordine il numero primo n e $Q \in \langle P \rangle$ calcolare il logaritmo discreto $k = \log_P Q$.

1. Selezionare il numero L.
2. Definire una funzione di partizione $H : \langle P \rangle \longrightarrow \{1, 2, ..L\}$.
3. Per $j = 1, .., L$
   - Selezionare $0 \leq a_j, b_j \leq n - 1$.
   - Calcolare $R_j = a_j P + b_j Q$.
4. Selezionare $0 \leq c', d' \leq n - 1$ e calcolare $X' = c'P + d'Q$.
5. Porre $X'' = X'$, $c'' = c'$ e $d'' = d'$.
6. Ripetere le seguenti operazioni:
   - Calcolare $j = H(X')$ e porre $X' = X' + R_j$, $c' = c' + a_j \mod n$, $d' = d' + b_j \mod n$.

- Per $i = 1, 2, ..$ calcolare $j = H(X'')$ e porre $X'' = X'' + R_j$, $c'' = c'' + a_j \bmod n$, $d'' = d'' + b_j \bmod n$.

fino a quando $X' = X''$.

7. Se $d' = d''$ si giunge ad un fallimento, altrimenti si calcola $k = (c' - c'')(d'' - d')^{-1} \bmod n$ e si restituisce k. □

L'algoritmo ρ di Pollard è considerato, ad oggi, uno dei metodi più veloce per risolvere l'ECDLP.

3.3.2 Esempio

Consideriamo la curva ellittica $\mathcal{E}$ definita sul campo $GF(229)$ con l'equazione:

$$y^2 = x^3 + x + 44.$$

Il punto $P = (5, 116) \in \mathcal{E}(GF(229))$ ha come ordine il numero primo $n = 239$. Consideriamo $Q = (155, 166) \in \langle P \rangle$.
Scegliamo come funzione di partizione $H : \langle P \rangle \longrightarrow \{1, 2, 3, 4\}$ con $L = 4$ tale che:

$$H(x, y) = (x \bmod 4) + 1,$$

e le quattro triple

$$[a_1, b_1, X_1] = [79, 163, (135, 117)]$$

$$[a_2, b_2, X_2] = [206, 19, (96, 97)]$$

$$[a_3, b_3, X_3] = [87, 109, (84, 62)]$$

$$[a_4, b_4, X_4] = [219, 68, (72, 134)].$$

Adesso scegliamo i due interi casuali $(c', d') = (54, 175)$ e consideriamo la seguente tabella:

$Passo$	c'	d'	X'	c''	d''	X''
0	54	175	$(39, 159)$	54	175	$(39, 159)$
1	34	4	$(160, 9)$	113	167	$(130, 182)$
2	113	167	$(130, 182)$	180	105	$(36, 97)$
3	200	37	$(27, 17)$	0	97	$(108, 89)$
4	180	105	$(36, 97)$	46	40	$(223, 153)$
5	20	29	$(119, 180)$	232	127	$(167, 57)$
6	0	97	$(108, 89)$	192	24	$(57, 105)$
7	79	21	$(81, 168)$	139	111	$(185, 227)$
8	46	40	$(223, 153)$	193	0	$(197, 92)$
9	26	108	$(9, 18)$	140	87	$(194, 145)$
10	232	127	$(167, 57)$	67	120	$(223, 153)$
11	212	195	$(75, 136)$	14	207	$(167, 57)$
12	192	24	$(57, 105)$	213	104	$(57, 105)$

L'algoritmo trova

$$192P + 24Q = 213P + 104Q,$$

e quindi

$$k = (192 - 213)(104 - 24)^{-1} \bmod 239 = 176.$$

□

3.4 Cenni sul metodo index–calculus

Il problema di verificare l'esistenza di algoritmi di tipo *index–calculus* che risolvano l'ECDLP se lo pose per primo Miller nel suo articolo del 1985 con il quale introdusse l'ECC. Egli riuscì ad osservare che nell'insieme $\mathcal{E}(GF(p))$, a differenza dei campi finiti $GF(p)$ o GF_{2^m}, la costruzione della *base dei fattori* S di elementi di G risulta complessa. La soluzione più immediata sembrerebbe quella di *sollevare* la curva $\mathcal{E}/GF(p)$ nella curva $\tilde{\mathcal{E}}$ definita sul campo dei razionali $\mathbb{Q}$ e nel costruire la *base* con i punti P_i di $\mathcal{E}/GF(p)$ tali che i corrispondenti punti $\tilde{P}_i$ siano $\mathbb{Z}$–indipendenti. Sia Miller che, successivamente, Silverman e Suzuki hanno osservato che tale soluzione non può funzionare. Ciò per due motivi. Prima di tutto perchè non si conosce un metodo efficiente per sollevare i punti di $\mathcal{E}(GF(p))$ in $\tilde{\mathcal{E}}(\mathbb{Q})$. Il secondo motivo è che solo un piccolissimo sottoinsieme di $\mathcal{E}(GF(p))$ risulta idoneo a formare la *base di fattori* S.

Un ulteriore attacco all'ECDLP, basato sull'idea di inveritire l'ordine di esecuzione delle varie fasi dell'*index–calculus*[3], è stato proposto da Silverman: l'algoritmo *xedni calculus*.

Varie analisi, comunque, hanno dimostrato che lo *xedni* è di scarso utilizzo pratico.

[3]Da ciò l'idea di utilizzare per tale algoritmo il nome *xedni* che corrisponde alla parola *index* invertita.

Capitolo 4

CONCLUSIONI

Una volta descritta l'ECC possiamo passare al confronto tra la sicurezza di questo tipo di crittografia e quella relativa ai problemi tradizionali della fattorizzazione degli interi e quello del logaritmo discreto. Viste le enormi differenze tra i sistemi utilizzati, il confronto si baserà sulla dimensione delle chiavi a parità di sicurezza.

Innanzitutto è necessario considerare che, a seconda della situazione, una chiave eccessivamente grande può influire negativamente sulle prestazioni di un algoritmo, dall'altra parte però l'utilizzo di chiavi troppo piccole può portare allo sviluppo di un sistema poco sicuro. Pertanto è necessario determinare il livello minimo della dimensione della chiave che garantisce un grado sufficiente di sicurezza.

Il confronto può, inoltre, essere effettuato relativamente ad un altro fattore: il *running–time*. I problemi IFP e DLP sono sostanzialmente equivalenti, quindi sono confrontabili a parità di

sicurezza, il problema del logaritmo discreto su curve ellittiche (ECDLP) è molto più complesso degli altri due. Ciò significa che a parità di lunghezza delle chiavi, gli algoritmi basati sul EC sono molto più sicuri dei sistemi RSA/DSA.

Le osservazioni effettuate per la chiave pubblica, però, ci consentono di confrontare anche le dimensioni della firma digitale. Per la firma valgono le stesse considerazioni, quindi il passaggio da un sistema RSA ad uno equivalente ECC comporta una riduzione della dimensione della chiave pubblica e della firma digitale. È normale concludere che ciò comporti un miglioramento prestazionale.

La crittografia basata su Curve Ellittiche è nata circa vent'anni fa. Il suo studio, inizialmente, fu affrontato in ambiti accademici, successivamente, grazie ai suoi importanti risultati, anche in ambiti che operano sulla sicurezza informatica. Il motivo di tutto questo interesse è, appunto, che l'ECC si basa su un problema matematico (ECDLP) molto complesso rispetto a quelli che hanno portato il successo dell'RSA e del DSA. Ciò porta ad affermare che l'ECC garantisce la stessa sicurezza con chiavi molto più piccole. Per tali motivi l'ECC e il relativo problema degli attacchi all'ECDLP hanno suscitato grande interesse in questi ultimi anni.

La diffusione della Crittografia su Curve Ellittiche è ancora modesta in confronto a quella dei sistemi tradizionali (RSA,

DSA, Diffie–Hellmann), anche se le prospettive sono molto promettenti soprattutto per la maggiore sicurezza ed efficienza che si prospetta per questo tipo di algoritmi.

Appendice A

CRITTOGRAFIA SU CURVE ELLITTICHE

A.1 Introduzione

L'utilizzo delle curve ellittiche in Crittografia si sviluppa grazie agli studi effettuati tra il 1985 e il 1987 da Victor Miller e Neil Koblitz. Furono i primi ad implementare sulle curve ellittiche gli algoritmi crittografici a chiave pubblica, già esistenti. Precisamente, Miller introdusse un analogo del protocollo Diffie–Hellman, mentre Koblitz un analogo del crittosistema ElGamal.

Nel 1991 fu presentato il primo analogo EC dell'RSA da Okamoto e Vanstone. Questo sistema si basa su un insieme di punti di una curva ellittica definita sull'anello $\mathbb{Z}_n$ (con n intero non primo). Studi successivi, però, mostrarono che tale sistema non apportava grossi miglioramenti rispetto al sistema originale.

Da questo momento in poi, quindi, la Crittografia basata su Curve Ellittiche (ECC) verrà considerata solo un'applicazione del Logaritmo Discreto sull'insieme dei punti di una curva ellittica (ECDLP).
I primi tre paragrafi di questo capitolo introducono dei concetti fondamentali dell'ECC: i paramentri di dominio, la generazione della coppia di chiavi e la rappresentazione del messaggio. Successivamente verranno introdotti alcuni algoritmi per produrre rispettivamente la Firma Digitale, la Cifratura e lo Scambio Chiavi.

A.2 Parametri di Dominio

Un crittosistema basato su Curve Ellittiche si avvale di un insieme di parametri che permettono di descrivere la curva ellittica $\mathcal{E}/GF(q)$: il campo finito $GF(q)$ su cui è definita, un punto P del campo ed il suo ordine n. Tali parametri devono essere opportunamente scelti in modo che l'ECDLP resista al maggior numero di attacchi possibili.
Gli elementi di tale insieme vengono detti *parametri di dominio*.

A.2.1 Definizione

I *parametri di dominio* di un crittosistema basato su curve ellittiche è l'insieme *D=(q,FR,S,a,b,P,n,h)*, dove:

- q è l'ordine del campo finito $GF(q)$;
- FR (*field representation*) indica la rappresentazione usata per gli elementi di $GF(q)$;
- S è il *random seed* che viene utilizzato nel caso in cui la curva ellittica venga generata in modo casuale;
- $a, b \in GF(q)$ sono i coefficienti dell'equazione di Weiestrass che descrive la curva $\mathcal{E}$;
- $P(x_p, y_p)$ è un punto di $\mathcal{E}(GF(q))$, con ordine primo, detto *punto base*;
- n è l'ordine di P;
- $h = \#\mathcal{E}(GF(q))/n$ è il *cofattore*. □

A.2.1 Criteri di scelta

I parametri di dominio sopra descritti devono essere scelti in modo da poter contrastare gli attacchi precedentemente visti. In particolare $\#\mathcal{E}(GF(q))$ deve essere divisibile per un primo n sufficientemente grande da poter resistere agli attacchi Pohling–Hellman e ρ di Pollard.

A.3 Generazione delle Chiavi

Innanzitutto dobbiamo ricordare che un crittosistema basato

su Curve Ellittiche è a *chiave pubblica*, quindi utilizzerà una coppia di chiavi. Tale coppia sarà associata ad un particolare insieme di paramentri di dominio *(q, FR, S, a ,b, P, n, h)*.
La chiave pubblica non è altro che un punto Q del gruppo $\langle P \rangle$ generato dal punto P, mentre la chiave privata, d, equivale al $\log_P Q$. Proprio per questo, la determinazione della chiave privata d a partire da quella pubblica Q, che non è altro che un ECDLP, deve essere resa intrattabile dalla scelta dei *parametri di dominio*.

A.3.1 Algoritmo. Generazione delle chiavi

Dati i parametri di dominio *D=(q, FR, S, a ,b, P, n, h)* calcolare la chiave pubblica Q e la chiave privata d.

1. Selezionare a caso $1 \leq d \leq n-1$;
2. Calcolare $Q = dP$;
3. Restituire (Q, d). □

A.3.1 Validazione delle Chiavi

L'obiettivo di tale operazione è quello di verificare che la chiave pubblica possegga certe proprietà.
Nel caso in cui la validazione restituisce esito positivo, ciò implica che la chiave privata è logicamente esistente, anche se non assicura

che essa sia stata veramente calcolata o che il proprietario ne sia in possesso.

A.3.2 Algoritmo. Validazione della chiave pubblica

Dati i parametri *D=(q, FR, S, a, b, P, n, h)* e la chiave pubblica $Q = (x_Q, y_Q)$, ci si aspetta una validazione o un rifiuto.

1. Verificare che $Q \neq \Theta$;
2. Verificare che x_Q e y_Q siano elementi di $GF(q)$;
3. Verificare che Q soddisfi l'equazione della curva $\mathcal{E}$, definita da a e b;
4. Verificare che $nQ = \Theta$;
5. Se almeno un controllo è fallito ritorna "rifiutata" altrimenti restituisce "accettata". □

Esistono vari metodi per verificare che $nQ = \Theta$. Per esempio, se $h = 1$ (che è il caso di curve ellittiche su campi primi che sono maggiormente usati in pratica), allora i punti 1, 2 e 3 dell'algoritmo precedente implicano che $nQ = \Theta$.

A.4 Rappresentazione dei messaggi

Esistono vari schemi diversi per rappresentare un messaggio su una curva ellittica.

Il metodo proposto da Koblitz è quello di rappresentare il messaggio come un punto della curva; in questo caso il messaggio prende il nome di *embedding*.

Sia $\mathcal{E}$ una curva ellittica del tipo $y^2 = x^3 + ax + b$ definita sul campo primo $GF(q)$. Sia p un primo tale che

$$p \equiv 3 (mod\ 4). \tag{A.1}$$

Sia m il messaggio, espresso come un intero tale che $0 \leq m \leq p/100 - 1$.

Definiamo $x_i = 100m + i$ con i che varia da 0 a 99. Tali valori appartengono tutti a $GF(q)$ dato che vale la relazione $i \leq x_i < p + (i - 100) < p$.

Per i che varia da 0 a 100, calcoliamo i valori di $z_i = x_i^3 + ax_i + b$ fino a quando non si ottiene un quadrato modulo p.

Visto il Criterio di Eulero[1], si può notare che se $s_i^{(p-1)/2} \equiv 1 (mod\ p)$ allora s_i è un residuo quadratico modulo p.

Data la A.1 si dimostra che la corrispondente radice quadrata è semplice da ricavare con: $y_i \equiv s_i^{(p+1)/4} (mod\ p)$. In questo modo si trova il punto $P_m = (x_i, y_i)$ sulla curva $\mathcal{E}$.

Per determinare il messaggio m da P_m basta calcolare $\lfloor x_i/100 \rfloor$ ossia il più grande intero minore o uguale di $x_i/100$.

Notiamo che la probabilità di non individuare un quadrato per i che varia da 0 a 99 è pari a $(1/2)^{100} = 2^{-100}$ che è trascurabile. Infatti, essendo s_i un elemento pseudo-casuale di $GF^*(p)$, che è

[1] Se p è un numero primo, per qualsiasi intero a vale $(\frac{a}{p}) = a^{(p-1)/2} (mod\ p)$.

ciclico e di ordine dispari, la probabilità che s_i sia un quadrato è pari circa ad 1/2.

A.5 Firma digitale

A.5.1 Introduzione

Nel primo capitolo abbiamo introdotto il concetto di Firma Digitale, diamone adesso una definizione più formale

A.5.1 Definizione. Uno schema di firma digitale Σ è costituito da quattro algoritmi:

1. Un *algoritmo per la generazione dell'insieme dei parametri di dominio*, cioè che genera D;

2. Un *algoritmo per la generazione delle chiavi* che prende in input l'insieme D e calcola (Q, d);

3. Un *algoritmo per la generazione della firma digitale* che considera come input l'insieme D, la chiave privata d, ed un messaggio m e produce la firma Σ;

4. Un *algoritmo per verificare la firma* che prende come input l'insieme D, la chiave pubblica Q, il messaggio m, e la presupposta firma Σ e accetta o rifiuta la firma. □

Introduciamo adesso un criterio di sicurezza per schemi di firma sviluppato da Goldwasser, Micali e Rivest (GMR).

A.5.2 Definizione. Uno schema di firma Σ è detto sicuro (o GMR–sicuro) se non può essere alterato da un avversario che abbia potenza di calcolo limitata e che possa eseguire attacchi di tipo *closen-message*. In altre parole, un avversario che può ottenere firme legittime di ogni messaggio di sua scelta non riesce a produrre una firma valida di un qualsiasi messaggio nuovo. □

In realtà, però, non sempre l'avversario è in grado di procurarsi le firme che desidera. Adesso presentiamo l'algoritmo di firma digitale basato su EC che ha raggiunto maggiore diffusione.

A.5.2 ECDSA

L'algoritmo ECDSA (Elliptic Curve Digital Signature Algorithm) è l'analogo, su curve ellittiche, dell'algoritmo DSA. Di seguito sono mostrati gli algoritmi di generazione e verifica della firma ECDSA dove $H(\cdot)$ indica una funzione hash.

A.5.3 Algoritmo. Generazione firma ECDSA
Dati i parametri di dominio $D = (q, FR, S, a, b, P, n, h)$, la chiave privata d e il messaggio m calcolare la firma (r, s).

1. Selezionare un intero k tale che $1 \leq k \leq n - 1$;

2. Calcolare $kP = (x_1, y_1)$ e considerare $r = x_1(mod\ n)$. Se $r = 0$ ritornare al punto 1;

3. Calcolare $e = H(m)$;

4. Calcolare $s = k^{-1}(e + dr)(mod\ n)$. Se $s = 0$ ritornare al punto 1;

5. Restituire (r, s). □

A.5.4 Algoritmo. Verifica firma ECDSA

Dati i parametri di dominio $D = (q, FR, S, a, b, P, n, h)$, la chiave pubblica Q, il messaggio m e la firma (r, s) accettare o rifiutare la firma.

1. Verificare che $1 \leq r, s \leq n-1$. Se ciò non si verifica, restituire "rifiuta";

2. Calcolare $e = H(m)$;

3. Calcolare $w = s^{-1}(mod\ n)$;

4. Calcolare $u_1 = ew(mod\ n)$ e $u_2 = rw(mod\ n)$;

5. Calcolare $X = u_1P + u_2Q$;

6. Se $X = \Theta$ restituire "rifiuta";

7. Calcolare $v = x_1(mod\ n)$;

8. Se $v = r$ allora ritorna "accetta" altrimenti ritorna "rifiuta". □

Dimostriamo che la verifica è corretta:
se (r, s) è la firma legittima del messaggio m, allora $s \equiv k^{-1}(e + dr)\ (mod\ n)$. Infatti:

$$k \equiv s^{-1}(e+dr)\ mod\ n \equiv s^{-1}e+s^{-1}dr \equiv we+wrd \equiv u_1+u_2d\ mod\ n$$

quindi: $X = u_1P + u_2Q = (u_1 + u_2d)P = kP$, e così $v = r$ come richiesto.

Affinché l'ECDSA sia GMR-sicuro, è necessario che l'ECDLP in $\langle P \rangle$ sia intrattabile e che la funzione hash $H(\cdot)$ sia crittograficamente sicura. Non è stato dimostrato, però, che queste condizioni siano anche sufficienti[2].

A.6 Cifratura a chiave pubblica

Gli algoritmi di cifratura a Chiave Pubblica permettono a due

[2]Il controllo nel punto 1 dell'algoritmo di verifica permette di prevedere degli attacchi; ad esempio nel caso in cui si ignora il controllo $r \neq 0$ e venga selezionato il punto base P con coordinata x nulla, ovvero $P = (0, \sqrt{b})$. In questo caso l'avversario può contraffare la firma di Alice su un qualsiasi messaggio m. Infatti è facilmente verificabile che ($r = 0$, $s = e$), con $e = H(m)$, è una firma valida di m: nel punto 4 si ottiene $u_1 = 1$ e $u_2 = 0$, quindi $X = P$ e infine $v = r = 0$, come è richiesto. Il *pre–message secrets* k nella generazione della firma è molto importante per la sicurezza dell'ECDSA. Se un avversario riuscisse ad ottenere il valore k che Alice utilizza per generare una firma (r, s) per un messaggio m, l'avversario è in grado di calcolare la chiave privata di Alice con: $d = r^{-1}(ks - e)\ (mod\ n)$ dove $e = H(m)$.

entità, Alice e Bob, di trasmettere in modo sicuro un messaggio.

A.6.1 ECIES

L'*Elliptic Curve Integrated Encryption Scheme (ECIES)* è stato proposto da Bellare e Rogaway, ed è una variante dello schema crittografico a chiave pubblica di ElGamal. È chiamato *integrato* perché riesce a combinare un sistema a chiave pubblica con un sistema simmetrico. In particolare, lo schema Diffie–Hellman viene utilizzato per ottenere le chiavi k_1 e k_2, dove le chiavi condivise, R e Z, vengono scelte dalla stessa entità. La chiave k_1 è usata per la cifratura simmetrica del messaggio m, k_2 invece fornisce l'autenticità del testo cifrato ottenuto. Gli algoritmi successivi mostrano le fasi di cifratura e decifratura.
Tale schema utilizza le seguenti funzioni:

- KDF$(\cdot)$: funzione di derivazione della chiave che, agendo come una funzione *hash*, genera la coppia di chiavi (k_1, k_2);
- $Enc_k(\cdot)$: funzione di cifratura di un sistema simmetrico, mentre $Dec_k(\cdot)$ di decifratura;
- $MAC_k(D)$: algoritmo per l'autenticazione del messaggio.

A.6.1 Algoritmo. Cifratura ECIES

Dati i paramentri di dominio *D=(q, FR, S, a, b, P, n, h)*, la

chiave pubblica Q e il messaggio m determinare il testo cifrato (R, C, t).

1. Selezionare $1 \leq k \leq n-1$;
2. Calcolare $R = kP$ e $Z = hkQ$. Se $Z = \Theta$ ritornare al punto 1;
3. Determinare $KDF(x_Z, R) = (k_1, k_2)$, dove x_Z è la coordinata x di Z;
4. Calcolare $C = Enc_{k_1}(m)$ e $t = MAC_{k_2}(C)$;
5. Ritornare (R, C, t). □

A.6.2 Algoritmo. Decifratura ECIES

Dati i paramentri di dominio *D=(q, FR, S, a, b, P, n, h)*, la chiave privata d e il testo cifrato (R, C, t) determinare il messaggio m o rifiutare il testo cifrato.

1. Eseguire una validazione della chiave R. Se la validazione fallisce restituire "testo cifrato rifiutato";
2. Calcolare $Z = (x_Z, y_Z) = hdR$. Se $Z = \Theta$ allora restituire "testo cifrato rifiutato";
3. Determinare $KDF(x_Z, R) = (k_1, k_2)$;
4. Calcolare $t' = MAC_{k_2}(C)$. Se $t \neq t'$ allora restituisci "testo cifrato rifiutato";

5. Calcolare $m = Dec_{k_1}(C)$ e restituirlo. □

Dimostriamo che la decifratura sia corretta:
se (R, C, t) è il legittimo testo cifrato del messaggio m, allora:

$$hdR = hd(kP) = hk(dP) = hkQ.$$

Il ricevente può generare la stessa coppia di chiavi (k_1, k_2) calcolate dal mittente e quindi può correttamente recuperare m. Possiamo inoltre osservare che il punto $Z = hkQ = hdR$ è un punto condiviso in modo segreto tra mittente e destinatario. Il punto $R = kP$ è invece una sorta di chiave pubblica. Possiamo infine osservare che le due chiavi simmetriche k_1 e k_2 dipendono, attraverso *KDF*, sia da Z che da R. La dipendeza da R non permette ad un avversario di sostituire R nel testo cifrato (R, C, t) ottenendo un altro testo cifrato (R', C, t) altrettanto valido per lo stesso messaggio m.

A.7 Scambio Chiavi

I protocolli di Scambio Chiavi sono caratterizzati da due o più entità che comunicano tra loro in modo sicuro attraverso l'utilizzo di una chiave segreta, da utilizzare con protocolli di cifratura simmetrici. La chiave condivisa viene detta *chiave di sessione*, ed in particolare con i protocolli di *key–transport*, un'entità

genera la chiave e la invia alla seconda entità. Nei protocolli *key–agreement*, invece, entrambe le entità contribuiscono a dare informazioni per calcolare la chiave da condividere.
In questo paragrafo ci soffermeremo su due protocolli *key–agreement* basati su EC, lo schema base Diffie–Hellman (ECDH) e l'ECMQV.

A.7.1 ECDH

Supponiamo che Alice e Bob condividano una chiave segreta grazie ad un reciproco scambio di messaggi. Il protocollo utilizzato è il seguente:

A.7.1 Protocollo. ECDH

1. Alice e Bob condividono i parametri di dominio *D=(q, FR, S, a, b, P, n, h)*;

2. Alice svolge le seguenti azioni:

   - Seleziona un intero casuale $1 \leq x \leq n-1$ e calcola $X = xP$;
   - Invia X a Bob;

3. Bob invece:

   - Seleziona un intero casuale $1 \leq y \leq n-1$ e calcola $Y = yP$;

- Invia Y ad Alice;

4. Alice riceve Y e calcola $K_A = xY = xyP$;

5. Bob riceve X e calcola $K_B = yX = yxP$;

6. Alice e Bob condividono la chiave comune $K = K_A = K_B$. □

I messaggi trasmessi (xP, yP) vengono detti *ephemeral public keys* (dall'inglese *effimere*) poichè sono una forma di chiavi pubbliche basate sul Logaritmo Discreto, ma durano per un breve periodo di tempo.
Così come avviene per il protocollo di Diffie–Hellman originale, il compito di ricavare xyP conoscendo xP e yP è detto *Elliptic Curve Diffie–Hellman Problem* (ECDHP). Quindi se si riesce a risolvere l'ECDLP, si può anche risolvere l'ECDHP, mentre l'implicazione inversa non è stata dimostrata.
Il sistema ECDH è molto versatile perché non richiede alcuna condivisione di informazioni a priori tra le due entità. Tale aspetto è anche il punto debole del protocollo, infatti:
se consideriamo la figura 4.1 essa mostra un attacco al protocollo ECDH.

Alice		**Joe**		**Bob**
x	$xP \rightarrow$	xP		
zP	$\leftarrow zP$	z		
		y	$\rightarrow yP$	yP
		wP	$\leftarrow wP$	w

Figura 4.1 Attacco a ECDH

In questo attacco Alice concorda la chiave $K_A = xzP$ con Joe, pensando di averla concordata con Bob. Allo stesso modo Bob concorda la chiave $K_B = ywP$ con Joe, convinto di comunicare con Alice. A questo punto Joe è in grado di scambiare messaggi con entrambi senza essere scoperto.
Il problema di questo protocollo è che Alice e Bob non conoscono la provenienza della *ephemeral public keys* ricevuta.
Un modo per risolvere questo problema è di firmare ogni messaggio con un opportuno algoritmo di firma. In questo modo Alice non invierà solamente xP ma il messaggio:

$$(xP, (r, s))$$

dove (r, s) è la firma ECDSA di xP.

A.7.2 ECMQV

L'*ECMQV (Elliptic Curve Menezes-Qu-Vanstone)* è un'evoluzione dell'ECDH. La differenza tra questi due protocolli riguarda

il fatto che questo prevede che ognuna delle due entità possegga la chiave pubblica dell'altra.
Ciò significa che:

- (Q_A, d_A) è la coppia di chiavi pubblica/privata di Alice;
- (Q_B, d_B) è la coppia di chiavi pubblica/privata di Bob;
- Alice ottiene in modo sicuro la chiave pubblica, Q_B, di Bob;
- Bob ottiene in modo sicuro la chiave pubblica, Q_A, di Alice.

Detto ciò, possiamo descrivere il funzionamento del protocollo di ECMQV:

A.7.2 Protocollo ECMQV

1. Alice e Bob condividono i parametri di dominio
   $D = (q, FR, S, a, b, P, n, h)$;
2. Alice svolge le seguenti azioni:
   - Seleziona un intero casuale $1 \leq x \leq n - 1$ e calcola $X = xP$;
   - Invia X a Bob;
3. Bob invece:
   - Seleziona un intero casuale $1 \leq y \leq n - 1$ e calcola $Y = yP$;
   - Invia Y ad Alice;

4. Alice calcola $s_A = (x + \overline{X} d_A)(mod\ n)$;

5. Bob calcola $s_B = (y + \overline{Y} d_B)(mod\ n)$;

6. Alice calcola $K_A = hs_A(Y + \overline{Y} Q_B)$;

7. Bob calcola $K_B = hs_B(X + \overline{X} Q_A)$;

8. Alice e Bob adesso condividono la chiave comune $K = K_A = K_B$. □

Possiamo, innanzitutto, osservare che la chiave comune K è un punto della curva ellittica e, facilmente, dimostrare che il protocollo è corretto, mostrando i valori K_A e K_B e, quindi, verificando la loro uguaglianza:

$$K_A = hs_A(Y + \overline{Y} Q_B) = hs_A(y + \overline{Y} d_B)P = hs_A s_B P$$

$$K_B = hs_B(X + \overline{X} Q_A) = hs_B(x + \overline{X} d_A)P = hs_B s_A P.$$

Il valore $s_A = (x + \overline{X} d_A)\ mod\ n$ è detto *firma implicita* di Alice per la chiave X essendo presente d_A che solamente lei può calcolare. Viene detta *implicita* perché viene verificata indirettamente da Bob attraverso l'uguaglianza:

$$s_A P = X + \overline{X} Q_A.$$

Sia Bob che Alice sono certi di condividere la chiave di sessione con l'altra entità, quindi, la chiave privata dell'avversario non consentirebbe di ottenere una chiave di sessione condivisa con una delle due entità.

Appendice B

Determinazione punti di $\mathcal{E}(GF(25))$

Consideriamo la curva $\mathcal{E}$ sul campo finito $GF(25)$ descritta dall'equazione $y^2 = x^3 + x + 1$. Il numero di punti di tale curva si ottiene mediante il metodo ingenuo in questo modo: si elencano tutti i possibili valori che può assumere $x \in GF(25)$, si calcolano i valori $x^3 + x + 1$ e successivamente si determina, se esiste, la radice quadrata.

Per una rappresentazione di $GF(25)$, consideriamo l'equazione $x^2 = 3$, irriducibile su $GF(5)$, e costruiamo il suo campo di spezzamento, aggiungendo una sua radice α. Per $x \in GF(5)$ valgono i calcoli già svolti nell'esempio 2.3.1, considerando stavolta che $y^2 = 3$ ha in $GF(25)$ due radici distinte:

x	$x^3+x+1 \pmod 5$	y	*Punti*
0	1	± 1	$(0,1),(0,4)$
1	3	$\pm\alpha$	$(1,\alpha),(1,-\alpha)$
2	1	± 1	$(2,1),(2,4)$
3	1	± 1	$(3,1),(3,4)$
4	4	± 2	$(4,2),(4,3)$

Consideriamo adesso gli altri 4 gruppi di cinque valori. A mo' di esempio, svolgiamo i calcoli per $x = \alpha$, invitando il lettore a valutare la laboriosità del metodo nel dover calcolare i rimanenti diciannove valori di x:

$\alpha^2 = 3 \Rightarrow \alpha^3 = 3\alpha \Rightarrow \alpha^3 + \alpha + 1 = 3\alpha + \alpha + 1 = 1 - \alpha.$

$(x, y) \in \mathcal{E}$ e $x = \alpha \Rightarrow y^2 = \alpha^3 + \alpha + 1 = 1 - \alpha.$

Poiché $GF(25)^*$ è ciclico, allora $y^{24} = 1$. Ne segue che $(y^2)^{12} = (1-\alpha)^{12} = 1$. Ma

$(1-\alpha)^3 = 1 - 3\alpha + 3\alpha^2 - \alpha^3 = -6\alpha + 10 = -\alpha$

$(1-\alpha)^{12} = (-\alpha)^4 = 3\alpha \neq 1$

quindi non esiste $y \in GF(25)$ tale che $y^2 = 1 - \alpha$. Per $x = \alpha$ cioè non otteniamo punti su $\mathcal{E}$.

x	$x^3 + x + 1 \pmod 5$	y	*Punti*
α	$1 - \alpha$		
$\alpha + 1$	$2\alpha + 2$		
$\alpha + 2$	$\alpha + 4$		
$\alpha + 3$	$\alpha + 3$	$\pm(2\alpha + 4)$	$(\alpha + 3, \pm(2\alpha + 4))$
$\alpha + 4$	2α		
2α	$\alpha + 1$		
$2\alpha + 1$	$2\alpha - 1$	$\pm(\alpha + 1)$	$(2\alpha + 1, \pm(\alpha + 1))$
$2\alpha + 2$	3	$\pm\alpha$	$(2\alpha + 2, \pm(\alpha))$
$2\alpha + 3$	4	± 2	$(2\alpha + 3, \pm 2)$
$2\alpha + 4$	$2\alpha + 3$		
3α	$1 - \alpha$		
$3\alpha + 1$	$3\alpha - 1$	$\pm(\alpha + 4)$	$(3\alpha + 1, \pm(\alpha + 4))$
$3\alpha + 2$	3	$\pm\alpha$	$(3\alpha + 2, \pm\alpha)$
$3\alpha + 3$	4	± 2	$(3\alpha + 3, \pm 2)$
$3\alpha + 4$	$3\alpha + 3$		
4α	$\alpha + 1$		
$4\alpha + 1$	$3\alpha + 2$		
$4\alpha + 2$	$4\alpha + 4$		
$4\alpha + 3$	$4\alpha + 3$	$\pm(2\alpha + 1)$	$(4\alpha + 3, \pm(2\alpha + 1))$
$4\alpha + 4$	3α		

Considerando inoltre il punto all'infinito abbiamo ottenuto che $\#\mathcal{E}(GF(25)) = 27$.

Bibliografia

[1] I. Blake, G. Seroussi, N. P. Smart, *'Advances in Elliptic Curve Cryptography'*, Cambridge University Press, 2005.

[2] Certicom, *Educational Newsletter 'Code & Cipher'*, http:\\www.certicom.com\codeandcipher.

[3] W. Chou, *'Elliptic Curve Cryptography and its Applications to Mobile Device'*, University Maryland Press, http:\\www.cs.umd.edu\Honors\reports\ECCpaper.pdf.

[4] W. Diffie, M. Hellman, *'New Directions in Cryptography'*, 1976, http:\\citeseer.ist.psu.edu.

[5] V. Gupta, S. Gupta, S. Chang, *'Performance Analysis of Elliptic Curve Cryptography for SSL'*, http:\\research.sun.com\projects\crypto.

[6] V. Gupta et al., *'Speeding up Secure Web Transaction Using Elliptic Curve Cryptography'*, http:\\research.sun.com\projects\crypto.

[7] R. Howlett, *'An Undergraduate Course in Abstract Algebra'*, http:\\math.usyd.edu.au\u\bobh\UoS\rfwhole.pdf.

[8] D. Husemoller, *'Elliptic Curves'*, Springer-Verlag, 1987.

[9] D. Kahn, *'The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet'*, Scribner, 1996.

[10] IETF, *'ECC Cipher Suites for TLS'*, http:\\www.ietf.org\internet-drafts\drafts-ietftls-ecc-12.txt.

[11] N.Koblitz, *'A Course in Number Theory and Cryptography'*, Springer, 1994.

[12] N. Koblitz, A. Menezes, *'A Survey of Public-Key Cryptosystems'*, http:\\www.cacr.math.uwaterloo.ca\∼ajmeneze\publications.

[13] N.Koblitz, A. Menezes, S. Vanstone, *'The State of Elliptic Curve Cryptography'*, Springer, 2004.

[14] J. Krasner, *'Using Elliptic Curve Cryptography for Enhanced Embedded Security'*, http:\\www.embedded-forecast.com\EMF–ECC–FINAL1204.pdf.

[15] S. Lang, *'Elliptic Curves; Diophantine Analysis'*, Springer-Verlang, Berlin and New York, 1978.

[16] A. Menezes, D. Hankerson, S. Vanstone, *'Guide to Elliptic Curve Cryptography'*, Springer, 2004.

[17] A. Menezes, P. Van Oorschot, S. Vanstone, *'Handbook of Applied Cryptography'*, CRC Press, 1997.

[18] E. Oswald, *'Introduction to Elliptic Curve Cryptography'*, http:\\www.iaik.tu-graz.ac.at\aboutus\people\oswald\.

[19] T.Satoh, B. Skjernaa, Y.Taguchi, *'Fast computation of canonical lifts of elliptic curves and its application to point counting'*, Finite Fields and Their Appl. 9 (2003), 89-101.

[20] R. Schoof, *'Elliptic curves over finite fields and the computation of square roots mod p'*, Math. Comp. 44 (1985), 170, 483–494.

[21] J. H. Silverman, *'The Arithmetic of Elliptic Curves'*, Springer-Verlag, 1986.