POLITECNICO DI TORINO

Corso di Laurea
in Ingegneria Matematica

Tesi di Laurea

Sistemi crittografici post quantum su curve ellittiche

Relatori
Dr.ssa Laura Capuano

. . . . . .

Candidato
Kairi Zuccarino

. . . . . .

Anno Accademico 2020-2021

A chi ha sempre creduto
in me

Prefazione

Negli ultimi decenni la potenza di calcolo dei computer è raddoppiata ogni 18 mesi grazie alla continua miniaturizzazione dei circuiti elettronici; tuttavia tale processo si sta arrestando a causa del raggiungimento della soglia della meccanica quantistica. Per questo motivo molte università e aziende private stanno investendo nella ricerca riguardante il computer quantistico: IBM ha presentato nel 2019 il primo computer quantistico commerciale e Google ha dichiarato di essere vicino alla costruzione di un computer quantistico. Dalla nascita dei computer quantistici sorge anche un grande problema, cioè quello della sicurezza informatica. I computer quantistici, infatti, sarebbero in grado di superare tutte le barriere della crittografia contemporanea, il che giustifica la grande attenzione verso lo sviluppo di nuovi protocolli crittografici resistenti ad attacchi da computer quantistici: la crittografia post-quantum. Il computer quantistico permette di immagazzinare le informazioni in q-bits (quantum bits), sovrapposizioni di diversi bit classici. In tal modo si ha una memoria stratificata data dalla sovrapposizione di stati diversi. Infatti mentre i bit classici possono assumere solo due stati, 1 o 0, i q-bit possono assumere come valori tutte le combinazioni lineari $v = \alpha 0 + \beta 1$ dove α e β sono numeri complessi che indicano la probabilità del bit di assumere rispettivamente valore 0 o 1. Quindi i q-bit possono essere rappresentati come una sfera unitaria nello spazio tridimensionale detta sfera di Bloch rappresentata in Figura 1, in cui il polo nord e il polo sud rappresentano rispettivamente gli stati 1 e 0.

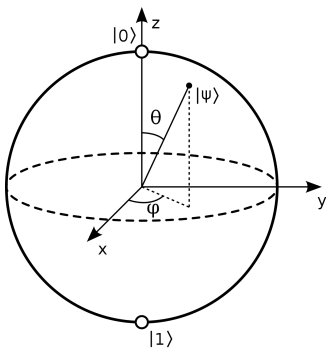

Figura 1. Sfera di Bloch

L'interesse per i computer quantistici deriva dall'algoritmo pubblicato nel 1994 dall'informatico teorico statunitense Peter Shor [13] in grado di fattorizzare gli interi in tempo polinomiale tramite l'utilizzo di un computer quantistico. Questo risulta essere un problema poichè molte crittografie tutt'ora utilizzate, come ad esempio RSA, basano la loro sicurezza sulla presunta difficoltà nella fattorizzazione di interi. Per questo motivo nel 2016 il National Institute of Standard Technologies ha indetto una competizione per la determinazione di nuovi protocolli crittografici che siano sicuri anche rispetto ad attacchi da parte di computer quantistici. Una delle proposte in gara usa la crittografia sui grafi di isogenie di curve ellittiche, argomento principale di questa tesi.

Nella prima parte del lavoro, quella relativa alle curve ellittiche, dopo aver dato alcuni richiami sulle estensioni dei campi, abbiamo definito le curve nel piano proiettivo, con una particolare attenzione alla definizione del piano proiettivo stesso e dei suoi legami con il piano euclideo. L'idea generale del piano proiettivo è la stessa di un viale alberato: anche se le due file di alberi sono parallele, ci sembrerà che esse si incontrino in un punto lontano. Nella geometria proiettiva tale punto viene detto punto all'infinito e fa si che tutte le rette in esso si incontrino in un unico punto.

Abbiamo poi definito le curve ellittiche, che sono curve lisce nel piano proiettivo molto utilizzate in crittografia per la struttura di gruppo che è possibile definire sui loro punti. Abbiamo analizzato due diverse forme in cui possono essere scritte tali curve: la classica forma di Weierstrass e quella di Montgomery, particolarmente utile dal punto di vista computazionale.

Successivamente siamo passati alla definizione di isogenie tra curve ellittiche, ossia morfismi che mandano il punto all'infinito della curva di partenza nel punto all'infinito della curva di arrivo. Dopo aver definito l'isogenia che compie la moltiplicazione per m e l'endomorfismo di Frobenius, ossia un'isogenia dalla curva in se stessa in cui ogni punto viene mandato in una potenza intera q delle sue coordinate, abbiamo definito l'isogenia duale che ha la particolarità di restituire, se composta con l'isogenia di partenza, la moltiplicazione per m. L'importanza dell'isogenia duale risiede nel fatto che essa è una sorta di inversa che in generale non è definibile dato che le isogenie non sono mappe iniettive.

Successivamente abbiamo analizzato in dettaglio le curve ellittiche su campi finiti, utili negli algoritmi utilizzati al giorno d'oggi, quali ad esempio RSA, e le curve ellittiche sul campo dei numeri complessi, che permettono di ricavare importanti proprietà, come ad esempio la relazione esistente tra un reticolo sul campo dei numeri complessi e una curva ellittica su tale campo. Per farlo è stato necessario riprendere alcuni concetti di analisi complessa tra cui la definizione di funzione di Weierstrass.

Per concludere la prima parte relativa alle curve ellittiche e alle isogenie tra di esse, abbiamo presentato il classico protocollo di scambio chiavi di Diffie-Hellmann su curve ellittiche che sfrutta proprio alcune delle proprietà delle curve ellittiche viste in precedenza. I protocolli di scambio chiavi sono molto importanti nella crittografia a chiave pubblica. Infatti la crittografia a chiave privata ha una sicurezza maggiore ed è più veloce, tuttavia ha il problema di scambiare una chiave condivisa tra più utenti. Per questo motivo viene spesso utilizzata la crittografia a chiave pubblica, più lenta, per effettuare tale scambio di chiavi. Di tale protocollo, oltre ad una descrizione più teorica, è stata fornita anche una sua implementazione in Python, per non dimenticare l'aspetto altamente applicativo dei

temi trattati.

Nella seconda parte dell'elaborato, dopo aver visto le proprietà delle isogenie, siamo passati al vero punto cardine di tutto il lavoro svolto: i grafi di isogenie. Essi sono particolari grafi i cui nodi sono i $j-$invarianti di curve ellittiche supersingolari (che descrivono classi di isomorfismo di curve ellittiche) e gli archi sono le isogenie tra di esse, che permettono di spostarsi tra curve ellittiche con diversi j-invarianti. Questi grafi sono grafi espansori che godono di proprietà particolarmente favorevoli alla definizione di protocolli crittografici. L'interesse verso tali protocolli è dettato dal fatto che essi risultano resistenti ad attacchi provenienti da computer quantistici in quanto, date due curve ellittiche, è difficile costruire un'isogenia di dato grado tra di esse, anche utilizzando un computer quantistico. In particolare, nella tesi, ci siamo occupati del SIDH (Supersingular Isogeny Diffie Hellmann), proposto per la prima volta dal matematico Luca De Feo, che è un protocollo di scambio di chiavi su tali grafi. L'idea di base per l'implementazione di tale protocollo è che Alice e Bob si muovono tramite cammini aleatori su due diversi grafi di isogenie con lo stesso insieme di nodi al fine di arrivare ad una chiave condivisa. Anche di tale protocollo abbiamo implementato un rudimentale codice Phyton che lo esegue e che mostra passo per passo i cammini che Alice e Bob devono compiere sui rispettivi grafi applicando diverse isogenie per arrivare alla chiave condivisa. Un esempio ottenuto tramite l'implementazione fornita, preso dall'articolo [9], si può osservare in Figura 2, in cui sia Alice che Bob partono dalla curva ellittica con j-invariante $190 + 87i$ definita su $\mathbb{F}_p$ in cui il primo $p = 431$. Alice e Bob si muovono sui grafi di isogenie di grado rispettivamente 2 e 3. Quindi tramite 4 isogenie di grado 2 Alice (percorso in giallo) arriverà al j-invariante $118 + 222i$ e, tramite 3 isogenie di grado 3, Bob (percorso in blu) arriverà al j-invariante $190 + 344i$. A questo punto Alice e Bob condivideranno le informazioni ottenute, e, muovendosi sui rispettivi grafi, arriveranno alla curva ellittica con j-invariante pari a 234, ossia la chiave condivisa. Il codice implementato per l'esempio, inoltre, può essere utilizzato per mettere in atto il protocollo in generale, modificando solo pochi parametri assegnati, come è stato fatto nell'ultima parte del lavoro.

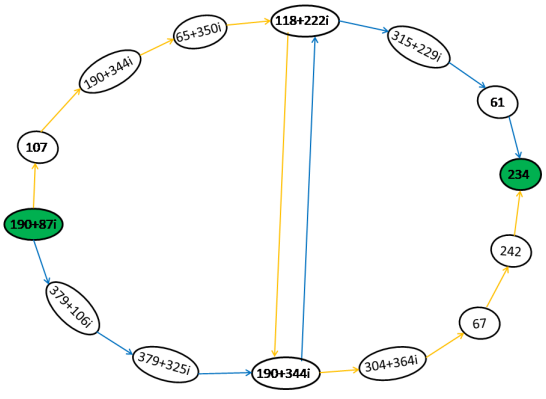

Figura 2. Esempio SIDH per $p = 431$.

Indice

Parte I

Curve ellittiche e applicazioni alla crittografia classica

Capitolo 1

Introduzione alle curve nel piano proiettivo

Alcuni dei protocolli crittografici attualmente utilizzati e quelli tutt'ora in studio volti alla crittografia post quantum affondano le radici nella teoria algebrica e geometrica delle curve ellittiche. Con l'obiettivo di comprenderne l'utilizzo e le proprietà che ne fanno uno strumento così largamente utilizzato in crittografia occorre introdurre la geometria proiettiva e le curve definite in essa. Ciò richiede anche conoscenze approfondite di algebra i cui concetti più importanti sono riportati nella sezione che segue.

1.1 Richiami sulle estensioni dei campi

Nel seguito della trattazione saranno di fondamentale importanza alcune nozioni della teoria dei campi utili alla costruzione dei protocolli crittografici. I concetti più importanti sono riportati in questa sezione e verranno ripresi nei capitoli successivi. Gli aspetti più basilari della teoria dei gruppi sono tralasciati ma possono essere facilmente reperiti ad esempio in [6].

Definizione 1.1.1. *Dati due campi $K \subseteq F$, si dice che F è un'estensione di K e viene denotata con F/K. Se la dimensione di F come spazio vettoriale su K è finita, viene detta grado dell'estensione e viene denotata con $[F : K]$.*

Definizione 1.1.2. *Siano $K \subseteq F$ due campi e $s \in F$; si dice che s è algebrico su K se è radice di un polinomio non nullo a coefficienti in K, cioè se esiste $P(x) \in K[x]\backslash\{0\}$ tale che $P(\alpha) = 0$ con $\alpha \in F$.*

Inoltre è utile osservare che, se s è algebrico su K, allora ogni elemento di $K(s)$ può essere scritto in modo unico come polinomio in s a coefficienti in K di grado inferiore al grado del polinomio minimo di s in K.
Da questa osservazione segue la seguente proposizione la cui dimostrazione può essere trovata in [6].

Proposizione 1.1.1. *Sia $K \subseteq K(s)$ un'estensione algebrica semplice. Se il polinomio minimo di s ha grado n, allora $[K(s) : K] = n$.*

Quindi si possono definire i campi di spezzamento che permetteranno a loro volta la definizione di estensioni normali e separabili.

Definizione 1.1.3. *Sia $f \in K[x]$; si dice che f spezza completamente in K se:*

$$f(x) = c\prod(x - a_i) \in K[x].$$

Definizione 1.1.4. *L'estensione si dice normale se e solo se, per ogni $f \in K[x]$, se f ha una radice in F, allora f spezza in F/K.*

Definizione 1.1.5. *Un'estensione F/K si dice separabile se il polinomio minimo di ogni elemento di F è separabile, cioè se ha tutte le radici distinte in un suo campo di spezzamento.*

Definizione 1.1.6. *Un'estensione normale e separabile viene detta estensione di Galois.*

Un'importante definizione per il seguito della trattazione è quella che segue:

Definizione 1.1.7. *Dato un gruppo G abeliano e finito, il sottogruppo di torsione G_{tor} è definito come segue:*

$$G_{tor} = \{g \in G \text{ tale che } ord(G) \text{ è finito}\},$$

dove l'ordine di elemento di un gruppo finito additivo è il più piccolo intero m tale che $mg = 0$.

1.2 Geometria proiettiva

L'interesse nei confronti della geometria proiettiva deriva dal fatto che essa è il punto d'incontro di tre diverse discipline matematiche quali algebra, geometria e teoria dei numeri. Di conseguenza seguiranno un'introduzione dal carattere geometrico e successivamente l'aspetto più algebrico della definizione del piano proiettivo e delle curve in di esso. Il legame tra il punto di vista algebrico e quello geometrico rimarrà sempre in primo piano nello sviluppo del lavoro.

Introduzione geometrica

Nel piano cartesiano due rette distinte possono avere o uno o nessun punto d'incontro come si osserva nella Figura 1.1. Nella geometria proiettiva ciò non è più valido poiché tutte le rette si incontrano in uno ed un solo punto, detto punto all'infinito nel caso in cui esse siano parallele, come si osserva nella Figura 1.2. Questo avviene tramite l'aggiunta dei punti all'infinito, ossia un punto per ogni direzione possibile nel piano cartesiano. Ciò porta a pensare al piano proiettivo denotato con $\mathbb{P}^2$ come l'unione del piano cartesiano $\mathbb{A}^2$ e delle direzioni nel piano cartesiano, denotate con $\mathbb{P}^1$. L'idea può sembrare ben lontana dalla realtà, tuttavia lo è molto di più di quanto si possa immaginare: infatti, basta pensare ad un viale alberato lunghissimo che alla vista pur essendo formato da due file di alberi parallele sembrerà avere un punto d'incontro.

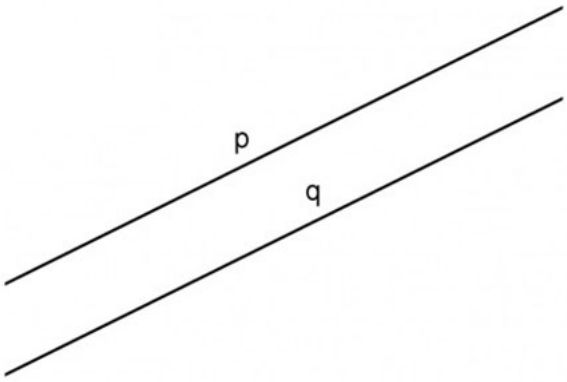

Figura 1.1. Le due rette parallele in figura non hanno un punto d'incontro.

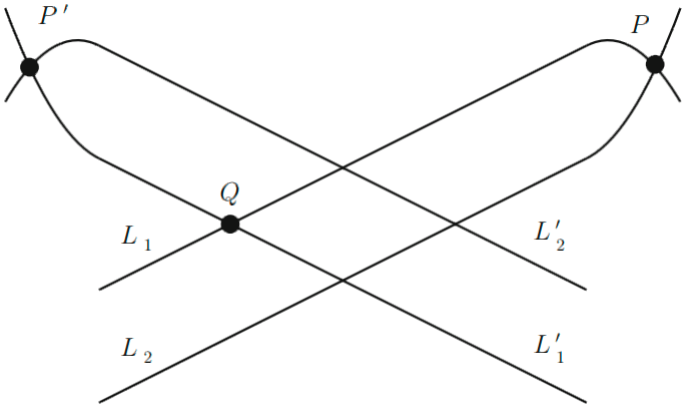

Figura 1.2. I punti P e P' fanno si che anche le coppie di rette L_1', L_2' e L_1, L_2 abbiano un punto d'incontro [3].

Teoria algebrica nella geometria proiettiva

Il piano proiettivo $\mathbb{P}^2$ può essere definito come:

$$\mathbb{P}^2 = \mathbb{A}^3 \backslash \{0\} / \sim,$$

in cui la relazione di equivalenza è definita nel modo che segue:

$$[a,b,c] \sim [a',b',c'] \quad \text{se} \quad (a,b,c) = \lambda\,(a',b',c') \quad con \quad \lambda \neq 0,$$

Dove i numeri a, b, c sono detti coordinate omogenee del punto [a,b,c]. Occorre osservare che all'interno del piano proiettivo esistono delle copie del piano euclideo e ciò può essere fatto in molti modi differenti tra cui i tre presentati che sono quelli canonici:

$$\begin{array}{lll} \phi_z : & \mathbb{A}^2 \longrightarrow \mathbb{P}^2 & \phi_y : \quad \mathbb{A}^2 \longrightarrow \mathbb{P}^2 \qquad \phi_x : \quad \mathbb{A}^2 \longrightarrow \mathbb{P}^2 \\ & (a,b) \longrightarrow [a,b,1] & (a,b) \longrightarrow [a,1,c] \qquad (a,b) \longrightarrow [1,b,c] \end{array}$$

Allo stesso modo possiamo creare le funzioni inverse che permettono di passare dalle tre coordinate del piano proiettivo a due del piano euclideo classico nel modo che segue:

$$\begin{array}{ccc} (\phi_z)^{-1}: & \mathbb{P}^2\backslash\{z\neq 0\} \longrightarrow \mathbb{A}^2 & \\ & [a,b,c] \longrightarrow (\frac{a}{c},\frac{b}{c}) & \end{array} \quad \begin{array}{ccc} (\phi_y)^{-1}: & \mathbb{P}^2\backslash\{y\neq 0\} \longrightarrow \mathbb{A}^2 & \\ & [a,b,c] \longrightarrow (\frac{a}{b},\frac{c}{b}) & \end{array} \quad \begin{array}{ccc} (\phi_x)^{-1}: & \mathbb{P}^2\backslash\{x\neq 0\} \longrightarrow \mathbb{A}^2 & \\ & [a,b,c] \longrightarrow (\frac{b}{a},\frac{c}{a}). & \end{array}$$

Grazie alle precedenti osservazioni si può infine osservare che l'aspetto geometrico precedentemente presentato e quello algebrico appena descritto sono equivalenti tra loro, come anticipato.

1.3 Curve nel piano proiettivo

Nel piano euclideo, una curva algebrica è il luogo dei punti in cui si annulla un polinomio. Allo stesso modo si vuole dare una definizione di curva nel piano proiettivo. Per farlo consideriamo un polinomio omogeneo $F(X,Y,Z)$, cioè tale per cui:

$$\exists\, d \geq 0 \quad \text{t.c.} \quad F(tx,ty,tz) = t^d F(x,y,z)\ \forall t \in \mathbb{R}$$

Tale proprietà implica che F è formato da monomi tutti dello stesso grado, i.e. $x^i y^j z^k$ con $i+j+k=d$. In particolare si osserva che, se (x_0,y_0,z_0) è tale che $F(x_0,y_0,z_0)=0$, allora per ogni $t \in \mathbb{R}$:

$$F(tx_0,ty_0,tz_0) = t^d F(x_0,y_0,z_0) = 0.$$

Definiamo quindi una curva proiettiva $C \subseteq \mathbb{P}^2$ come il luogo degli zeri di un polinomio omogeneo.

Omogeneizzazione

Dato un polinomio $f \in \mathbb{R}[x,y]$, si definisce il suo omogeneizzato come:

$$f^*(x,y,z) = z^d \left(\frac{x}{z},\frac{y}{z}\right) \text{ con } d = deg(f).$$

Tale operazione permette, data una curva affine, di definire la sua chiusura proiettiva. In generale, data C curva affine in $\mathbb{A}^2$ definita da un'equazione $f(x,y)=0$, si definisce la sua chiusura proiettiva:

$$\bar{C} = \{[x_0 : x_1 : x_2] \in \mathbb{P}^2 | f^*(x_0,x_1,x_2) = 0\}.$$

Esempio 1.3.1. *Sia $C \subseteq \mathbb{A}^2$la curva definita da $f(x,y)=0$, con*

$$f(x,y) = y^2 - x^3 - Ax - B,$$

la sua chiusura proiettiva $\bar{C} \subseteq \mathbb{P}^2$ sarà data dal luogo di zeri di $f^*(x,y,z)=0$, con

$$f^*(x,y,z) = z^3 f\left(\frac{x}{z},\frac{y}{z}\right) = \frac{y^2z^3}{z^2} - \frac{x^3z^3}{z^3} - A\frac{xz^3}{z} - Bz^3 = y^2z - x^3 - Axz^2 - Bz^3.$$

Deomogenizzazione (rispetto a una coordinata)

Viceversa, dato un polinomio omogeneo $f(x, y, z)$, si definisce il suo deomogeneizzato rispetto alla coordinata z il polinomio:

$$f_z(x, y) = f(x, y,1).$$

Notiamo che tale deomogeneizzazione può essere fatta rispetto ad ogni coordinata.
Osserviamo che questa operazione permette di passare da una curva proiettiva ad una affine. Infatti, data $\bar{C} \subseteq \mathbb{P}^2$ una curva proiettiva definita da $f(x, y, z) = 0$, allora una sua parte affine è:

$$\bar{C} \cap \{z \neq 0\} = \{(x, y) \in \mathbb{A}^2 \ : \ f_z(x, y) = 0\}.$$

Esempio 1.3.2. *Data la curva $\bar{C}$ nel piano proiettivo definita da:*

$$f(x, y, z) = y^2z - x^3 - Axz^2 - Bz^3 = 0,$$

la sua parte affine sarà ottenuta ponendo $z = 1$, cioè come luogo di zeri di:

$$f(x, y) = y^2 - x^3 - Ax - B = 0.$$

I risultati precedenti permettono di concludere che:

$$\begin{array}{rl} \bar{C} = & \{[x : y : z] \in \mathbb{P}^2 \ : \ y^2z = x^3 + Axz^2 + Bz^3\} \\ = & \{(x, y) \in \mathbb{A}^2 : y^2 = x^3 + Ax + B\} \cup [0 : 1 : 0], \end{array}$$

cioè la curva proiettiva definita dall'equazione:

$$y^2z = x^3 + Axz^2 + Bz^3,$$

è l'unione di una curva affine più un punto speciale, detto punto all'infinito. Ciò sarà utile nello studio delle curve ellittiche.

1.4 Intersezioni tra curve proiettive

Il risultato teorico che permette di determinare il numero di intersezioni tra curve nel piano proiettivo è dato dal teorema di Bezout. Per introdurlo è opportuno dare una definizione e un teorema importanti per la sua comprensione.

L'indice di intersezione tra due curve nel punto P, indicato da $I(C_1 \cap C_2, P)$ indica quanto queste curve sono tangenti, ossia ha valore 0 se le due curve non si intersecano, valore 1 se si intersecano in modo tangente e valore maggiore o uguale a 2 se si intersecano in modo tangente, come si osserva nella Figura 1.3. Ricordiamo il teorema fondamentale dell'algebra che può essere visto, geometricamente, come una descrizione del numero di intersezioni di una curva $y = f(x)$ nel piano affine, con l'asse y.

Teorema 1.4.1 (Teorema fondamentale dell'algebra)**.** *Dato un polinomio $f(x)$ di grado n, allora questo polinomio ha n radici su $\mathbb{C}$ contate con molteplicità, cioè:*

$$f(x) = c\prod_{i=1}^{n}(x - \alpha_i).$$

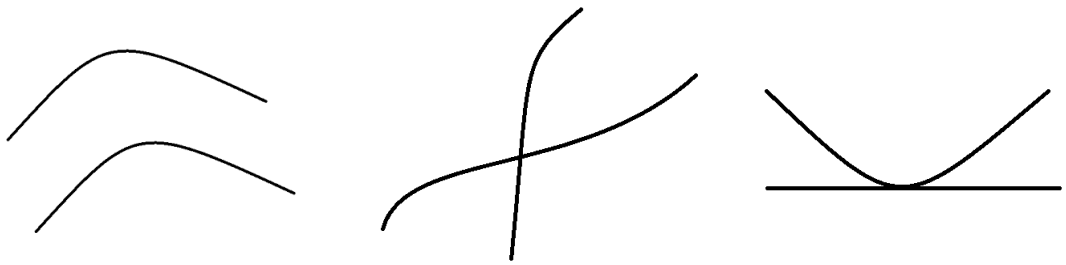

Figura 1.3. Nella prima immagine I=0 perché le curve non si intersecano, nella seconda I=1 perché si intersecano trasversalmente, nella terza $I \geq 2$ perché sono tangenti

Il teorema di Bezout è un'estensione geometrica del teorema fondamentale dell'algebra:

Teorema 1.4.2 (Teorema di Bezout)**.** *Siano C_1 e C_2 due curve proiettive senza componenti in comune; allora:*

$$\sum_{P \in C_1 \cap C_2} I(C_1 \cap C_2, P) = \deg(C_1)\deg(C_2).$$

Questo teorema fornisce un'informazione relativa al numero di intersezioni tra le due curve C_1 e C_2. Infatti, se le intersezioni in P sono tutte trasversali, allora si ha che $I(C_1 \cap C_2, P) = 1$ in tutti i punti P quindi:

$$\#C_1 \cap C_2 = \deg(C_1)\deg(C_2);$$

se invece le intersezioni non sono tutte trasversali allora si avrà la relazione:

$$\#C_1 \cap C_2 \leq \deg(C_1)\deg(C_2).$$

Questo teorema sarà particolarmente utile nel caso dell'intersezione tra una curva ellittica e una retta nel piano proiettivo come vedremo in seguito.

Capitolo 2

Curve ellittiche

2.1 Curve ellittiche in forma di Weierstrass

Le curve proiettive a cui siamo interessati sono definite da un'equazione della forma

$$Y^2 + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3, \tag{2.1}$$

detta "equazione di Weierstrass" . In generale, utilizzando le coordinate non omogenee $x = \frac{X}{Z}$ e $\frac{Y}{Z}$, possiamo riscrivere l'equazione come:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6.$$

Inoltre, se la caratteristica del campo è diversa da 2, allora possiamo effettuare la sostituzione $y \longmapsto \frac{1}{2}(y - a_1x - a_3)$; sostituendo si ottiene:

$$y^2 = 4x^3 + b_2x^2 + 2b_4x + b_6,$$

dove i coefficienti sono: $b_2 = a_1^2 + 4a_4,\ b_4 = 2a_4 + a_1a_3,\ b_6 = a_3^2 + 4a_6$. Infine supponendo che la caratteristica del campo non sia 2 o 3 possiamo effettuare l'ulteriore sostituzione $(x, y) \longmapsto \left(\frac{x-3b_2}{36}, \frac{y}{108}\right)$ e così otteniamo l'equazione:

$$y^2 = x^3 - 27c_4x - 54c_6,$$

in cui i coefficienti sono: $b_8 = a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2$, $c_4 = b_2^2 - 24b_4$, $c_6 = -b_2^3 + 36b_2b_4 - 216b_6$. Quindi, se il campo ha caratteristica diversa da 2 o 3, si può sempre supporre che la curva abbia una equazione di Weierstrass breve, cioè:

$$y^2 = x^3 + Ax + B. \tag{2.2}$$

A tale curva sono associate due quantità utili in futuro:

- Il discriminante $\Delta = -16(4A^3 + 27B^2)$;
- Il j-invariante $j = -1728\frac{(4A)^3}{\Delta}$.

Nel seguito sarà importante che le curve considerate siano non singolari, perciò siamo interessati a curve che siano abbastanza lisce. A tal fine viene data la seguente definizione:

Definizione 2.1.1. *Se il discriminante è tale che* $\Delta \neq 0$ *allora la curva proiettiva considerata viene detta curva ellittica.*

Siamo, inoltre, interessati alle curve ellittiche a meno di isomorfismo perché, come vedremo, è possibile costruire delle mappe tra curve ellittiche che siano uniche a meno di curve isomorfe ad esse. In particolare è importante la proposizione che segue poiché essa permetterà di identificare una curva ellittica con il suo $j-invariante$.

Proposizione 2.1.1. *Due curve ellittiche* E *ed* E' *sono isomorfe sulla chiusura di un campo* $\bar{K}$ *se e solo se hanno lo stesso* j*-invariante.*

Dimostrazione. "⇒"
Date le due curve ellittiche:

$$E : y^2 = x^3 + Ax + B,$$
$$E' : y'^2 = x'^3 + A'x' + B',$$

l'unico cambio di variabile che preserva la struttura della curva ellittica è il seguente:

$$(x, y) \to (u^2x', u^3y').$$

Sostituendo si osserva come si ottiene lo stesso j-invariante, infatti il j-invariante della curva E' è:

$$j = -1728\frac{(4A')^3}{\Delta'} = -1728\frac{(4A')^3}{-16(4A'^3 + 27B'^2)}.$$

Effettuando il cambio di coordinate precedentemente introdotto si osserva come la curva E possa essere scritta in funzione delle variabili x' e y':

$$(u^3y')^2 = (u^2x')^3 + Au^2x' + B,$$
$$u^6y'^2 = u^6x'^3 + Au^2x' + B;$$

da cui si ottengono le relazioni tra A e A' e tra B e B' seguenti:

$$A = u^4A' \quad B = u^6B'.$$

Quindi scrivendo il j-invariante in funzione di A' e B' si osserva che esso è uguale al j-invariante di E' già calcolato:

$$j = -1728\frac{(4u^4A')^3}{-16((4u^4A')^3 + 27(u^6B')^2)} = -1728\frac{64u^{12}A'}{-16[64u^{12}A'^3 + 27u^{12}B'^2]} =$$
$$= -1728\frac{(4A')^3}{-16(4A'^3 + 27B'^2)}.$$

"⇐" Se le curve E ed E' hanno lo stesso j-invariante allora possiamo scrivere:

$$\frac{(4A)^3}{4A^3 + 27B^2} = \frac{(4A')^3}{4A'^3 + 27B'^2} \Rightarrow A^3B'^2 = A'^3B^2.$$

Da tale uguaglianza si dimostra che le due curve sono legate dal seguente cambio di variabili: $(x, y) \to (u^2x', u^3y')$ in cui l'espressione di u dipende dai valori di A e B:

- $A = 0 \Rightarrow B \neq 0 \Rightarrow u = (\frac{B}{B'})^{\frac{1}{6}}$;
- $B = 0 \Rightarrow A \neq 0 \Rightarrow u = (\frac{A}{A'})^{\frac{1}{6}}$;
- $B \neq 0, A \neq 0 \Rightarrow u = (\frac{A}{A'})^{\frac{1}{4}} = (\frac{B}{B'})^{\frac{1}{4}}$.

□

Proposizione 2.1.2. *Per ogni $j_0 \in \bar{K}$ esiste una curva su $K(j_0)$ che ha j_0 come j-invariante.*

2.2 Legge di gruppo sulla curva ellittica

Le curve ellittiche sono importanti perché su di esse è possibile definire una legge di gruppo. Per capire al meglio la formula algebrica per tale definizione si osserva cosa accade a livello geometrico. Per farlo occorre ricordare un caso specifico del teorema di Bezout.

Osservazione 2.2.1. *Un retta e una curva ellittica si incontrano sempre in 3 punti. Infatti se si passa alle coordinate euclidee e si considera il sistema (2.3) per il calcolo delle intersezioni tra la curva e la retta:*

$$\begin{cases} f(x,y) = 0 \\ ax + by + c = 0 \end{cases} \tag{2.3}$$

e si suppone che $a \neq 0$, allora si può scrivere $x = -\frac{b}{a}y - \frac{c}{a}$ e quindi effettuare la sostituzione $f(x,y) = f(-\frac{b}{a}y - \frac{c}{a}, y)$. A questo punto $f(y)$ sarà un polinomio e avrà grado esattamente uguale al grado di f poiché dalle ipotesi del teorema di Bezout la retta non può essere una componente di $f(x,y)$ quindi il polinomio avrà tre zeri contati con molteplicità.

L'osservazione precedente permette di definire l'operazione tra punti di una curva ellittica che prende il nome di somma. Per farlo, come si osserva nella Figura 2.1, si traccia la retta passante per i due punti P e Q da sommare: tale retta incontrerà la curva in un altro punto (ciò è garantito dall'osservazione precedente). Infine viene preso il punto simmetrico rispetto all'asse x a quello così trovato. Nel caso in cui si voglia sommare tra P con se stesso,invece, si traccia la retta tangente alla curva nel punto P che incontrerà la curva stessa in un altro punto e si procede allo stesso modo del caso precedente (come si può osservare nella Figura 2.2). Occorre osservare che nella definizione delle curve ellittiche si è supposto $\Delta \neq 0$ proprio perché così facendo si elimina la presenza di nodi e cuspidi che non permetterebbero di trovare la somma in casi come questo a causa della non unicità della retta tangente nel nodo o nella cuspide.

Per dare una definizione algebrica occorre seguire gli stessi passi di quella geometrica. Si considera il caso in cui $P \neq Q$ e si procede come segue:

- L'equazione della retta passante per P e Q se essi sono distinti è data da:

$$y = \lambda(x - x_P) + y_P \quad \text{con} \quad \lambda = \frac{y_Q - y_P}{x_Q - x_P}.$$

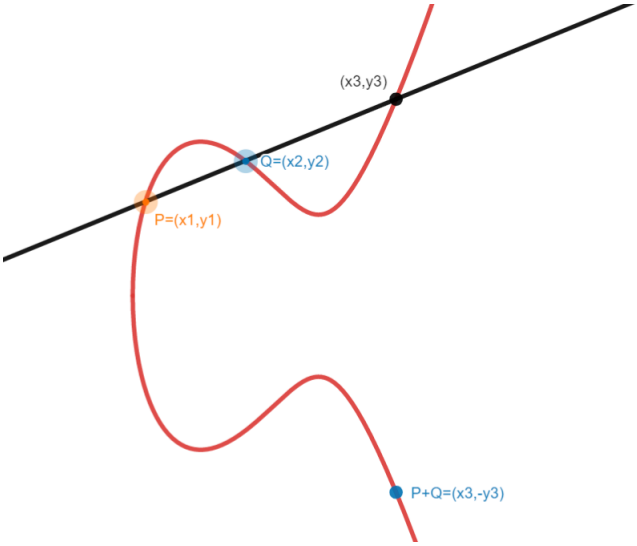

Figura 2.1. Per fare la somma tra P e Q tracciamo la retta e troviamo il punto di coordinate (x_3, y_3), così otteniamo che le coordinate di $P + Q$ saranno $(x_3, -y_3)$.

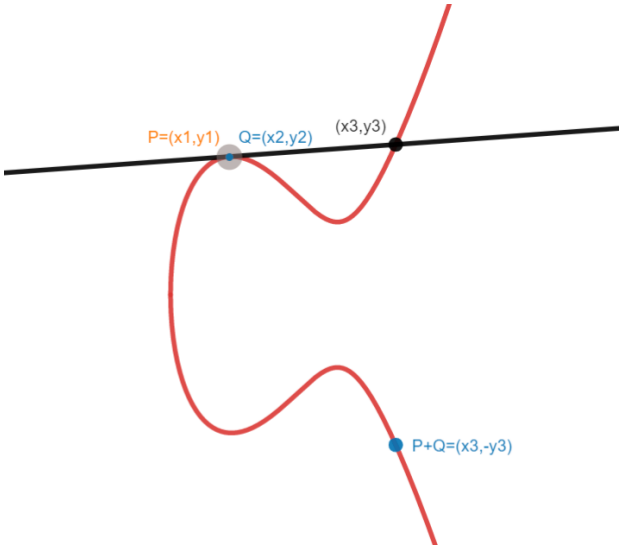

Figura 2.2. Per fare la somma tra P e Q con $P = Q$ tracciamo la retta tangente al punto P e troviamo il punto di coordinate (x_3, y_3), così otteniamo che le coordinate di $P + Q = 2P$ saranno $(x_3, -y_3)$.

- Sostituendo l'equazione della retta alla coordinata y della curva ellittica e così si ottiene l'equazione di terzo grado:

$$x^3 - \lambda^2 x^2 + ...$$

dove i termini successivi a quello di secondo grado non sono stati calcolati poiché non necessari in seguito.

- Per trovare la terza intersezione con la curva ellittica si osserva che la somma delle soluzioni di un'equazione di terzo grado è pari al coefficiente del termine di secondo grado cambiato di segno. Quindi è sufficiente scrivere:

$$x_P+x_Q+x_{P+Q} = -(-\lambda^2) \quad \Rightarrow \quad x_{P+Q} = \lambda^2-x_P-x_Q \quad \text{e} \quad y_{P+Q} = \lambda(x_{P+Q}-x_P)+y_P.$$

- Infine riflettendo rispetto all'asse x si ottiene la somma tra P e Q:

$$P + Q = \lambda^2 - x_P - x_Q, -\lambda(x_{P+Q} - x_P) + y_P).$$

Se invece si vuole calcolare $P + P = 2P$ occorrerà calcolare la retta tangente al punto P e trovare l'altra intersezione, dunque si avrà $\lambda = \frac{3x_P^2+A}{2y_P}$ e il punto cercato, procedendo come in precedenza, avrà coordinate:

$$2P = (\lambda^2 - 2x_P, \lambda(x_{2P} - x_P) + y_P).$$

L'insieme delle curve ellittiche con l'operazione appena definita è un gruppo il cui elemento neutro è il punto all'infinito. Infatti si può dimostrare che valgono le proprietà dei gruppi per ogni curva ellittica G.

- **Associatività:** $\forall P_1, P_2, P_3 \in G \quad \Rightarrow \quad (P_1 + P_2) + P_3 = P_1 + (P_2 + P_3)$.
- **Esistenza elemento neutro:** $\forall P \in G \quad \Rightarrow \quad P + O = O + P = P$.
- **Esistenza dell'inverso:** $\forall P \in G \quad \Rightarrow \quad P + P' = P' + P = O$.

La dimostrazione è omessa ma può essere trovata in [2]. Si può però osservare nella Figura 2.3 come nel caso dei punti scelti essa effettivamente sia valida.

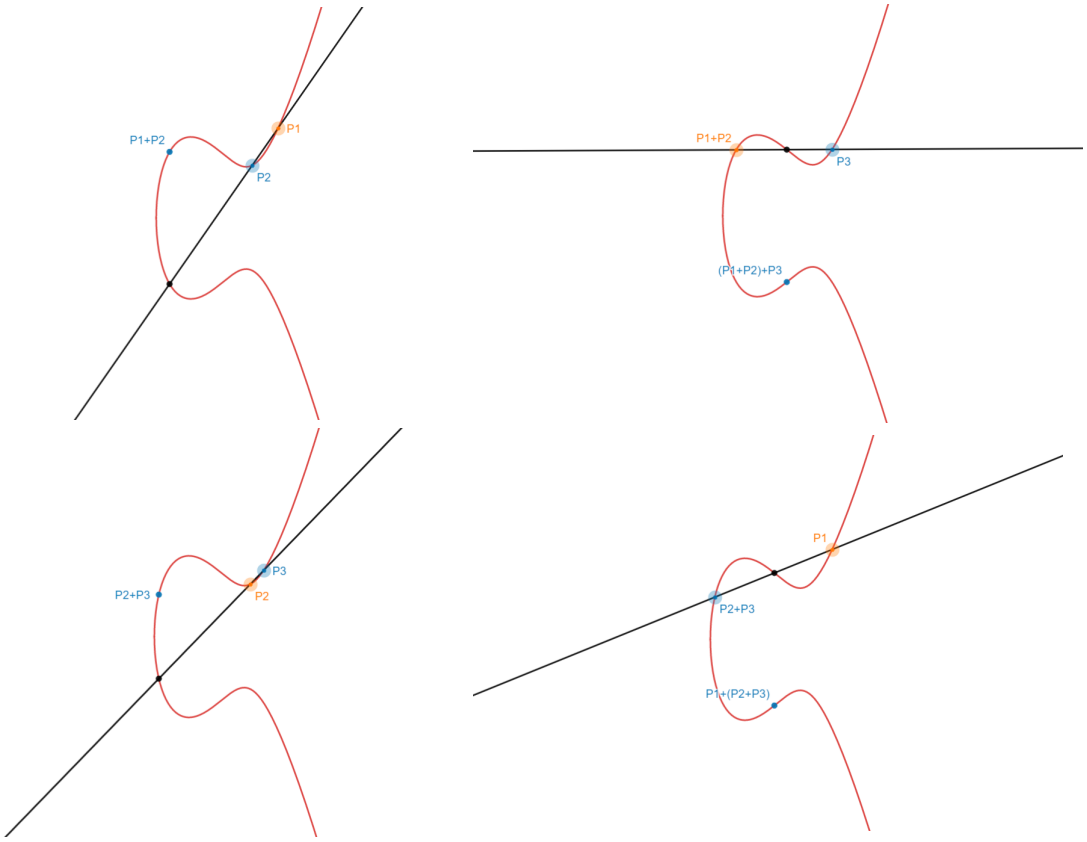

Figura 2.3. Le immagini sono state ottenute utilizzando il software di calcolo su curve ellittiche desmos. Nella prima immagine si osserva la somma $P_1 + P_2$, nella seconda $(P_1 + P_2) + P_3$; nella terza si calcola prima $P_2 + P_3$ e poi $P_1 + (P_2 + P_3)$. Si può quindi osservare come i due punti trovati siano uguali e dunque sia valsa la proprietà associativa nel caso considerato.

2.3 Curve ellittiche in forma di Montgomery

Fino ad ora si sono analizzate le curve ellittiche in forma di Weierstrass. Questa tuttavia non è l'unica forma utilizzata, infatti sella seconda parte della trattazione si osserverà come sia più conveniente computazionalmente utilizzare la forma di Montgomery. Ulteriori dettagli possono essere trovati in [7].

Definizione 2.3.1. *Una curva ellittica in forma di Montgomery è definita come:*

$$\mathcal{E} : \{(x, y) \in \mathbb{A}^2 \ : \ By^2 = x(x^2 + Ax + 1)\} \cup \{O\}$$

dove, come prima, denotiamo con $\mathcal{O}$ *il punto all'infinito.*

Analogamente a quanto fatto per le curve ellittiche in forma di Weierstrass, si può

calcolare il j-invariante che in questo caso dipenderà solo dal parametro A; si ha infatti:

$$j = \frac{256\left(A^2-3\right)^3}{A^2-4}.$$

Come per le curve ellittiche in forma di Weierstrass, la stessa costruzione geometrica della sezione precedente permette di definire una legge di gruppo.

- In questo caso, se si vogliono sommare i punti P e Q, il parametro λ è:

$$\lambda = \begin{cases} \left(y_Q - y_P\right) / \left(x_Q - x_P\right) & \text{se } P \neq Q \\ \left(3x_P^2 + 2Ax_P + 1\right) / \left(2By_P\right) & \text{se } P = Q \end{cases}$$

  e le coordinate di $P + Q$ sono:

$$x_\oplus = B\lambda^2 - \left(x_P + x_Q\right) - A,$$

$$y_\oplus = \lambda\left(x_P - x_\oplus\right) - y_P.$$

E' interessante per le applicazioni crittografiche definire esplicitamente le formule che consentono la moltiplicazione per 2 e per 3 di un punto di una curva ellittica in forma di Montgomery.
Si considera una curva ellittica in forma di Montgomery con $B = 1$, cioè di equazione:

$$E : y^2 = x(x^2 + Ax + 1).$$

Dato un punto $P = (x_P, y_P)$ applicando la formula per la somma si ha:

$$x_{2P} = \lambda^2 - \left(x_P + x_P\right) - A = \lambda^2 - 2x_P - A,$$

da cui, sostituendo il valore di λ dalla definizione:

$$x_{2P} = \left(\frac{3x_P^2 + 2Ax_P + 1}{2y_P}\right)^2 - 2x_P - A;$$

sostituendo a y_P^2 l'equazione stessa della curva data dalla definizione della curva ellittica otteniamo:

$$x_{2P} = \frac{(3x_P^2 + 2Ax_P + 1)^2}{4x_P(x_P^2 + Ax_P + 1)} - 2x_P - A,$$

da cui, Svolgendo il minimo comune multiplo si ha:

$$x_{2P} = \frac{(3x_P^2 + 2Ax_P + 1)^2 - 8x_P^2(x_P^2 + Ax_P + 1) - 4Ax_P(x_P^2 + Ax_P + 1)}{4x_P(x_P^2 + Ax_P + 1)}.$$

Svolgendo il quadrato e semplificando si ottiene la formula di duplicazione:

$$x_{2P} = \frac{x_P^4 - 2x_P^2 + 1}{4x_P(x_P^2 + Ax_P + 1)}. \tag{2.4}$$

Con calcoli molto simili si ottiene anche la formula di triplicazione che segue:

$$x_{3P} = \frac{x_P(x_P^4 - 6x_P^2 - 4Ax_P^3 - 3)^2}{(3x_P^4 + 4Ax_P^3 + 6x_P^2 - 1)^2}. \tag{2.5}$$

Capitolo 3

Isogenie tra curve ellittiche e loro proprietà

3.1 Introduzione alle isogenie

Dopo aver analizzato nel dettaglio le curve ellittiche e le applicazioni crittografiche tutt'ora in uso, rivolgiamo lo sguardo verso le mappe tra curve ellittiche che permetteranno di definire nuovi protocolli crittografici post quantum, scopo primario di questa tesi. In particolar modo rivolgiamo la nostra attenzione alle isogenie: mappe tra curve ellittiche con proprietà particolarmente interessanti.

Definizione 3.1.1. *Date due curve ellittiche E_1 ed E_2, un'isogenia tra esse è un morfismo*

$$\phi : E_1 \longrightarrow E_2 \quad \textit{tale che} \quad \phi(\mathcal{O}) = \mathcal{O}. \tag{3.1}$$

Due curve ellittiche E_1 ed E_2 si dicono isogene se esiste un'isogenia da E_1 in E_2 tale che $\phi(E_1) \neq \{O\}$.

Osservazione 3.1.1. *Poiché le curve ellittiche sono curve lisce per definizione, un'isogenia tra E_1 ed E_2 o è costante, e quindi manda ogni punto nel punto all'infinito poiché per definizione il punto all'infinito di E_1 deve essere mandato in quello di E_2, oppure è suriettiva, ossia $\phi(E_1) = E_2$.*

Quindi si può notare che dato ϕ isogenia non costante, per la precedente osservazione essa induce una mappa iniettiva:

$$\begin{aligned} \phi^* : \bar{K}(E_2) &\longrightarrow \bar{K}(E_1) \\ f &\longmapsto f \cdot \phi. \end{aligned}$$

In particolare quindi $\phi * (\bar{K}(E_2)) \subseteq \bar{k}(E_1)$ è un campo e l'estensione di campi è finita. Sfruttando le proprietà dei morfismi si riesce anche a definire il grado dell'isogenia come:

- $\deg \phi = 0$ se $\phi = 0$;

- se $\phi \neq 0$ $\deg \phi = [\bar{K}(E_1) : \phi^*(\bar{K}(E_2))]$.

Osservazione 3.1.2. *Le isogenie formano un gruppo rispetto all'operazione:*

$$\phi, \psi \in Hom(E_1, E_2) \quad (\phi + \psi)(P) = \phi(P) + \psi(P).$$

Chiamiamo endomorfismo una isogenia da una curva ellittica in se stessa.

Esempio 3.1.1 (Moltiplicazione per m)**.** *Un endomorfismo, la cui importanza verrà compresa nel seguito, è la moltiplicazione per m, indicata con $[m]$ e definita nel modo che segue:*

- *se* $m \geq 0$ $\quad [m] : E \longrightarrow E$, $P \longmapsto P + \cdots + P$
- *se* $m = 0$ $\quad [m] : E \longrightarrow E$, $P \longmapsto \mathcal{O}$,
- *se* $m \leq 0$ $\quad [m] : E \longrightarrow E$, $P \longmapsto -P - \cdots - P$

Infine, la seguente proposizione, oltre ad essere un interessante risultato algebrico, risulterà particolarmente utile nel seguito.

Proposizione 3.1.1. *Date E_1 ed E_2 curve ellittiche, allora $Hom(E_1, E_2)$ è uno $\mathbb{Z}-$modulo senza torsione.*

3.2 Endomorfismo di Frobenius

Nel caso di curve ellittiche su campi finiti, un endomorfismo molto importante è quello di Frobenius è un endomorfismo da una curva ellittica in se stessa che permette di verificare molte proprietà importanti delle isogenie. Per questo motivo è importante comprenderne la definizione. Per poter dare la sua definizione occorre la seguente proposizione:

Proposizione 3.2.1. *Dato un campo K con caratteristica $p \geq 0$ con $q = p^r$ elementi e una curva ellittica in forma di Weierstrass definita su esso $E : y^2 = x^3 + Ax + B$, allora anche la curva $E^q : y^2 = x^3 + A^q x + B^q$ è una curva ellittica.*

Dimostrazione. Per verificare che E^q è una curva ellittica è sufficiente osservare che:

$$(j_E)^q = j_{E^q},$$

dato che l'elevamento alla q è un omomorfismo, il che è facilmente verificabile. □

Definizione 3.2.1. *Si dice endomorfismo di Frobenius la seguente mappa:*

$$\phi_q : E \longrightarrow E, \quad (x, y) \mapsto (x^q, y^q),$$

il cui insieme di punti fissati è esattamente $E(\mathbb{F}_q)$.

3.3 Proprietà delle isogenie

Il primo teorema della sezione permette di avere un ulteriore punto di vista riguardo le isogenie ed è il seguente:

Teorema 3.3.1. *Sia $\phi : E_1 \longrightarrow E_2$ un'isogenia, allora ϕ è un morfismo di gruppi, ossia:*

$$\forall\, P, Q \in E_1 \quad \phi(P+Q) = \phi(P) + \phi(Q),$$

dove la somma a destra dell'uguale è quella definita tra punti della curva ellittica E_2.

Da tale teorema segue l'importante corollario:

Corollario 3.3.1. *Data $\phi : E_1 \longrightarrow E_2$ isogenia non nulla, allora $ker\phi \subseteq E_1$ è un sottogruppo finito di E_1.*

A questo punto è interessante la seguente proposizione, utile nella seconda parte della trattazione quando si vorranno definire i grafi di isogenie:

Teorema 3.3.2. *Sia E una curva ellittica e $\Phi \subseteq E$ un sottogruppo finito, allora esiste una curva ellittica E' unica a meno di isomorfismi, e un'isogenia separabile $\varphi : E \longrightarrow E'$ tale che $ker\varphi = \Phi$.*

3.4 Isogenia duale

Come spesso accade, dopo aver definito le isogenie e averne visto le caratteristiche principali, è naturale porsi la seguente domanda: esiste l'inversa di una isogenia? E' unica? E' facilmente calcolabile? In generale un'isogenia non è una mappa iniettiva, quindi non è possibile definire l'inversa. Tuttavia, data un'isogenia $\phi : E_1 \to E_2$ si può definire un'isogenia duale $\hat{\phi} : E_2 \longrightarrow E_1$ che funge da inversa di ϕ qualora essa esista. Con lo scopo di poter definire tale isogenia duale e successivamente studiarne le più importanti proprietà, diamo la seguente proposizione:

Proposizione 3.4.1. *Sia $\psi : C_1 \longrightarrow C_2$ mappa non costante tra curve lisce definite su un campo K perfetto con $charK \geq 0$. Allora ψ si fattorizza come*

$$\psi = \phi \circ \lambda,$$

con:

- $q = \deg(\psi)$*;*
- $\phi : C_1 \longrightarrow C_1^{(q)}$ *endomorfismo di Frobenius;*
- $\lambda : C_1^{(q)} \longrightarrow C_2$ *separabile.*

Ossia ψ può essere scomposta nella composizione di un morfismo di Frobenius e di una mappa separabile. Adesso è finalmente possibile definire l'isogenia duale.

Definizione 3.4.1 (Isogenia duale)**.** *Sia $\phi : E_1 \longrightarrow E_2$ un'isogenia non costante di grado m, esiste ed è unica l'isogenia $\hat{\phi} : E_2 \longrightarrow E_1$ tale che $\hat{\phi} \circ \phi = [m]$ ed essa prende il nome di isogenia duale.*

Osservazione 3.4.1. *E' interessante osservare che se il grado dell'isogenia ϕ è $m = 1$ allora l'isogenia duale coincide con l'isogenia inversa di ϕ.*

Dimostrazione. Prima di verificare l'esistenza si può verificare facilmente l'unicità. Supponendo che esistano due isogenie duali $\hat{\phi}$, $\hat{\phi}'$ si avrebbe che:

$$\left(\hat{\phi} - \hat{\phi}'\right) \circ \phi = [m] - [m] = 0,$$

ma ϕ non è costante quindi non può essere nulla, per cui si ottiene la tesi. Verificare l'esistenza è più complesso e richiede ulteriori nozioni algebriche, per cui tale dimostrazione è omessa. Tuttavia può essere reperita ad esempio in [4]. □

Teorema 3.4.1. *Sia ϕ un'isogenia si hanno le seguenti proprietà:*

- *Sia $m = \deg(\phi)$, allora*

$$\hat{\phi} \circ \phi = [m] \text{ su } E_1 \text{ e } \phi \circ \hat{\phi} = [m] \text{ su } E_2 \ ;$$

- *Sia $\lambda : E_2 \longrightarrow E_3$ un'isogenia, allora:*

$$\widehat{\lambda \circ \phi} = \hat{\phi} \circ \hat{\lambda};$$

- *Sia $\psi : E_1 \longrightarrow E_2$ un'altra isogenia, allora:*

$$\widehat{\phi + \psi} = \hat{\phi} + \hat{\psi};$$

- $\forall m \in \mathbb{Z}$*:*

$$\hat{[m]} = [m] \text{ e } \deg[m] = m^2;$$

- $\deg(\hat{\phi}) = \deg(\phi)$*;*
- $\hat{\hat{\phi}} = \phi$.

Dimostrazione. Si dimostrano singolarmente tutte le proprietà sopraelencate supponendo ϕ e una tra λ e ψ non costante altrimenti il teorema sarebbe banale.

- La prima parte deriva dalla definizione di $\hat{\phi}$; per la seconda invece:

$$(\phi \circ \hat{\phi}) = \phi \circ [m] = [m] \circ \phi,$$

  da cui poiché ϕ non è costante $\phi \circ \hat{\phi} = [m]$.

- Sia $n = \deg(\lambda)$, allora:

$$(\hat{\phi} \circ \hat{\lambda}) \circ (\lambda \circ \phi) = \hat{\phi} \circ [n] \circ \phi = [n] \cdot \hat{\phi} \circ \phi = [nm],$$

  allora per unicità si ottiene la tesi.

- La dimostrazione è più complessa ed è svolta in [2].
- Si dimostra per induzione. L'asserzione è vera per $m = 0$ e $m = 1$. Supponendo vero che $[\hat{m}] = [m]$, allora:

  $$[m \hat{+} 1] = [\hat{m}] + [\hat{1}] = [m] + [1] = [m + 1],$$

  dove per la prima uguaglianza si è utilizzata la proprietà precedente. Sia ora $d = \deg[m]$, considerando la mappa di moltiplicazione per m si ha che:

  $$[d] = [\hat{m}] \cdot [m] = [m] \cdot [m] = [m^2],$$

  ma poiché $End(E)$ è uno $\mathbb{Z} - modulo$ privo di torsione allora $d = m^2$.
- Sia $d = \deg(\phi)$, allora

  $$d^2 = \deg[d] = \deg(\phi \circ \hat{\phi}) = (\deg(\phi))(\deg(\hat{\phi})) = d \cdot (\deg(\phi)),$$

  da cui la tesi.
- Sia $d = \deg(\phi)$, allora:

  $$\hat{\phi} \circ \phi = [d] = [\hat{d}] = \widehat{\hat{\phi} \circ \phi} = \hat{\phi} \circ \hat{\hat{\phi}},$$

  per cui dall'unicità dell'isogenia duale si ottiene la tesi.

□

Capitolo 4

Altre proprietà delle curve ellittiche

Finora abbiamo visto la definizione delle curve ellittiche, la legge di gruppo e le mappe tra curve ellittiche. In realtà proprietà più fini delle curve in questione dipendono fortemente dalla natura del campo dove la curva ellittica è definita. In questo capitolo analizzeremo alcune proprietà delle curve ellittiche su campi finiti e delle curve ellittiche sul campo dei numeri complessi.

4.1 Curve ellittiche su campi finiti

Fino ad ora si sono considerate le curve ellittiche come curve a coefficienti in $\mathbb{R}$, cioè del tipo:

$$y^2 = x^3 + Ax + B \quad \text{con} \quad A, B \in \mathbb{R}.$$

Tuttavia l'interesse crittografico è rivolto alle curve ellittiche definite su campi finiti, cioè curve del tipo:

$$y^2 = x^3 + Ax + B \pmod{p} \quad \text{con} \quad A, B \in \mathbb{F}_q,$$

dove $q = p^n$ per qualche n. Questo è dato dal fatto che la presunta difficoltà del problema del logaritmo discreto su curve ellittiche permette, rispetto al problema del logaritmo discreto classico su campi finiti, di lavorare modulo p con un primo inferiore, il che rende gli algoritmi più efficienti. Da tali premesse scaturisce l'importanza di poter calcolare quanti punti appartengono alla curva definita in un campo finito, ossia dati p un numero primo e $q = p^n$ si vuole calcolare $\#(E|\mathbb{F}_q)$. Si può innanzi tutto osservare che se abbiamo una curva in forma di Weierstrass allora per ogni $x_0 \in \mathbb{F}_q$ l'equazione $f(x_0, y) = 0$ ha al più due soluzioni, quindi si ha la seguente maggiorazione per il numero di punti:

$$\#(E|\mathbb{F}_q) \leq 2q + 1,$$

dove q è il numero massimo di punti x_0 che soddisfano l'equazione a cui viene aggiunto il punto all'infinito. Inoltre, se il polinomio $f(x_0, y)$ si comporta in maniera casuale al

variare di $x_0 \in \mathbb{F}_q$ allora esso sarà irriducibile il 50% delle volte quindi ci si aspetta che:

$$\#(E|\mathbb{F}_q) \sim q.$$

Un teorema interessante in tal senso è quello di Hasse, prima di enunciarlo e dimostrarlo occorrono le nozioni preliminari che seguono.

Definizione 4.1.1. *Sia A un gruppo abeliano, allora $d : A \longrightarrow \mathbb{R}$ è detta forma quadratica se soddisfa:*

- $d(a) = d(-a) \quad \forall a \in A$;
- $A \times A \longrightarrow \mathbb{R} \quad$ *tale che* $\quad (a, b) \to d(a + b) - d(a) - d(b)$ *è bilineare.*

Inoltre essa è definita positiva se:

- $d(a) = 0 \iff a = 0$;
- $d(a) \geq 0 \quad \forall a \in A$.

Lemma 4.1.1. *La funzione $deg : Hom(E_1, E_2) \longrightarrow \mathbb{Z} \subseteq \mathbb{R}$ è una forma quadratica definita positiva.*

Lemma 4.1.2. *Sia d una forma quadratica, per ogni $\phi, \psi \in A$ si ha*

$$\#(d(\psi - \phi) - d(\phi) - d(\psi)) \leq 2\sqrt{d(\phi)d(\psi)}.$$

A questo punto è possibile enunciare il teorema di Hasse che da una maggiorazione più accurata per il numero di punti della curva ellittica.

Teorema 4.1.1 (Teorema di Hasse)**.** *Data una curva ellittica E definita su $\mathbb{F}_q$, vale la seguente relazione:*

$$|\#(E|\mathbb{F}_q) - (q + 1)| \leq 2\sqrt{q}.$$

Dimostrazione. Sia ϕ l'endomorfismo di Frobenius così definito:

$$\begin{aligned} \phi : E &\to E \\ (x, y) &\longmapsto (x^q, y^q), \end{aligned}$$

Allora $P \in E(\mathbb{F}_q) \iff \phi(P) = P \iff (1 - \phi)(P) = 0$; quindi si ha che:

$$\#(E|\mathbb{F}_q) = \#(ker(1 - \phi)) = deg(1 - \phi),$$

dove l'ultima uguaglianza è data dal fatto che ϕ è separabile. A questo punto applicando il Lemma (4.1.2) con $\psi = 1$ e $\phi = Frobenius$ si ottiene la tesi. □

Infine definiamo le curve ellittiche supersingolari, utili nel seguito.

Definizione 4.1.2 (Curve ellittiche supersingolari)**.** *Sia K un campo di caratteristica p e sia E una curva ellittica tale che*

$$E[p^n](\bar{K}) = \begin{cases} 0 & oppure \\ \mathbb{Z}/p^r\mathbb{Z} & r = 1,2,3,\dots \end{cases}$$

allora se E soddisfa la prima condizione essa viene detta curva ellittica supersingolare.

Più intuitivamente la curva è supersingolare se nessuno dei suoi punti, anche a coordinate nella chiusura del campo considerato, ha ordine p.
Una volta definite le curve ellittiche supersingolari è importante analizzare alcune delle loro proprietà che saranno utili nel seguito.

- La prima proprietà è che l'anello degli endomorfismi di una curva ellittica supersingolare sulla chiusura algebrica di un campo è particolarmente grande. Infatti per una curva ellittica supersingolare la sua algebra degli endomorfismi è un ordine in un'algebra dei quaternioni.
- Un'altra proprietà importante nella costruzione dei grafi di isogenie è che il $j-$invariante di una curva ellittica supersingolare definita su $\mathbb{F}_p$ appartiene a $\mathbb{F}_{p^2}$.

Un esempio di curva ellittica su un campo finito è la curva $y^2 = x^3 + 7x + 3 \pmod{113}$ rappresentata nella Figura 4.1.

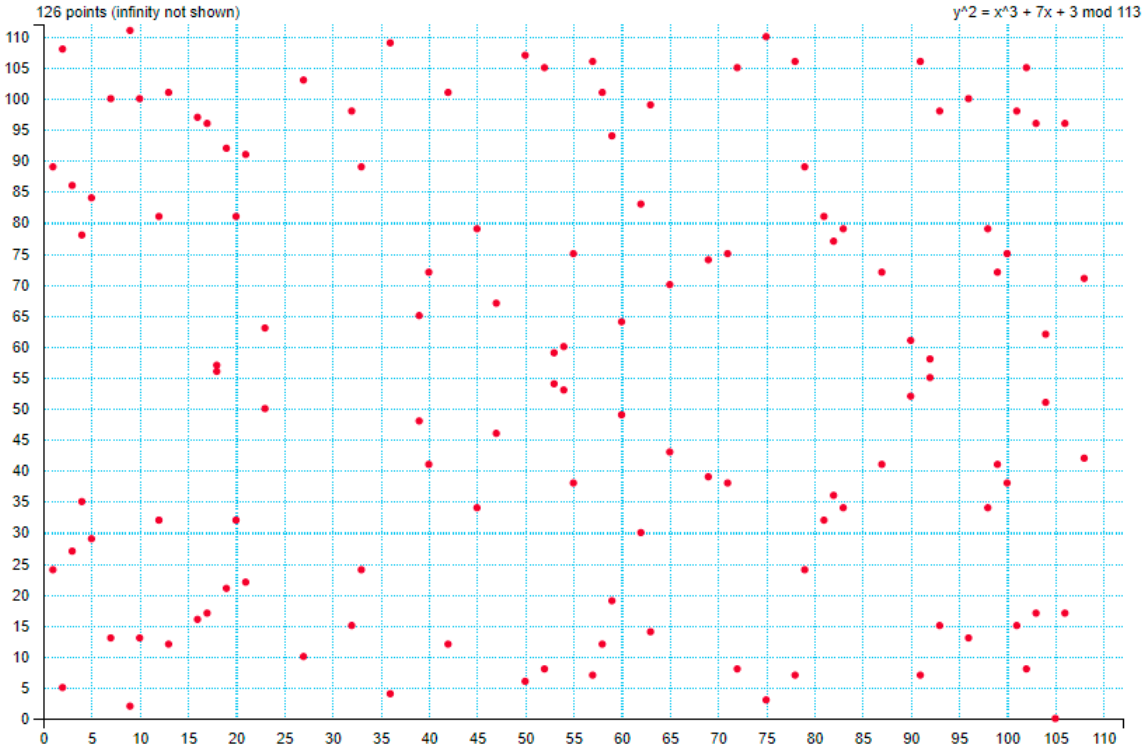

Figura 4.1. Nella figura si osservano i punti della curva

4.2 Curve ellittiche su $\mathbb{C}$

L'obiettivo della sezione è quello di mettere in corrispondenza le curve ellittiche su $\mathbb{C}$ e i tori su $\mathbb{C}/L$ con L reticolo. Il vantaggio nel fare ciò è dato principalmente da due motivi:

- comprendere le proprietà aritmetiche dei reticoli risulta più semplice che comprendere le proprietà della curva ellittica corrispondente;
- grazie alla scelta di reticoli appropriati è possibile costruire curve ellittiche con un numero di punti razionali fissato.

Definizione 4.2.1. *Un reticolo complesso L generato da una base ω_1, ω_2 tale che $\omega_1 \neq \lambda\omega_2$ per ogni $\lambda \in \mathbb{R}$ è un oggetto della forma:*

$$L = \omega_1\mathbb{Z} + \omega_2\mathbb{Z}.$$

Se L è generato da ω_1, ω_2, scriveremo $L = [\omega_1, \omega_2]$

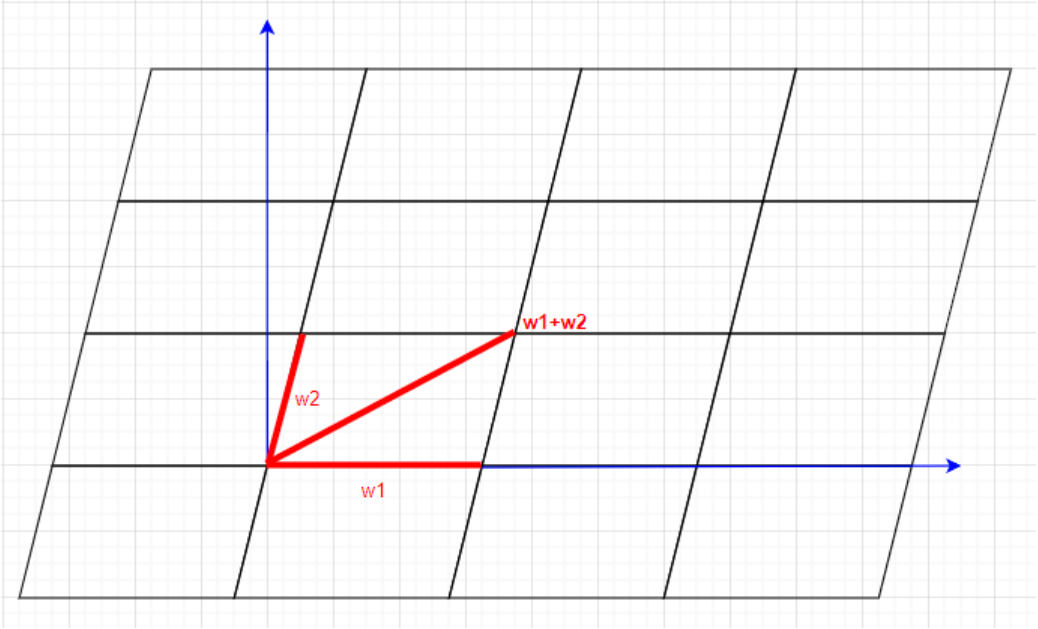

Figura 4.2. Reticolo su $\mathbb{C}$.

Un esempio può essere visto nella Figura 4.2.

Definizione 4.2.2. *Un parallelogramma si dice fondamentale per* $L = [\omega_1, \omega_2]$ *se è un qualsiasi insieme della forma:*

$$\mathcal{F}_\alpha = \{\alpha + t_1\omega_1 + t_2\omega_2 : \alpha \in \mathbb{C}, O \leq t_1, t_2 < 1\}.$$

Notiamo che se quozientiamo $\mathbb{C}$ per un reticolo L, lo spazio risultante sarà un toro. Infatti un punto in un parallelogramma di tale reticolo è equivalente ad un punto del parallelogramma fondamentale tramite la relazione di equivalenza seguente:

$$\text{dati} \quad m, n \in \mathbb{C} \quad m \sim n \iff m - n \in L,$$

quindi lo spazio quoziente $\mathbb{C}/L$ è in corrispondenza con un parallelogramma fondamentale. Osservando in particolare che i punti sui lati paralleli del parallelogramma sono equivalenti si può pensare al parallelogramma fondamentale come ad un toro in $\mathbb{C}$. Infatti come si può osservare nella Figura 4.3 ogni punto n è equivalente al punto m con la relazione di equivalenza sopra citata. Si osserva inoltre che anche i punti al bordo del parallelogramma sono equivalenti allo stesso punto sul lato parallelo a quello su cui si trovano. Di conseguenza lo spazio quoziente può essere visto come un incollamento tra la base superiore e la base inferiore del parallelogramma. Allo stesso modo si possono incollare i due lati obliqui ottenendo così un toro.

Una definizione rilevante per la trattazione deriva dall'analisi complessa ed è quella che segue:

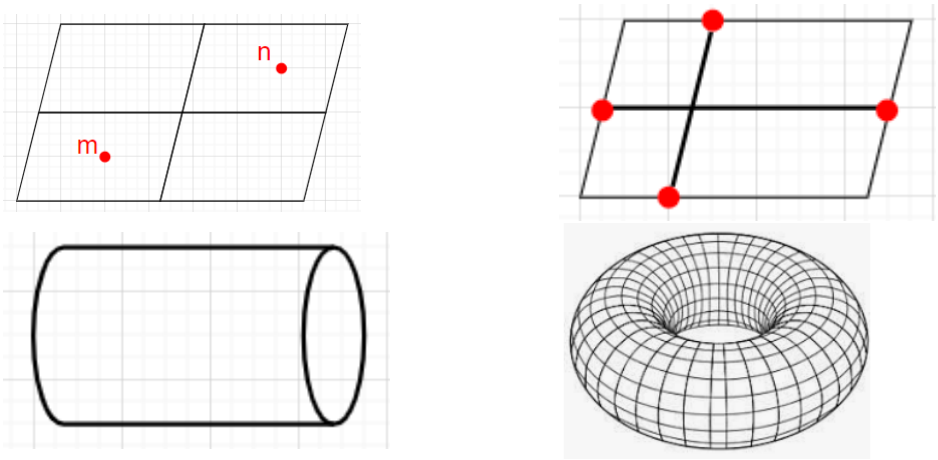

Figura 4.3. Equivalenza tra toro e reticolo in $\mathbb{C}$

Definizione 4.2.3. *Una funzione ellittica per un reticolo L è una funzione $f(z)$ tale che:*

- *f meromorfa, ossia è olomorfa su L tranne che in un insieme discreto di poli.*
- *f periodica rispetto ad L (cioè $f(z+\omega) = f(z) \quad \forall \omega \in L$).*

In particolare l'ordine di una funzione ellittica è il numero di poli che ha in un parallelogramma fondamentale contati con molteplicità.

Funzione $\wp$ di Weierstrass

L'interesse per questa funzione è dato dal fatto che tramite essa possono essere costruite tutte le altre funzioni ellittiche.

Definizione 4.2.4. *La funzione $\wp$ di Weirstrass di un reticolo L è definita da:*

$$\wp := \wp(z, L) := \frac{1}{z^2} + \sum_{\omega \in L^*} \left(\frac{1}{(z-\omega)^2} - \frac{1}{\omega^2} \right). \tag{4.1}$$

Dalla definizione si può facilmente calcolare anche la derivata della funzione di Weierstrass che sarà la seguente:

$$\wp'(z) = -2 \sum_{\omega \in L} \frac{1}{(z-\omega)^3}.$$

Teorema 4.2.1. *Per ogni reticolo L, la funzione $\wp$ e la sua derivata $\wp'$ soddisfano le seguenti proprietà:*

- *$\wp(z)$ è una funzione ellittica rispetto a L di ordine 2 i cui poli sono punti di L;*
- *$\wp'(z)$ è una funzione ellittica rispetto a L di ordine 3 i cui poli sono punti di L.*

Definito $\mathbb{C}(L)$ il campo delle funzioni ellittiche relative al reticolo L il teorema che segue mette in evidenza l'importanza della funzione $\wp$ di Weierstrass definita precedentemente.

Teorema 4.2.2. *Sia L un reticolo, allora:*

$$\mathbb{C}(L) = \mathcal{C}(\wp(z), \wp'(z)).$$

Il teorema precedente afferma che ogni funzione ellittica è una combinazione razionale di $\wp$ e $\wp'$. La dimostrazione può essere trovata nel Capitolo VI di [2]. Per la comprensione della sezione successiva sono inoltre necessari la seguente definizione e teorema.

Definizione 4.2.5. *Sia L un reticolo e sia $k \geq 0$ un intero; la serie di Einstein di peso k è la somma:*

$$G_k(L) = \sum_{\omega \in L^*} \frac{1}{\omega^k} \quad \text{con } L^* = L \backslash \{0\}. \tag{4.2}$$

Teorema 4.2.3. *Sia L un reticolo, l'espansione in serie di Laurent di $\wp(z, L)$ è:*

$$\wp(z, L) = \frac{1}{z^2} + \sum_{n=1} (2n+1) G_{2n+2}(L) z^{2n},$$

dove $G_k(L)$ è la serie di Einstein di peso k.

Relazione tra reticolo e curva ellittica su $\mathbb{C}$

Vogliamo ora mostrare che esiste una corrispondenza tra curve ellittiche e tori complessi. Infatti si può costruire la mappa seguente che associa una curva ellittica al reticolo L:

$$\begin{aligned} \Phi : \mathbb{C}/L &\to E/\mathbb{C} \\ z &\longmapsto \begin{cases} (\wp(z), \wp'(z)) & \text{se} \quad z \neq 0 \\ 0 \quad \text{se} \quad z = 0. \end{cases} \end{aligned}$$

Per dimostrare questa importante relazione occorrono dei teoremi preliminari di cui si darà solo una parziale dimostrazione. Maggiori dettagli possono essere trovati in [4].

Teorema 4.2.4. *Sia L un reticolo; allora $\wp(z, L)$ soddisfa la seguente equazione differenziale:*

$$\wp'(z) = 4\wp(z)^3 - g_2(L)\wp(z) - g_3(L),$$

dove $g_2 = 60\ G_4(L)$ e $g_3 = 140\ G_6(L)$ con G_4, G_6 definiti come in 4.2.

Perciò definendo $x = \wp(z)$ e $y = \wp'(z)$ si ha che $\wp(z)$ verifica la seguente equazione differenziale:

$$y^2 = 4x^3 - g_2(L)x - g_3(L).$$

Quindi si può costruire la mappa:

$$\begin{aligned} \Phi : \mathbb{C}/L &\to \mathbb{P}^2(\mathbb{C}) \\ z \neq 0 &\longmapsto [\wp(z) : \wp'(z) : 1] \\ 0 &\longmapsto [0 : 1 : 0]. \end{aligned}$$

Si osserva quindi che:

$$\operatorname{Im}\Phi = \Big\{[x:y:1] \mid y^2 = 4x^3 - g_2(L)x - g_3(L)\Big\} \cup \{[0:1:0]\},$$

ed essa è una curva ellittica se e solo se è abbastanza liscia, cioè $\Delta = g_2^3 - 27g_3^3 \neq 0$.

Poiché l'equazione differenziale è soddisfatta per il Teorema 4.2 , occorre dimostrare solo che il discriminante è diverso da zero. Per fare ciò enunciamo il seguente lemma:

Lemma 4.2.1. *Un punto $z \in L$ è uno zero di $\wp'(z) \iff 2z \in L$.*

teorema 2.5.4. Sia $L = [\omega_1, \omega_2]$; ponendo $r_1 = \frac{\omega_1}{2}$, $r_2 = \frac{\omega_2}{2}$ e $r_3 = \frac{\omega_1+\omega_2}{2}$ questi saranno gli unici tre punti tali che $r_i \notin L$ e $2r_i \in L$ $\forall i$. Perciò dal lemma precedente si ha che $\wp(r_i)$ sono i tre zeri della cubica $y^2 = 4x^3 - g_2x - g_3$. Si vuole verificare che essi sono tutti distinti, infatti

$$\Delta \neq 0 \iff \text{i } \wp(r_i) \text{ sono tutti distinti,}$$

dove il discriminante è calcolato nel modo che segue:

$$\Delta(L) = \frac{1}{16}\prod_{i\leq j}(\wp(r_i) - \wp(r_j))^2.$$

Quindi si osserva che se $h_i(z) = \wp(z) - \wp(r_i)$ allora $h_i(z)$ è una funzione ellittica si ordine 2 poiché ha gli stessi poli di $\wp$, cioè essa ha due zeri.
Però r_i è uno zero almeno doppio poiché $h_i'(r_i) = \wp'(r_i) = 0$ da cui si può concludere che $\wp(r_i) \neq \wp(r_j) \quad \forall i \neq j$, da cui la tesi. □

Teorema 4.2.5. *Sia $L \subseteq \mathbb{C}$ un reticolo e $E_L : y^2 = 4x^3 - g_2x - g_3$ la curva ellittica corrispondente. Allora la mappa $\Phi : \mathbb{C}/L \longrightarrow E/\mathbb{C}$ è un isomorfismo di gruppi additivi.*

Le dimostrazioni di questo teorema e di quelli che seguono sono omesse per non appesantire la trattazione eccessivamente ma possono essere facilmente reperite in [2]. Per proseguire definiamo l'equivalenza omotetica tra due reticoli.

Definizione 4.2.6. *Due reticoli L_1 e L_2 su $\mathbb{C}$ si dicono omoteticamente equivalenti se esiste $\alpha \in \mathbb{C}$ tale che $\alpha L_1 = L_2$.*

Una proprietà interessante che riguarda i reticoli omoteticamente equivalenti è quella fornita dalla seguente proposizione che tornerà particolarmente utile nel seguito.

Proposizione 4.2.1. *Siano L_1 e L_2 due reticoli in $\mathbb{C}$ omoteticamente equivalenti. Allora la moltiplicazione per scalare per α induce un omomorfismo ben definito:*

$$\begin{aligned}\Phi_\alpha : \mathbb{C}/L_1 &\to \mathbb{C}/L_2 \\ z &\longmapsto \alpha z \pmod{L_2},\end{aligned}$$

e queste sono le uniche mappe olomorfe da $\mathbb{C}/L_1$ a $\mathbb{C}/L_2$ che preservano l'identità.

Teorema 4.2.6. *Siano $E_1/\mathbb{C}$ e $E_2/\mathbb{C}$ due curve ellittiche che corrispondono a due reticoli L_1 e L_2; allora sono isomorfe se e solo se L_1 e L_2 sono omoteticamente equivalenti.*

Inoltre dalla proposizione 2.1.1 è noto che due curve ellittiche sono isomorfe se e solo se hanno lo stesso j invariante. Definiamo il j invariante del reticolo L è dato da:

$$j(L) = 1728\frac{g_2^3}{g_2^3 - 27g_3^2}.$$

Osservando che la curva ellittica associata al reticolo L è definita come:

$$y^2 = 4x^3 - g_2x - g_3,$$

questa è isomorfa ad una curva di equazione

$$y^2 = x^3 + Ax + B \quad \text{con} \quad A = -\frac{1}{4}g_2 \quad \text{e} \quad B = -\frac{1}{4}g_3,$$

quindi $j(L) = j(E_L)$.

Il teorema che segue sarà utile per il corollario che dà un risultato molto importante, ossia quello che mette in relazione le curve ellittiche con i reticoli su $\mathbb{C}$ che permette di lavorare in uno spazio più semplice.

Teorema 4.2.7. *Per ogni $\beta \in \mathbb{C}$ esiste un reticolo L tale che $j(L) = \beta$.*

Corollario 4.2.1 (Teorema di uniformizzazione)**.** *Per ogni $E/\mathbb{C}$ curva ellittica, esiste un reticolo $L \subseteq \mathbb{C}$ tale che $E_L \simeq E$.*

Dimostrazione. Data una curva ellittica E si può calcolare il suo j invariante $j(E)$. Dal Teorema 4.2.7 è noto che esiste un reticolo $L \subseteq \mathbb{C}$ tale che $j(L) = j(E)$. Ma $j(L) = j(E_L)$ dove E_L è la curva ellittica associata al reticolo. Perciò si avrà che $j(E) = j(E_L)$ per cui da 2.1.1 E ed E_L sono isomorfe. □

Questo corollario ha delle importanti conseguenze sulle proprietà delle moltiplicazioni per un intero; infatti, data $E/\mathbb{C}$ una curva ellittica e $m \geq 1$ un intero, allora si ha

- Esiste un isomorfismo di gruppi astratti

$$E[m] \simeq \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z},$$

- La moltiplicazione per m, cioè $[m] : E \longrightarrow E$ ha grado m^2.

Capitolo 5

Scambio di chiavi sulle curve ellittiche

5.1 Protocollo di scambio chiavi di Diffie-Hellman

La crittografia può essere suddivisa in due macro sezioni: la crittografia a chiave privata e quella a chiave pubblica. Nella prima le operazioni di criptatura e decriptatura avvengono tramite l'utilizzo di una chiave condivisa tra due utenti, mente nella seconda tali operazioni avvengono mediante l'utilizzo di due chiavi, una pubblica ed una privata, per ogni singolo utente. La crittografia a chiave privata è meno sicura e difficile da realizzare ma al tempo stesso permette di svolgere tali operazioni molto più velocemente, per questo motivo spesso si utilizza la crittografia a chiave pubblica per lo scambio della chiave condivisa utile alla crittografia a chiave privata. In questo modo di procedere avviene, quindi, uno scambio di chiavi. Uno tra i primi protocolli ideati in tal senso è quello del 1976 attribuibile ai due crittografi statunitensi Whitfield Diffie e Martin Hellman, successivamente adattato all'utilizzo delle curve ellittiche.
In tale protocollo si procede nel modo seguente:

- Alice e Bob decidono quale curva ellittica utilizzare, definiranno quindi i parametri a e b della curva, un generatore G e un primo p così da lavorare sulla curva:

  $$y^2 = x^3 + ax + b \qquad (\text{mod } p)$$

  che avrà n punti, calcolabili come visto nel capitolo precedente.

- Alice e Bob vogliono ottenere una chiave condivisa per cui entrambi genereranno due numeri casuali p_A e p_B con $1 \leq p_A, p_B \leq n-1$ che costituiranno le chiavi private.

- Alice e Bob calcoleranno le chiavi pubbliche P_A e P_B come $[p_A]G$ e $[p_B]G$.

- Alice calcolerà la chiave condivisa come: $[p_A]P_B$ e Bob calcolerà $[p_B]P_A$ così che entrambi abbiano la stessa chiave condivisa senza che sia necessario lo scambio della chiave privata. Infatti, dal punto precedente Alice otterrà la chiave $[p_A][p_B]G = [p_A p_B]G$ e Bob la chiave $[p_B][p_A]G = [p_B p_A]G$.

Questo protocollo basa la sua sicurezza sulla presunta difficoltà nella risoluzione del logaritmo discreto.

5.2 Implementazione in Python

Procedendo si può vedere un esempio di implementazione di tale protocollo sulla curva:

$$y^2 = x^3 + 7x + 3 \qquad (\text{mod } 13)$$

che ha $n = 13$ punti il cui generatore è $G = (3,5)$. Le due chiavi private possono essere scelte dall'utente e in questo esempio sono state poste pari a $p_A = 5$ e $p_B = 3$.

Codice

```
from typing import Tuple, Optional

Point = Tuple[int,int]

#Parametri condivisi per la curva y^2=x^3++7x+3
p = 13
n = 13
G = [3,5]
a = 7
b = 3

#Queste due funzioni restituiscono la coordinata x e y di un punto di una curva (rispettivamente)
def x(P : Point) -> int:
    return P[0]

def y(P):
    return P[1]

#Questa funzione somma due punti di una curva ellittica, quindi restituisce P3 = P1 + P2 = (x3, y3)
#in cui il punto puo' essere indicato dalle coordinate x e y intere oppure da none se e' il punto all'infinito
def point_add(P1 : Optional[Point], P2 : Optional[Point]) -> Optional[Point]:
    if P1 is None:
        return P2
    if P2 is None:
        return P1
    if (x(P1) == x(P2)) and (y(P1) != y(P2)):
        return None
    if P1 == P2:
        lam = ((3 * x(P1)**2 + a) * pow(2 * y(P1), p - 2, p)) % p
    else:
        lam = ((y(P2) - y(P1)) * pow(x(P2) - x(P1), p - 2, p)) % p

    x3 = (lam**2 - x(P1) - x(P2)) % p
```

```
34      y3 = (lam * (x(P1) - x3) - y(P1)) % p
35
36      return x3, y3
37
38  #Questa funzione calcola il punto R = dP
39  def point_mul(d : int, P : Point) -> Point:
40      R = None
41
42      for i in range(256):
43          if (d >> i) & 1:
44              R = point_add(R,P)
45
46          P = point_add(P,P)
47
48      return R
49
50  #Questa funzione restituisce una chiave pubblica a partire da una chiave
        privata
51  def pubkey_gen(private_key : int) -> Point:
52      R = point_mul(private_key, G)
53
54      assert R is not None
55      return R
56
57  #Questa funzione serve per controllare che il punto trovato
58  #appartenga alla curva ellittica:
59  def isPoint(P):
60      sx=(y(P)**2) % p
61      dx=((x(P)**3)+a*x(P)+b) % p
62      if(sx==dx):
63          return True
64      else:
65          return False
66
67  #Adesso possiamo iniziare il protocollo di scambio di chiavi di Diffie-
        Hellman
68  print("Inserire la chiave privata di Alice")
69  private_key_a=int(input())
70  print("Inserire la chiave privata di Bob")
71  private_key_b=int(input())
72
73  pubblic_key_a=pubkey_gen(private_key_a)
74  print("La chiave pubblica di Alice e': %s" %(pubblic_key_a,))
75  print("Controllo che appartenga alla curva ellittica:")
76  print(isPoint(pubblic_key_a))
77
78  pubblic_key_b=pubkey_gen(private_key_b)
79  print("La chiave pubblica di Bob e': %s" %(pubblic_key_b,))
80  print("Controllo che appartenga alla curva ellittica:")
81  print(isPoint(pubblic_key_b))
82
83  shared_key_a=point_mul(private_key_a,pubblic_key_b)
84  shared_key_b=point_mul(private_key_b,pubblic_key_a)
```

```
85 print("La chiave condivisa calcolata da Alice e': %s" %(shared_key_a,))
86 print("La chiave condivisa calcolata da Bob e': %s" %(shared_key_b,))
87
88 print("Controllo che appartenga alla curva ellittica:")
89 print(isPoint(shared_key_a))
```

Listing 5.1. Implementazione del protocollo di scambio chiavi su curve ellittiche di Diffie-Hellman

Risultati

```
1  Inserire la chiave privata di Alice
2  5
3  Inserire la chiave privata di Bob
4  3
5  La chiave pubblica di Alice e': (6, 1)
6  Controllo che appartenga alla curva ellittica:
7  True
8  La chiave pubblica di Bob e': (2, 5)
9  Controllo che appartenga alla curva ellittica:
10 True
11 La chiave condivisa calcolata da Alice e': (4, 2)
12 La chiave condivisa calcolata da Bob e': (4, 2)
13 Controllo che appartenga alla curva ellittica:
14 True
```

Listing 5.2. Risultati

Parte II

Crittografia con grafi di isogenie

Capitolo 6

Grafi di isogenie

Fino ad ora abbiamo analizzato la crittografia su curve ellittiche, tuttavia lo scopo principale della tesi è quello di descrivere i protocolli crittografici post quantum basati sui grafi di isogenie tra curve ellittiche. In realtà l'aspetto interessante è quello di utilizzare non un'unica isogenia bensì diversi grafi di isogenie di dato grado che rendano sicuro il protocollo stesso. Quindi lo scopo di questa seconda parte della trattazione sarà quello di dare una rapida panoramica sui grafi e su alcune loro proprietà, analizzare in che modo si possono utilizzare le isogenie per generare i grafi e, infine, analizzare alcuni dei protocolli tutt'ora in studio per entrare nello standard del NIST, dandone una semplice implementazione in Python.

6.1 Panoramica sui grafi

I grafi sono, un oggetto matematico particolarmente utilizzato grazie alla grande versatilità che ne fa un ottimo strumento in un gran numero di applicazioni.
La panoramica sui concetti relativi ai grafi riguarderà i cosiddetti grafi indiretti, poiché sono quelli che ci interesseranno dal punto di vista crittografico. Un grafo è un insieme di elementi, detti nodi, collegati tra loro da archi. Una definizione più formale è la seguente:

Definizione 6.1.1. *Si dice grafo una coppia ordinata $G = (V, E)$ di insiemi, con V insieme dei nodi ed E insieme degli archi, tali che gli elementi di E siano coppie di elementi di V. Tale grafo si dice indiretto se le coppie di elementi di V che formano l'insieme E non sono coppie ordinate.*

Un esempio di grafo è considerato nella Figura 6.1 in cui si possono osservare delle differenze tra il grafo a destra e quello a sinistra che saranno più chiare a breve.
Dato un nodo v del grafo i suoi vicini saranno i nodi $\{v', v'', \dots\}$ collegati ad esso da un arco: nell'esempio della Figura 6.1 il nodo **3** avrà come vicini i nodi **2,4,5**.Un cammino tra due nodi n ed n' è una sequenza di archi che collega i due nodi. Nell'esempio a destra della Figura 6.1 un cammino è dato da **1,5,3,4,6**. La distanza tra due nodi è il cammino più corto che li collega. Un grafo può essere connesso o disconnesso; in particolare, si dice

connesso se ogni coppia di nodi è collegata almeno da un cammino. Il grado di un nodo è il numero di archi che lo collegano ad altri nodi, e un grafo si dice $k-regolare$ se tutti i nodi hanno lo stesso grado pari a k.

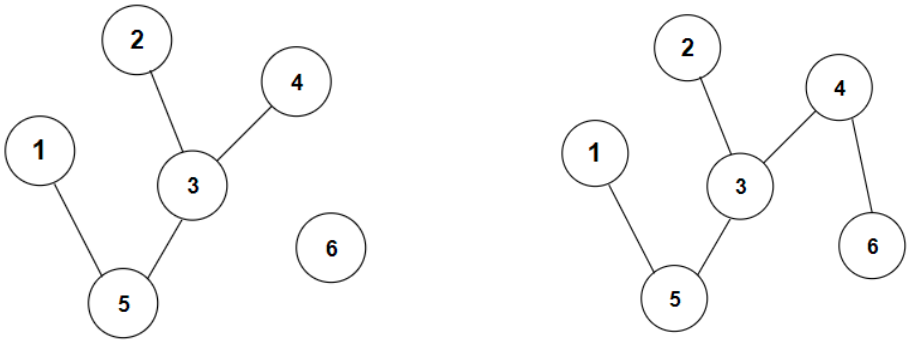

Figura 6.1. Esempio di grafo indiretto rispettivamente non connesso e connesso con 6 nodi e rispettivamente 4 o 5 archi.

La matrice di adiacenza di un grafo con n nodi è una matrice di dimensioni $n \times n$ che ha come entrata (i, j) rispettivamente 1 se c'è un arco tra i nodi i e j e 0 altrimenti. Questa matrice, sempre simmetrica poiché consideriamo grafi indiretti, avrà n autovalori reali, che quindi possono essere ordinati. Denotiamo perciò con $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ gli autovalori di tale matrice. E' interessante osservare la seguente proposizione riguardante i grafi $k-regolari$, che sarà particolarmente utile nel caso di grafi di isogenie. Gli aspetti più utili al fine di definire tali grafi, reperibili in [1], sono presentati di seguito. Ulteriori dettagli possono essere trovati in [8].

Proposizione 6.1.1. *Se G è un grafo $k-regolare$, allora l'autovalore maggiore λ_1 e quello minore λ_n la relazione:*

$$k = \lambda_1 \geq \lambda_n \geq -k.$$

La dimostrazione, omessa per semplicità, può essere trovata in [8]. Hanno una grande importanza i grafi detti ϵ-espansori per le loro proprietà pseudo-randomiche molto utili in ambito computazionale e, soprattutto, crittografico.

Definizione 6.1.2. *Un grafo si dice grafo ϵ-espansore in un lato se è $k-regolare$ con $k \geq 1$ e, dato $\epsilon > 0$, si ha:*

$$\lambda_2 \leq (1-\epsilon)k;$$

invece è detto grafo ϵ-espansore in due lati se:

$$\lambda_n \geq -(1-\epsilon)k.$$

La definizione appena data non dà un'idea intuitiva su cosa effettivamente voglia dire, a livello topologico, avere un grafo di espansione. Per questo motivo introduciamo l'espansione degli archi:

Definizione 6.1.3. *Sia $F \subset V$ un sottoinsieme dei vertici di G. La frontiera di F, indicata con $\partial F \subset E$, è l'insieme di archi di G che collegano nodi in F con nodi in $V \setminus F$. Il rapporto di espansione degli archi del grafo G, denotato con $h(G)$, è la quantità:*

$$h(G) = \min_{\substack{F \subset V, \\ F \leq \#V/2}} \frac{\#\partial F}{\#F},$$

dove $h(G) = 0$ solo se G è sconnesso.

In pratica, a livello topologico, questo rapporto ci dice quanto il grafo è lontano dall'essere sconnesso. Le proprietà spettrali dei grafi di espansione e quella appena presentata sono in relazione grazie alla disuguaglianza discreta di *Cheeger*:

Teorema 6.1.1. *Sia G un grafo $k-regolare$ e $\epsilon-espanso$ in un lato; allora:*

$$\frac{\varepsilon}{2}k \leq h(G) \leq \sqrt{2\varepsilon}k.$$

Le importanti proprietà crittografiche di questo tipo di grafi sono evidenziate dal teorema di mixing che necessita di una proposizione preliminare per essere compreso.

Proposizione 6.1.2. *Sia G un grafo $k-regolare$ ed $\epsilon-espanso$ da un lato. Per ogni vertice v ed ogni raggio $r > 0$, definiamo $B(v, r)$ la palla che contiene i vertici a distanza massima r dal vertice v. Allora, esiste una costante $c > 0$, che dipende solo da k e ϵ, tale che:*

$$\#B(v, r) \geq \min\{(1 + c)^r, \#V\}.$$

Questa proposizione permette di osservare che il diametro del grafo con n nodi preso in considerazione avrà un limite dato da $O(\log n)$. Il teorema più interessante, come già anticipato, è il teorema di mixing che permette di osservare che, selezionando un cammino random, ossia un cammino aleatorio tra i nodi di un grafo in cui si fanno passi successivi procedendo in direzioni casuali, di lunghezza i, selezionando in modo casuale uniformemente distribuito i vertici del cammino, se il cammino ha lunghezza simile al diametro del grafo, esso potrebbe terminare in ogni vertice con la stessa probabilità.

Teorema 6.1.2. *Sia G un grafo $k-regolare$ $\epsilon-espanso$ in due lati e sia $F \subset V$ un sottoinsieme dei vertici di G. Sia inoltre v un vertice in V; allora un cammino random con lunghezza almeno*

$$\frac{\log \#F^{1/2}/2\#V}{\log(1 - \varepsilon)},$$

che parte da v terminerà in F con probabilità almeno $\#F/2\#V$.

6.2 Grafi di isogenie

Ricordiamo cue curve ellittiche E ed E' vengono dette isogene se esiste un'isogenia da E in E'. Ad ogni curva ellittica è associato, come visto in precedenza, un j-invariante. Si dice che due j-invarianti sono isogeni se le curve a cui sono associati lo sono. Questa breve premessa permette di dare la definizione centrale della trattazione:

Definizione 6.2.1 (Grafi di isogenie)**.** *Un grafo di isogenie è un grafo i cui nodi sono i j-invarianti di curve isogene e i cui archi sono le isogenie tra di esse.*

Tali grafi di isogenie sono grafi indiretti poichè, come visto in 3.4.1, ad ogni isogenia $\phi : E \to E'$ è associata un'isogenia duale $\hat{\phi} : E' \to E$ dello stesso grado.
Alcune proprietà fondamentali dei grafi di isogenie sono garantite dalla seguente proposizione:

Proposizione 6.2.1. *Sia $E : y^2 = x^3 + Ax + B$ una curva ellittica in forma di Weierstrass definita su un campo finito k di caratteristica p e sia $l \neq p$ un primo.*

- *Ci sono $l + 1$ isogenie di grado l con dominio E definite sulla chiusura di un campo algebrico $\bar{k}$.*
- *Ci sono 0,1,2 o $l + 1$ isogenie di grado l con dominio E definite su k.*
- *Se E è una curva ellittica ordinaria c'è un'unica isogenia separabile di grado p con dominio E, se invece è supersingolare non ce n'è nemmeno una.*
- *La mappa $(x, y) \to (x^p, y^p)$ è un'isogenia puramente inseparabile di grado p da E in $E^{(p)} : y^2 = x^3 + A^p x + B^p$.*

Come si può osservare anche dal secondo punto della proposizione precedente, ci sono grandi differenze tra le proprietà delle isogenie tra curve ellittiche ordinarie e tra curve ellittiche supersingolari. Per i fini crittografici sono utili quelle supersingolari su cui concentreremo l'attenzione d'ora in avanti.

Dopo aver presentato alcune proprietà dei grafi $\epsilon-$espansi ed aver definito i grafi di isogenie, ci aspettiamo che essi abbiano proprietà adeguate ad essere utili in un protocollo crittografico. Fortunatamente questo è vero ed è garantito dalle proposizioni che seguono, che necessitano di una definizione preliminare:

Definizione 6.2.2 (Grafo di Ramanujan)**.** *Sia $k \geq 1$ e sia G_i una sequenza di grafi $k-$regolari. Allora:*

$$\max\left(|\lambda_2|, |\lambda_n|\right) \geq 2\sqrt{k-1} - o(1),$$

per $n \to \infty$. Un grafo tale che $|\lambda_i| \leq 2\sqrt{k-1}$ per ogni λ_i eccetto λ_1 è chiamato grafo di Ramanujan.

La seguente proposizione sarà importante nel seguito e garantisce che i grafi di isogenie sono Ramanujan.

Proposizione 6.2.2. *Siano p, l due primi distinti, allora:*

- *Tutti i j-invarianti supersingolari di una curva in $\bar{\mathbb{F}}_p$ sono definiti in $\mathbb{F}_{p^2}$;*
- *Ci sono*

$$\left\lfloor \frac{p}{12} \right\rfloor + \begin{cases} 0 & \text{se } p - 1 \mod 12 \\ 1 & \text{se } p = 5,7 \mod 12 \\ 2 & \text{se } p = 11 \mod 12 \end{cases}$$

  classi di isomorfismi di curve ellittiche supersingolari su $\bar{\mathbb{F}}_p$;

- *Il grafo di curve supersingolari in $\bar{\mathbb{F}}_p$ con l isogenie è connesso, $l + 1 -$ regolare ed è Ramanujan.*

Capitolo 7

Protocolli crittografici

7.1 Supersingular isogeny Diffie-Hellman (SIDH)

Il protocollo crittografico che andiamo ad esaminare, proposto per la prima volta da [1] è lo scambio di chiavi di Diffie-Hellman tramite i grafi di isogenie tra curve ellittiche. Nonostante il protocollo risulti certamente meno efficiente rispetto a ECDH, esso è studiato per la sua resistenza al computer quantistico. Infatti l'algoritmo di Shor ha mostrato come, utilizzando un computer quantistico, si può risolvere il problema del logaritmo discreto in tempo polinomiale, rendendo di fatto inutilizzabili tutti i protocolli precedentemente architettati basati sulle curve ellittiche.
L'idea generale è quella di spostarsi su curve ellittiche tramite isogenie. L'idea di base è la seguente: Alice e Bob partono dalla stessa curva ellittica E_0 e si muovono tramite un cammino casuale segreto fino ad arrivare a due curve, E_A ed E_B. Successivamente si scambiano le due curve e compiono altri due cammini casuali, arrivando alla stessa curva E_S. Più in dettaglio si fissano due primi piccoli ℓ_A ed ℓ_B e si fissa un primo p della forma $p = \ell_A^{e_A}\ell_B^{e_B}$. Costruiamo allora due grafi di isogenie i cui nodi saranno per entrambi i $j-invarianti$ delle curve supersingolari definite su $\mathbb{F}_p$, i cui archi saranno per Alice le isogenie di grado ℓ_A e per Bob le isogenie di grado ℓ_B. Ricordiamo che un'isogenia separabile è unicamente determinata dal suo nucleo e in questo caso il grado dell'isogenia è uguale alla cardinalità del nucleo, (dal Teorema 3.3.2). Di conseguenza, un cammino di lunghezza e_A in un grafo di isogenie di grado ℓ_A corrisponde a un nucleo di cardinalità $\ell_A^{e_A}$, e tale nucleo è ciclico se e solo se il cammino non ha cicli. Si ha quindi che la scelta di Alice di un cammino segreto di lunghezza e_A corrisponderà a scegliere un sottogruppo ciclico $\langle A\rangle \subset E[\ell_A^{e_A}]$; allo stesso modo, Bob sceglierà un sottogruppo ciclico $\langle B\rangle \subset E[\ell_B^{e_B}]$. Perciò, se $\ell_A \neq \ell_B$, il sottogruppo $\langle A\rangle + \langle B\rangle = \langle A, B\rangle$ è un sottogruppo ciclico di ordine $\ell_A^{e_A}\ell_B^{e_B}$. L'obiettivo è definire nel dettaglio un protocollo in cui Alice e Bob, partendo da due sottogruppi segreti $\langle A\rangle$ e $\langle B\rangle$, si scambino abbastanza informazioni per calcolare entrambi la curva $E/\langle A, B\rangle$ senza rivelare i rispettivi segreti. Una volta fatto daremo un esempio di piccole dimensioni e forniremo un codice Python che lo implementa.
Nel protocollo che segue è conveniente lavorare solo con le curve supersingolari perché i $j-$invarianti di curve supersingolari in $\mathbb{F}$ di caratteristica p appartengono tutti all'insieme

$\mathbb{F}_{p^2}$. Inoltre è particolarmente utile considerare le curve ellittiche in forma di Montgomery perché si può facilmente calcolare la coordinata y dell'immagine di un punto tramite un'isogenia conoscendo soltanto la x del punto di partenza. Infatti supponendo che una isogenia mandi x in $f(x)$, allora si ha che y viene mandata in $cyf'(x)$ per un certo $c \in K$, il che semplifica particolarmente l'implementazione. Le formule di Vélu permettono di calcolare esplicitamente un'isogenia di dato nucleo che sappiamo esistere ed essere unica dal Teorema 3.3.2 (per maggiori dettagli si può consultare [10]). Per le isogenie di secondo grado si ha la mappa seguente:

$$\phi : E_a \to E_{a'}, \quad x \to \frac{x(\alpha x - 1)}{x - \alpha}, \tag{7.1}$$

dove a è il parametro della curva ellittica in forma di Montgomery tale che $y^2 = x^3 + ax^2 + x$, e α è una radice dell'equazione $x^2 + ax + 1 = 0$, ossia è un punto che annulla il denominatore della formula di duplicazione vista in (2.4). Il parametro a' dell'immagine $E_{a'}$ è data da:

$$a' = 2(1 - 2\alpha^2).$$

Ragionando in modo simile per le isogenie di terzo grado si può costruire l'isogenia:

$$\phi : E_a \to E_{a'}, \quad x \to \frac{x(\beta x - 1)^2}{(x - \beta)^2}, \tag{7.2}$$

in cui β è tale da annullare il denominatore della formula di triplicazione delle curve in forma di Montgomery presentata in (2.5). Anche in questo caso si può facilmente calcolare la curva ellittica di arrivo $E_{a'}$; infatti si ha:

$$a' = (a\beta - 6\beta^2 + 6)\beta,$$

in cui a, come in precedenza, è il parametro identificativo della curva ellittica di partenza E_a.
Tali isogenie permettono di spostarsi all'interno del grafo e passare da un $j-$invariante all'altro. Ricordiamo che una curva ellittica è univocamente determinata (a meno di isomorfismi) dal suo $j-$invariante.
Il teorema di Tate garantisce che due curve ellittiche E ed E' su $\mathbb{F}_q$ sono isogene se e soltanto se hanno lo stesso numero di punti. Nell'esempio considerato in seguito considereremo $\ell_A = 2, \ell_B = 3$ e il primo p della forma $p = 2^{e_A}3^{e_B} - 1$, con e_A ed e_B tali che $2^{e_A} \approx 3^{e_B}$. Nel caso specifico di primi p scelti in questo modo il Teorema 29 di [15] (per maggiori dettagli si può consultare il Lemma 4.8 di [14]) garantisce che, per un primo di questa forma si ha che, se E è supersingolare,

$$E(\mathbb{F}_{p^2}) \cong \mathbb{Z}_{p+1} \times \mathbb{Z}_{p+1},$$

e il gruppo dei punti della curva su $\mathbb{F}_{p^2}$ coincide esattamente con la $(p+1)-$torsione, cioè

$$E(\mathbb{F}_{p^2}) \cong \mathbb{Z}_{2^{e_A}3^{e_B}} \times \mathbb{Z}_{2^{e_A}3^{e_B}}.$$

7.2 Protocollo SIDH

Il protocollo di scambio di chiavi segue i seguenti passi:

- Alice e Bob scelgono e_A ed e_B e viene definito il parametro $p = 2^{e_A}3^{e_B} - 1$, con $2^{e_A} \approx 3^{e_B}$.
- Alice si muoverà tramite isogenie di grado 2^{e_A}, e calcolerà due generatori pubblici del sottogruppo della $2^{e_A}-$torsione cioè
$$\langle P_A, Q_A \rangle = E[2^{e_A}] \simeq \mathbb{Z}_{2^{e_A} \times 2^{e_A}}.$$
- Allo stesso modo Bob si muoverà tramite isogenie di grado 3^{e_B}, e calcolerà due generatori pubblici del sottogruppo di 3^{e_B} cioè
$$\langle P_B, Q_B \rangle = E[3^{e_B}] \simeq \mathbb{Z}_{3^{e_B} \times 3^{e_B}}.$$
- Per calcolare la chiave pubblica, Alice sceglie $k_A \in \{0,1,\ldots,2^{e_A}-1\}$ con k_A coprimo con 2 e calcola il punto
$$S_A = P_A + [k_A]Q_A \tag{7.3}$$
e l'isogenia segreta $\phi_A : E \to E_A$ con $E_A = E/\langle S_A \rangle$. Tale isogenia ha grado e_A, e può essere calcolata componendo e_A isogenie di grado 2, cioè avanzando di e_A passi nel grafo delle isogenie di grado 2.
A questo punto, la chiave pubblica di Alice sarà data da
$$P_{k_A} = (E_A, P'_B, Q'_B) = (\phi_A(E), \phi_A(P_B), \phi_A(Q_B)). \tag{7.4}$$
- Allo stesso modo Bob per calcolare la chiave pubblica, sceglie $k_B \in \{0,1,\ldots,3^{e_B}-1\}$ e k_B coprimo con 3 e calcola il punto
$$S_B = P_B + [k_B]Q_B \tag{7.5}$$
e l'isogenia segreta $\phi_B : E \to E_B$ con $E_B = E/\langle S_B \rangle$. Tale isogenia ha grado e_B, e può essere calcolata componendo e_B isogenie di grado 3, cioè avanzando di e_B passi nel grafo delle isogenie di grado 3.
A questo punto, la chiave pubblica di Bob sarà data da
$$P_{k_B} = (E_B, P'_A, Q'_A) = (\phi_B(E), \phi_B(P_A), \phi_B(Q_A)). \tag{7.6}$$
- A questo punto Alice e Bob si scambiano le chiavi pubbliche P_{k_A} e P_{k_B} che hanno appena calcolato e procedono.
- Alice calcola il nuovo sottogruppo segreto sfruttando la sua chiave privata e le informazioni aggiuntive ottenute da Bob:
$$S'_A = P'_A + [k_A]Q'_A = \phi_B(S_A),$$
la nuova isogenia associata a questo sottogruppo, cioè $\phi'_A : E_B \to E_{AB} = E_B/\langle S'_A \rangle$ e il segreto condiviso, cioè il rispettivo $j-$invariante pari a $j(E_{AB})$.

- Bob calcola il nuovo sottogruppo segreto sfruttando la sua chiave privata e le informazioni aggiuntive ottenute da Alice:

$$S'_B = P'_B + [k_B]Q'_B = \phi_A(S_B),$$

la nuova isogenia associata a questo sottogruppo, cioè $\phi'_B : E_A \to E_{BA} = E_A/\langle S'_A \rangle$ e il segreto condiviso, cioè il rispettivo $j-$invariante pari a $j(E_{BA})$.

Notiamo che il protocollo permette ad Alice e Bob di arrivare alla stessa curva ellittica. Infatti dalla costruzione si vede che

$$E_{AB} = E_B/\langle S'_A \rangle = E_B/\langle \phi_B(S_A) \rangle;$$

ma $E_B = E/\langle S_B \rangle$ e ϕ_B è la corrispondente proiezione, quindi $E_{AB} = E/\langle S_A, S_B \rangle$. Allo stesso modo per costruzione si vede che

$$E_{BA} = E_A/\langle S'_B \rangle = E_A/\langle \phi_A(S_B) \rangle;$$

ma $E_A = E/\langle S_A \rangle = \phi_A(E)$ da cui $E_{BA} = E/\langle S_A, S_B \rangle = E_{AB}$.

7.3 Esempio di piccole dimensioni

In questo esempio, preso dall'articolo [9], consideriamo le seguenti assegnazioni dei parametri: $e_A = 4$ ed $e_B = 3$. Quindi i cammini di Alice saranno sequenze di 4 isogenie di secondo grado, invece quelli di Bob sanno sequenze di 3 isogenie di terzo grado. Dall'osservazione del precedente sulla scelta di p si ha

$$p = 2^{e_A}3^{e_B} - 1 \Rightarrow p = 2^4 3^3 - 1 = 431.$$

Il grafo in questione avrà 37 $j-$invarianti in cui ciascuno avrà 3 archi nel caso di Alice e 4 archi nel caso di Bob. I parametri pubblici del protocollo saranno i seguenti:

- La curva ellittica di partenza è $E_{a_0} : y^2 = x^3 + a_0x^2 + x$ con $a_0 = 329i + 423$, il cui j-invariante è $j = 87i + 190$.
- I generatori di Alice della 2^4-torsione sono $P_A := (100i + 248, 304i + 199)$ e $Q_A := (426i + 394, 51i + 79)$.
- I generatori di Bob della 3^3-torsione sono $P_B := (358i + 275, 410i + 104)$ e $Q_B := (20i + 185, 281i + 239)$.

Una volta definiti questi parametri pubblici inizia il protocollo vero e proprio, per cui Alice e Bob, dopo aver scelto le rispettive chiavi private $k_A = 11$ e $k_B = 2$, calcoleranno S_A ed S_B come visto rispettivamente in (7.3) e (7.5):

$$S_A = (100i + 248, 304i + 199) + [11](426i + 394, 51i + 79) = (271i + 79, 153i + 430);$$

$$S_B = (358i + 275, 410i + 104) + [2](20i + 185, 281i + 239) = (122i + 309, 291i + 374).$$

Quindi inizieranno a calcolare le isogenie per spostarsi sul grafo nel modo che segue.

- Per implementare l'algoritmo Alice deve calcolare l'isogenia associata al sottogruppo generato da S_A, che essendo un'isogenia di grado 2^4, può essere calcolata come composizione di 4 isogenie di grado 2. Per calcolare tali igogenie si procede come segue: Alice calcolerà $[2^3]S_A$ che ha ordine 2 e tramite le formule di Vèlu calcolerà l'isogenia corrispondente. Successivamente, per calcolare la seconda isogenia, Alice aggiornerà il valore di S_A e calcolerà $[2^2]S_A$, che ha ancora ordine 2, ricavando l'isogenia come prima dalle formule di Vèlu. Procederà allo stesso modo per le altre due isogenie. Quindi Alice si sposterà nel modo che segue:

$$\begin{array}{llll} \phi_0 : E_{a_0} \to E_{a_1}, & \text{con} & a_1 = 289i + 341 & \text{e } j\left(E_{a_1}\right) = 107; \\ \phi_1 : E_{a_1} \to E_{a_2}, & \text{con} & a_2 = 273i + 76 & \text{e } j\left(E_{a_2}\right) = 344i + 190; \\ \phi_2 : E_{a_2} \to E_{a_3}, & \text{con} & a_3 = 93i + 65 & \text{e } j\left(E_{a_3}\right) = 350i + 65; \\ \phi_3 : E_{a_3} \to E_{a_4}, & \text{con} & a_4 = 423i + 179 & \text{e } j\left(E_{a_4}\right) = 222i + 118, \end{array}$$

  aggiornando ad ogni spostamento i punti P_B e Q_B tramite l'isogenia calcolata. L'ultimo aggiornamento avrà mandato i punti di partenza in $Q'_B = (142i+183, 119i+360)$ e $Q'_B = (220i + 314, 219i + 10)$.

- Bob, seguendo il medesimo ragionamento, per calcolare le sue 3 isogenie procederà come segue. Calcolerà $[3^2]S_B$ che ha ordine 3 e tramite le formule di Vèlu calcolerà l'isogenia corrispondente. Successivamente aggiornerà il valore di S_B e calcolerà $[3^1]S_B$, che ha nuovamente ordine 3, ricavando l'isogenia come prima dalle formule di Vèlu. Procederà allo stesso modo per l'ultima isogenia. Quindi Bob si sposterà nel modo che segue:

$$\begin{array}{llll} \phi_0 : E_{a_0} \to E_{a_1}, & \text{con} & a_1 = 134i + 2i + 341 & \text{e } j\left(E_{a_1}\right) = 106i + 379; \\ \phi_1 : E_{a_1} \to E_{a_2}, & \text{con} & a_2 = 117i + 54 & \text{e } j\left(E_{a_2}\right) = 325i + 379; \\ \phi_2 : E_{a_2} \to E_{a_3}, & \text{con} & a_3 = 273i + 76 & \text{e } j\left(E_{a_3}\right) = 273i + 76; \end{array}$$

  aggiornando ad ogni spostamento i punti P_A e Q_A tramite l'isogenia calcolata. L'ultimo aggiornamento avrà mandato i punti di partenza in $P'_A = (187i+226, 43i+360)$ e $Q'_A = (325i + 415, 322i + 254)$.

A questo punto Alice e Bob si scambieranno la chiave pubblica P_{k_A} e P_{k_B} come visto rispettivamente in (7.4) e (7.6):

$$P_{k_A} = ((423i + 190), (142i + 183, 119i + 360), (220i + 314, 219i + 10));$$

$$P_{k_B} = ((273i + 76), (187i + 226, 43i + 360), (325i + 415, 322i + 254)).$$

Così, dopo essersi scambiati le chiavi pubbliche, Alice e Bob possono proseguire il loro cammino scambiandosi di posto, cioè Alice proseguirà dalla curva di arrivo di Bob e Bob da quella di Alice. Entrambi, utilizzando le informazioni fornite e la chiave privata calcoleranno i nuovi S_A ed S_B:

$$S_A = (187i + 226, 43i + 360) + [11](325i + 415, 322i + 254) = (125i + 357, 415i + 249);$$

$$S_B = (142i + 183, 119i + 360) + [2](220i + 314, 219i + 10) = (339i + 124{,}187i + 380).$$

Quindi potranno calcolare i nuovi cammini che porteranno entrambi nella curva con j-invariante pari a 234, infatti:

- Alice si sposterà tramite 4 isogenie nel modo che segue:

$$\begin{array}{llll} \phi_0 : E_{a_0} \to E_{a_1}, & \text{con} & a_1 = 289i + 341 & \text{e } j\left(E_{a_1}\right) = 364i + 304; \\ \phi_1 : E_{a_1} \to E_{a_2}, & \text{con} & a_2 = 414i + 428 & \text{e } j\left(E_{a_2}\right) = 67; \\ \phi_2 : E_{a_2} \to E_{a_3}, & \text{con} & a_3 = 246i & \text{e } j\left(E_{a_3}\right) = 242; \\ \phi_3 : E_{a_3} \to E_{a_4}, & \text{con} & a_4 = 230 & \text{e } j\left(E_{a_4}\right) = 234 \end{array}$$

- Bob, invece, si sposterà tramite 3 isogenie:

$$\begin{array}{llll} \phi_0 : E_{a_0} \to E_{a_1}, & \text{con} & a_1 = 183i + 177 & \text{e } j\left(E_{a_1}\right) = 299i + 315; \\ \phi_1 : E_{a_1} \to E_{a_2}, & \text{con} & a_2 = 31 & \text{e } j\left(E_{a_2}\right) = 61; \\ \phi_2 : E_{a_2} \to E_{a_3}, & \text{con} & a_3 = 230 & \text{e } j\left(E_{a_3}\right) = 234. \end{array}$$

Quindi, una volta terminato il protocollo, entrambi giungeranno alla curva ellittica condivisa (indicata dal j-invariante a meno di isomorfismi), ossia alla curva:

$$E : y^2 = x^3 + 230x^2 + x.$$

Per concludere l'esempio ed avere un'idea più visiva di cosa sia accaduto si possono vedere nella Figura 7.1 in verde i due j-invarianti delle curve ellittiche di partenza, in giallo il percorso compiuto da Alice e in blu quello compiuto da Bob. Osservando le frecce si può, inoltre, notare il momento nel quale si scambiano la chiave pubblica.

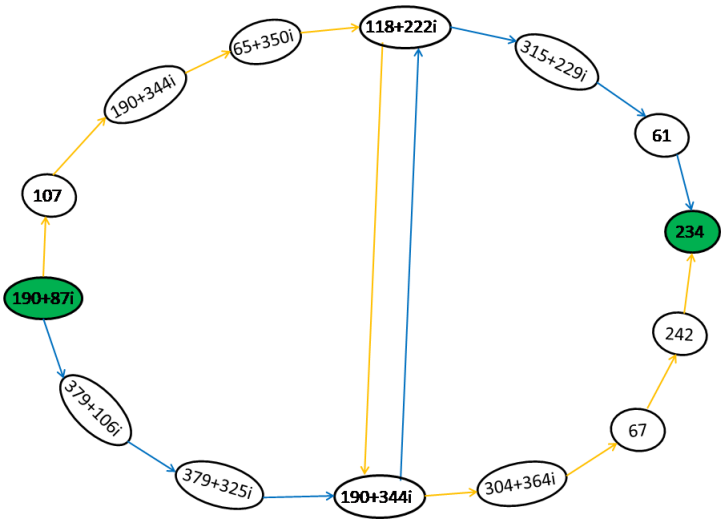

Figura 7.1. Grafo relativo all'esempio numerico del SIDH

7.4 Implementazione Python

Il codice in linguaggio Python che segue implementa l'esempio appena visto. Tuttavia è sufficiente modificare alcuni parametri, come le chiavi private di Alice e Bob, la curva di partenza, i punti P_A, P_B, Q_A, Q_B e il parametro p per implementari il protocollo SIDH anche in altri esempi.
La prima parte del codice fornisce tutte le funzioni necessarie nel seguito per eseguire tutti gli spostamenti sui nodi del grafo in questione. Successivamente, dalla riga 259 è implementato il protocollo vero e proprio.

```
1 #Prima di tutto definiamo il modulo di numeri comlessi:
2 from typing import Tuple, Optional
3 import networkx as nx
4 import matplotlib.pyplot as plt
5
6 Point = Tuple[complex,complex]
```

Listing 7.1. SIDH

Dopo aver definito la variabile di tipo Point, occorre definire il modulo di numeri complessi e le operazioni, quelle basilari come la moltiplicazione e la potenza di numeri complessi modulo p e quelle più complesse ma molto utili come l'inversa e la radice di un numero complesso modulo p.

```
1  def mod(x:complex,p:int):
2      rep=(x.real)%p
3      imp=(x.imag)%p
4      return (complex(rep,imp))
5
6  #adesso vogliamo definire la moltiplicazione modulo p tra numeri
       complessi
7
8  def mol(x1:complex,x2:complex,p:int):
9      return(mod((x1*x2),p))
10
11 #adesso si vuole definire la potenza, per farlo
12 #occorre utilizzare un algoritmo adatto:
13
14 def pot(x:complex,d:int,p:int):
15     R=1
16     for i in range(256):
17         if (d >> i) &1:
18             R = mol(x,R,p)
19         x = mol(x,x,p)
20     return (R)
21
22 #Serve anche definire il contrario di un numero complesso
23 # a+bi cioe' a-bi
24
25 def contr(x:complex,p):
26     re=x.real
27     im=-x.imag
28     return mod((complex(re,im)),p)
```

```
29
30 #adesso per semplicita' definiamo l'inversa
31
32 def inv(x,p):
33     contrario=contr(x,p)
34     potenza=pot(((x.real)**2+(x.imag)**2),p-2,p)
35     return (mol(contrario,potenza,p))
```

Listing 7.2. SIDH

Successivamente ci sono le funzioni relative alle curve ellittiche, ossia quelle che permettono di calcolare la somma tra due punti P_1 e P_2 della curva in forma di Montgomery, e quelle che riguardano la duplicazione e la moltiplicazione per n di un punto della curva.

```
1  #implementazione della duplicazione di un punto
2
3  def dup(x,p,a):
4      x2=mol(x,x,p)
5      x21=mod((x2-1),p)
6      num=mol(x21,x21,p)
7      ax=mol(x,a,p)
8      xper4=mol(x,4,p)
9      den1=mod((x2+ax),p)
10     den2=mod((den1+1),p)
11     den3=mol(den2,xper4,p)
12     den4=inv(den3,p)
13     return(mol(num,den4,p))
14
15 #Queste due funzioni restituiscono la coordinata x e y di un punto di
       una curva (rispettivamente)
16 def x(P : Point) -> int:
17     return P[0]
18
19 def y(P):
20     return P[1]
21
22 #Questa funzione somma due punti di una curva ellittica, cioe' ci
       restituisce P3 = P1 + P2 = (x3, y3)
23 #Optional[x] significa che l'elemento x  a cui ci si sta riferendo e' di
        tipo Point oppure None
24
25 def point_add(P1 : Optional[Point], P2 : Optional[Point],p) -> Optional[
       Point]:
26     if P1 is None:
27         return P2
28     if P2 is None:
29         return P1
30     if (x(P1) == x(P2)) and (y(P1) != y(P2)):
31         return None
32     if P1 == P2:
33         lam = mod(((3 * x(P1)**2 + 2*a*x(P1)+1) * inv(2 * y(P1),p)),p)
34     else:
35         lam =mod( ((y(P2) - y(P1)) * inv(x(P2) - x(P1), p)),p)
36
37     x3 = mod((lam**2 - x(P1) - x(P2)-a),p)
```

```
38      y3 = mod((lam * (x(P1) - x3) - y(P1)),p)
39
40      return x3, y3
41
42  #Questa funzione calcola il punto R = dP
43  def point_mul(d : int, P : Point,p) -> Point:
44      R = None
45
46      for i in range(256):
47          if (d >> i) & 1:
48              R = point_add(R,P,p)
49
50          P = point_add(P,P,p)
51
52      return R
53
54  #Questa funzione restituisce una chiave pubblica a partire da una chiave
        privata
55  def pubkey_gen(private_key : int,G) -> Point:
56      R = point_mul(private_key, G)
57      assert R is not None
58      return R
59
60  #questa funzione permette di assegnare una variabile di tipo Point
61
62  def ass(x:complex,y:complex)->Point:
63      return x,y
64
65  # questa funzione permette di ottenere il nuovo a di una curva data un
        isogenia
66  # di grado 2
```

Listing 7.3. SIDH

Seguono, poi, le funzioni che permettono di calcolare il parametro c tale che, dato un punto P e una funzione f, la coordinata x viene mandata in $x_{prime} = f(x)$ ed è possibile calcolare la coordinata y_{prime} come $ycf'(x)$ ne caso di isogenie di grado 2 e di grado 3. Infine, terminano la parte di codice dedicata alla definizione delle funzioni utili le funzioni che permettono, dato un punto P di calcolare la sua immagine tramite l'isogenia ϕ anche in questo caso sia per isogenie di grado 2 che di grado 3.

```
1   def new_a(x:complex,p)->complex:
2       a_prime=mol(mod(1-mol(pot(x,2,p),2,p),p),2,p)
3       return a_prime
4
5   # questa funzione permette di ottenere il nuovo a di una curva data un
        isogenia
6   # di grado 3
7
8   def new_aB(x:complex,p,a)->complex:
9       ax=mol(a,x,p)
10      x2=pot(x,2,p)
11      x26=mol(6,x2,p)
12      a_prime=mol(mod(ax-x26+6,p),x,p)
```

```
13      return a_prime
14
15  # Questa funzione restituisce il j-invariante di una curva
16
17  def j_inv(a:complex,p)->complex:
18      A2=pot(a,2,p)
19      A32=mod(A2-3,p)
20      A323=pot(A32,3,p)
21      num=mol(256,A323,p)
22      den=inv(mod(A2-4,p),p)
23      return mol(num,den,p)
24
25  # Questa funzione calocla la radice quadrata di un numero complesso
        modulo p
26
27  def root(x,p,q):
28      a1=pot(x,107,p)
29      alpha=mol(a1,mol(a1,x,p),p)
30      a0=pot(alpha,q+1,p)
31      if a0==-1:
32          return None
33      x0=mol(a1,x,p)
34      if alpha==-1:
35          x=complex(0,x0)
36      else:
37          b=pot(mod(1+alpha,p),215,p)
38          x=mol(b,x0,p)
39      return x
40
41  # Questa funzione calcola il parametro c dell'isogenia di grado 2
42
43  def calcolo_c_A(P:Point,p,a,xr):
44      x=P[0]
45      l=mol(x,xr,p)
46      l=mod(l-1,p)
47      l=mol(l,x,p)
48      den=inv(mod(x-xr,p),p)
49      x_new=mol(l,den,p)
50      y=P[1]
51      x3_new=pot(x_new,3,p)
52      x2_new=pot(x_new,2,p)
53      ax2_new=mol(a,x2_new,p)
54      arg=mod((x3_new+ax2_new+x_new),p)
55      rad1=root(arg,p,p**2)
56      rad2=mol(-1,rad1,p)
57      x2=pot(x,2,p)
58      xrx=mol(xr,x,p)
59      xrx2=mol(xrx,2,p)
60      fprimo=mol(mol(xr,x2-xrx2+1,p),inv(mol(x-xr,x-xr,p),p),p)
61      c1=mol(rad1,inv(mol(y,fprimo,p),p),p)
62      c2=mol(rad2,inv(mol(y,fprimo,p),p),p)
63      minimum=min(abs(c1),abs(c2))
64      if minimum==abs(c1):
```

```
65          return c1
66      else:
67          return c2
68
69  # Questa funzione permette di calcolare dove viene mandato un punto
70  # tramite un isogenia di grado 2
71
72  def PhiA_i(P:Point,p,a,xr,c):
73      x=P[0]
74      l=mol(x,xr,p)
75      l=mod(l-1,p)
76      l=mol(l,x,p)
77      den=inv(mod(x-xr,p),p)
78      x_new=mol(l,den,p)
79      y=P[1]
80      x2=pot(x,2,p)
81      xrx=mol(xr,x,p)
82      xrx2=mol(xrx,2,p)
83      fprimo=mol(mol(xr,x2-xrx2+1,p),inv(mol(x-xr,x-xr,p),p),p)
84      y_prime=mol(c,mol(y,fprimo,p),p)
85      P_prime=ass(x_new,y_prime)
86      print('le vecchie coordinate erano', P, 'invece le nuove sono',
        P_prime)
87      return P_prime
88
89  # Questa funzione calcola il parametro c dell'isogenia di grado 2
90
91  def calcolo_c_B(P:Point,p,a,xr):
92      x=P[0]
93      l=mol(x,xr,p)
94      l=mod(l-1,p)
95      l=mol(l,l,p)
96      l=mol(l,x,p)
97      den=inv(pot(mod(x-xr,p),2,p),p)
98      x_new=mol(l,den,p)
99      y=P[1]
100     x3_new=pot(x_new,3,p)
101     x2_new=pot(x_new,2,p)
102     ax2_new=mol(a,x2_new,p)
103     arg=mod((x3_new+ax2_new+x_new),p)
104     rad1=root(arg,p,p**2)
105     rad2=mol(-1,rad1,p)
106     x2=pot(x,2,p)
107     xrx2=mol(xr,x2,p)
108     xrx=mol(xr,x,p)
109     xr2=pot(xr,2,p)
110     xr23=mol(xr2,3,p)
111     num=mol((xrx-1),(xrx2+mol(1-xr23,x,p)+xr),p)
112     den=inv(pot(x-xr,3,p),p)
113     fprimo=mol(num,den,p)
114     c1=mol(rad1,inv(mol(y,fprimo,p),p),p)
115     c2=mol(rad2,inv(mol(y,fprimo,p),p),p)
116     minimum=min(abs(c1),abs(c2))
```

```
117     if minimum==abs(c1):
118         return c1
119     else:
120         return c2
121 
122 # Questa funzione permette di calcolare dove viene mandato un punto
123 # tramite un isogenia di grado 2
124 
125 def PhiB_i(P:Point,p,a,xr,c):
126     x=P[0]
127     l=mol(x,xr,p)
128     l=mod(l-1,p)
129     l=mol(l,l,p)
130     l=mol(l,x,p)
131     den=inv(pot(mod(x-xr,p),2,p),p)
132     x_new=mol(l,den,p)
133     y=P[1]
134     x2=pot(x,2,p)
135     xrx2=mol(xr,x2,p)
136     xrx=mol(xr,x,p)
137     xr2=pot(xr,2,p)
138     xr23=mol(xr2,3,p)
139     num=mol((xrx-1),(xrx2+mol(1-xr23,x,p)+xr),p)
140     den=inv(pot(x-xr,3,p),p)
141     fprimo=mol(num,den,p)
142     y=mol(c,mol(y,fprimo,p),p)
143     P_prime=ass(x_new,y)
144     print('le vecchie coordinate erano', P, 'invece le nuove sono',
        P_prime)
145     return P_prime
```

Listing 7.4. SIDH

Aquesto punto iniziala seconda parte di codice, quella dedicata al protocollo vero e proprio, con la definizione dei parametri pubblici ed il calcolo dei punti S_A ed S_B da parte di Alice e Bob.

```
1  # Inizia il vero e proprio algoritmo
2  
3  #parametri pubblici iniziali
4  p=431
5  a=423+329j
6  xPA=248+100j
7  yPA=199+304j
8  xQA=394+426j
9  yQA=79+51j
10 xPB=358j+275
11 yPB=410j+104
12 xQB=20j+185
13 yQB=281j+239
14 
15 PA=ass(xPA,yPA)
16 QA=ass(xQA,yQA)
17 PB=ass(xPB,yPB)
18 QB=ass(xQB,yQB)
```

```
19
20 kA=11
21 kB=2
22
23 expA=4
24 expB=3
25
26 esp=expA
27 SA=point_add(PA,point_mul(kA,QA,p),p)
28 SB=point_add(PB,point_mul(kB,QB,p),p)
29 j=j_inv(a,p)
```

Listing 7.5. SIDH

A questo punto del codice è implementata la parte in cui Alice e Bob svolgono il primo percorso sul grafo fino a giungere alle chiavi pubbliche da scambiarsi per poi proseguire.

```
1  print('RAPPRESENTAZIONE DEL GRAFO')
2  GA=nx.DiGraph()
3  GA.add_node(j)
4  nx.draw(GA, with_labels=True, font_weight='5',node_size=3000,font_size
       =8)
5  plt.show()
6
7  for i in range(1,esp+1,1):
8
9      print('Aggiornamento di A',i)
10     R=point_mul(2**(esp-i),SA,p)
11     a=new_a(x(R),p)
12     j_new=j_inv(a,p)
13     print('RAPPRESENTAZIONE DEL GRAFO')
14     GA.add_node(j)
15     GA.add_edge(j, j_new)
16     nx.draw(GA, with_labels=True, font_weight='5',node_size=3000)
17     plt.show()
18     c=calcolo_c_A(PB,p,a,x(R))
19     print('Aggiornamento di PB')
20     PB=(PhiA_i(PB,p,a,x(R),c))
21     print('Aggiornamento di QB')
22     QB=(PhiA_i(QB,p,a,x(R),c))
23     print('Aggiornamento di SA')
24     SA=(PhiA_i(SA,p,a,x(R),c))
25     j=j_new
26
27 print('Il j invariante e' dato da',j)
28
29 PkA=[a,PB,QB]
30 print(PkA)
31
32 print("-----------------------------------------------------------")
33
34 esp=expB
35 a=329j+423
36 j=190+87j
37 GB=nx.DiGraph()
```

```
GB.add_node(j)
nx.draw(GB, with_labels=True, font_weight='5',node_size=3000,font_size
    =8)
plt.show()

for i in range(1,esp+1,1):

    print('Aggiornamento di Bob',i)
    R=point_mul(3**(esp-i),SB,p)
    a=new_aB(x(R),p,a)
    j_new=j_inv(a,p)
    print('RAPPRESENTAZIONE DEL GRAFO')
    GB.add_node(j)
    GB.add_edge(j, j_new)
    nx.draw(GB, with_labels=True, font_weight='5',node_size=3000)
    plt.show()
    c=calcolo_c_B(PA,p,a,x(R))
    print('Aggiornamento di PA')
    PA=(PhiB_i(PA,p,a,x(R),c))
    print('Aggiornamento di QA')
    QA=(PhiB_i(QA,p,a,x(R),c))
    print('Aggiornamento di SB')
    SB=(PhiB_i(SB,p,a,x(R),c))
    j=j_new

print('Il j invariante e' dato da',j)

PkB=[a,PA,QA]
```

Listing 7.6. SIDH

In quest'ultima parte di codice Alice e Bob si scambiano la chiave pubblica e all'interno del ciclo for calcolano le isogenie che permettono loro di compiere gli ultimi spostamenti sul grafo fino a raggiungere il nodo comune.

```

SA=point_add(PkB[1],point_mul(kA,PkB[2],p),p)
#SB=point_add(PkA[1],point_mul(kB,PkA[2],p),p)
#SA=ass(125j+357,415j+249)

esp=expA
a=PkB[0]
j=j_inv(a,p)
print('RAPPRESENTAZIONE DEL GRAFO')
GA1=nx.DiGraph()
GA1.add_node(j)
nx.draw(GA1, with_labels=True, font_weight='5',node_size=3000,font_size
    =8)
plt.show()

print("------------------------------------------------------------")

for i in range(1,esp+1,1):

    print('Aggiornamento Alice',i)
```

```
20      R=point_mul(2**(esp-i),SA,p)
21      a=new_a(x(R),p)
22      j_new=j_inv(a,p)
23      print('RAPPRESENTAZIONE DEL GRAFO')
24      GA1.add_node(j)
25      GA1.add_edge(j, j_new)
26      nx.draw(GA1, with_labels=True, font_weight='5',node_size=3000,)
27      plt.show()
28      print('Il j invariante di A al passo', i, 'e'',j)
29      c=calcolo_c_A(PkB[1],p,a,x(R))
30      PkB[1]=(PhiA_i(PkB[1],p,a,x(R),c))
31      PkB[2]=(PhiA_i(PkB[2],p,a,x(R),c))
32      SA=(PhiA_i(SA,p,a,x(R),c))
33      j=j_new
34
35  j_A=j
36  print('Il j invariante e' dato da',j)
37
38  esp=expB
39  a=PkA[0]
40  #PkA=[423j+179, ass(142j+183,119j+360), ass(220j+314, 289j+10)]
41  SB=point_add(PkA[1],point_mul(kB,PkA[2],p),p)
42  j=j_inv(a,p)
43  GB1=nx.DiGraph()
44  GB1.add_node(j)
45  nx.draw(GB1, with_labels=True, font_weight='5',node_size=3000)
46  plt.show()
47
48  #SB=ass(393j+124,187j+380)
49
50  print("------------------------------------------------------------")
51
52  for i in range(1,esp+1,1):
53
54      print('Aggiornamento di Bob',i)
55      R=point_mul(3**(esp-i),SB,p)
56      a=new_aB(x(R),p,a)
57      j_new=j_inv(a,p)
58      print('RAPPRESENTAZIONE DEL GRAFO')
59      GB1.add_node(j_new)
60      GB1.add_edge(j, j_new)
61      nx.draw(GB1, with_labels=True, font_weight='3',node_size=3000,
        font_size=8)
62      plt.show()
63      print('Il j invariante di B al passo', i, 'e'',j)
64      c=calcolo_c_B(PkA[1],p,a,x(R))
65      PkA[1]=(PhiB_i(PkA[1],p,a,x(R),c))
66      PkA[2]=(PhiB_i(PkA[2],p,a,x(R),c))
67      SB=(PhiB_i(SB,p,a,x(R),c))
68      j=j_new
69
70
71  print('Il j invariante e' dato da',j)
```

```
72
73 print('I due j-invarianti che costituiscono la chiave pubblica condivisa
       sono:', j_A,j)
```

Listing 7.7. SIDH

Risultati

Seguono i risultati che si ottengono facendo girare il codice descritto precedentemente. Occorre osservare che, nei risultati, è stato rimosso il grafo dato in output ad ogni iterazione, per non appesantire troppo la trattazione. Tuttavia si può avere una visione grafica nella Figura 7.1 già commentata in precedenza.

```
1  Aggiornamento di A 1
2  Aggiornamento di PB
3  le vecchie coordinate erano ((275+358j), (104+410j)) invece le nuove
       sono ((85+118j), (150+274j))
4  Aggiornamento di QB
5  le vecchie coordinate erano ((185+20j), (239+281j)) invece le nuove sono
        ((124+62j), (269+64j))
6  Aggiornamento di SA
7  le vecchie coordinate erano ((79+271j), (430+153j)) invece le nuove sono
        ((111+36j), (67+175j))
8  Aggiornamento di A 2
9  Aggiornamento di PB
10 le vecchie coordinate erano ((85+118j), (150+274j)) invece le nuove sono
        ((251+274j), (59+316j))
11 Aggiornamento di QB
12 le vecchie coordinate erano ((124+62j), (269+64j)) invece le nuove sono
       ((94+214j), (193+354j))
13 Aggiornamento di SA
14 le vecchie coordinate erano ((111+36j), (67+175j)) invece le nuove sono
       ((374+274j), (77+84j))
15 Aggiornamento di A 3
16 Aggiornamento di PB
17 le vecchie coordinate erano ((251+274j), (59+316j)) invece le nuove sono
        ((209+77j), (352+356j))
18 Aggiornamento di QB
19 le vecchie coordinate erano ((94+214j), (193+354j)) invece le nuove sono
        ((356+339j), (12+419j))
20 Aggiornamento di SA
21 le vecchie coordinate erano ((374+274j), (77+84j)) invece le nuove sono
       ((150+227j), 0j)
22 Aggiornamento di A 4
23 Aggiornamento di PB
24 le vecchie coordinate erano ((209+77j), (352+356j)) invece le nuove sono
        ((183+142j), (71+312j))
25 Aggiornamento di QB
26 le vecchie coordinate erano ((356+339j), (12+419j)) invece le nuove sono
        ((314+220j), (421+142j))
27 Aggiornamento di SA
28 le vecchie coordinate erano ((150+227j), 0j) invece le nuove sono (0j, 0
       j)
```

```
il j invariante e' dato da (118+222j)
--------------------------------------------------------------------------------
Aggiornamento di Bob 1
Aggiornamento di PA
le vecchie coordinate erano ((248+100j), (199+304j)) invece le nuove sono ((155+418j), (331+288j))
Aggiornamento di QA
le vecchie coordinate erano ((394+426j), (79+51j)) invece le nuove sono ((242+159j), (425+310j))
Aggiornamento di SB
le vecchie coordinate erano ((309+122j), (374+291j)) invece le nuove sono ((256+295j), (64+253j))
Aggiornamento di Bob 2
Aggiornamento di PA
le vecchie coordinate erano ((155+418j), (331+288j)) invece le nuove sono ((425+252j), (19+315j))
Aggiornamento di QA
le vecchie coordinate erano ((242+159j), (425+310j)) invece le nuove sono ((81+412j), (172+111j))
Aggiornamento di SB
le vecchie coordinate erano ((256+295j), (64+253j)) invece le nuove sono ((405+102j), (313+375j))
Aggiornamento di Bob 3
Aggiornamento di PA
le vecchie coordinate erano ((425+252j), (19+315j)) invece le nuove sono ((226+187j), (71+388j))
Aggiornamento di QA
le vecchie coordinate erano ((81+412j), (172+111j)) invece le nuove sono ((415+325j), (177+109j))
Aggiornamento di SB
le vecchie coordinate erano ((405+102j), (313+375j)) invece le nuove sono (0j, 0j)
il j invariante e' dato da (190+344j)
--------------------------------------------------------------------------------
Aggiornamento Alice 1
il j invariante di A al passo 1 e' (190+344j)
le vecchie coordinate erano ((226+187j), (71+388j)) invece le nuove sono ((79+268j), (144+319j))
le vecchie coordinate erano ((415+325j), (177+109j)) invece le nuove sono ((428+257j), (319+155j))
le vecchie coordinate erano ((357+125j), (182+16j)) invece le nuove sono ((290+27j), (316+38j))
Aggiornamento Alice 2
il j invariante di A al passo 2 e' (304+364j)
le vecchie coordinate erano ((79+268j), (144+319j)) invece le nuove sono ((39+16j), (17+410j))
le vecchie coordinate erano ((428+257j), (319+155j)) invece le nuove sono ((144+414j), (245+243j))
le vecchie coordinate erano ((290+27j), (316+38j)) invece le nuove sono ((408+408j), (111+406j))
Aggiornamento Alice 3
il j invariante di A al passo 3 e' (67+0j)
```

```
66 le vecchie coordinate erano ((39+16j), (17+410j)) invece le nuove sono
       ((114+83j), (15+364j))
67 le vecchie coordinate erano ((144+414j), (245+243j)) invece le nuove
       sono ((161+215j), (220+52j))
68 le vecchie coordinate erano ((408+408j), (111+406j)) invece le nuove
       sono (196j, 0j)
69 Aggiornamento Alice 4
70 il j invariante di A al passo 4 e' (242+0j)
71 le vecchie coordinate erano ((114+83j), (15+364j)) invece le nuove sono
       ((68+385j), (296+18j))
72 le vecchie coordinate erano ((161+215j), (220+52j)) invece le nuove sono
        ((382+405j), (106+256j))
73 le vecchie coordinate erano (196j, 0j) invece le nuove sono (0j, 0j)
74 il j invariante e' dato da (234+0j)
75 ------------------------------------------------------------------------
76 Aggiornamento di Bob 1
77 a di Bob al passo 1 e' (177+183j)
78 il j invariante di B al passo 1 e' (118+222j)
79 le vecchie coordinate erano ((183+142j), (71+312j)) invece le nuove sono
        ((64+102j), (351+400j))
80 le vecchie coordinate erano ((314+220j), (421+142j)) invece le nuove
       sono ((315+177j), (303+140j))
81 le vecchie coordinate erano ((124+393j), (51+244j)) invece le nuove sono
        ((49+9j), (150+352j))
82 Aggiornamento di Bob 2
83 a di Bob al passo 2 e' (31+0j)
84 il j invariante di B al passo 2 e' (315+299j)
85 le vecchie coordinate erano ((64+102j), (351+400j)) invece le nuove sono
        ((346+213j), (51+150j))
86 le vecchie coordinate erano ((315+177j), (303+140j)) invece le nuove
       sono ((326+15j), (45+152j))
87 le vecchie coordinate erano ((49+9j), (150+352j)) invece le nuove sono
       ((307+0j), (158+0j))
88 Aggiornamento di Bob 3
89 a di Bob al passo 3 e' (230+0j)
90 il j invariante di B al passo 3 e' (61+0j)
91 le vecchie coordinate erano ((346+213j), (51+150j)) invece le nuove sono
        ((350+126j), (23+361j))
92 le vecchie coordinate erano ((326+15j), (45+152j)) invece le nuove sono
       (103j, (30+64j))
93 le vecchie coordinate erano ((307+0j), (158+0j)) invece le nuove sono (0
       j, 0j)
94 il j invariante e' dato da (234+0j)
95 I due j-invarianti che costituiscono la chiave pubblica condivisa sono:
       (234+0j) (234+0j)
```

Listing 7.8. SIDH

Effettivamente, come si nota dalla riga 95 dei risultati l'implementazione ha fornito i risultati corretti.
Un altro esempio è quello che si osserva nella Figura 7.2 in cui si sono tenuti tutti i parametri uguali a quelli precedenti tranne le chiavi private di Alice e Bob che sono state posta rispettivamente pari a 15 e 7. Anche in questo caso, si può osservare, Alice e Bob

riescono a raggiungere un nodo condiviso che avrà proprio il valore della chiave condivisa nel protocollo di Diffie-Hellman.

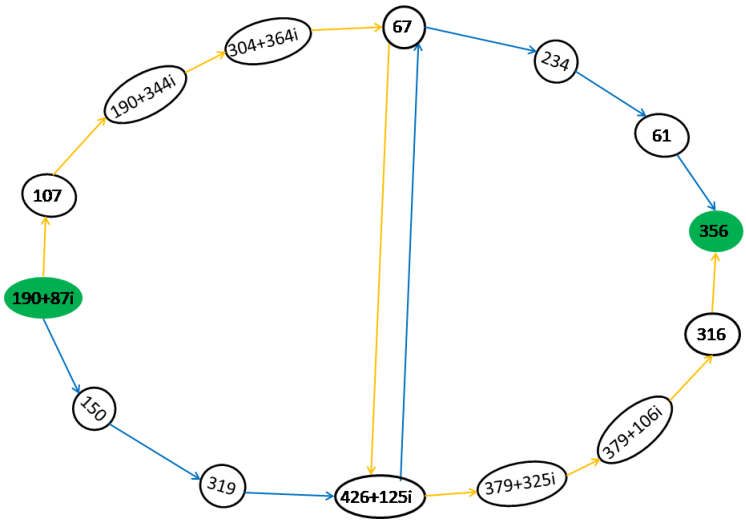

Figura 7.2. Secondo esempio di implementazione SIDH

Commenti sul codice

Il codice presentato non è certamente ben ottimizzato, tuttavia permette comunque di avere un'implementazione funzionante del protocollo. Un limite del codice implementato è dettato dal fatto che stiamo lavorando con il calcolo delle isogenie sul piano affine ma le curve ellittiche sono curve proiettive, di conseguenza esisterà sempre un punto che viene mandato all'infinito e che, se scelto, fa sì che il protocollo non funzioni. Per risolvere questo problema occorrerebbe aggiungere un controllo che riconosca tale punto e cambi di conseguenza la scelta di α, ossia del parametro che annulla di denominatore della formula di duplicazione, per le isogenie di secondo grado, e di triplicazione, per le isogenie di terzo grado. Così facendo il punto considerato non sarà più mandato all'infinito e il protocollo funzionerà nuovamente.
Ciò in cui eccelle particolarmente questo protocollo è nella dimensione estremamente ridotta delle chiavi sufficiente a garantire la sicurezza che è, in assoluto, la più piccola tra i protocolli post quantum. Questo perché la dimensione dello spazio delle chiavi rimane comunque estremamente ridotta rispetto alla dimensione del grafo di isogenie. La dimensione stessa delle chiavi può essere inoltre ancora ridotta utilizzando delle tecniche di compressione delle chiavi, reperibili in [11].

7.5 Considerazioni

Negli esempi numerici precedenti si sono presi in considerazione dei valori di e_A ed e_B decisamente piccoli. Ciò che accade effettivamente affinché il protocollo possa essere utilizzato per garantire la sicurezza crittografica è di prendere valori ben più alti. Infatti i più comuni sono $e_A = 216$ ed $e_B = 137$. Così facendo si ha un numero di j-invarianti compreso tra 2^{429} e 2^{430}, ben più alto rispetto ai 37 j-invarianti degli esempi precedenti.
Quando aumenta così considerevolmente la dimensione ci si pone davanti il problema della complessità computazionale. Infatti nel codice precedentemente presentato, definito n come $n = 2e_A + 2e_B$, ci sono n iterazioni di cicli for che coinvolgono il calcolo di una moltiplicazione di S_A o S_B, l'aggiornamento del parametro a della curva con il relativo calcolo del j-invariante e l'aggiornamento di 3 punti della curva ellittica. Quindi la complessità computazionale è abbastanza elevata pur rimanendo polinomiale. Una soluzione parziale, proposta da De Feo e Jao stessi [12] è quella di calcolare la moltiplicazione del generatore solo una volta e memorizzare tutti i passaggi coinvolti. Infatti per calcolare il valore ad esempio di $[2^m]P$ occorre conoscere anche $[2^{m-1}]P$ e così via. Questo procedimento abbasserebbe leggermente la complessità computazionale.
La sicurezza di tale protocollo si basa sulla difficoltà in generale di calcolare, date due curve ellittiche, un'isogenia tra di esse. Infatti anche i migliori algoritmi post quantum necessitano di un tempo esponenziale per effettuare tale calcolo. Maggiori dettagli possono essere trovati in [12].
Concludiamo osservando che in questa tesi abbiamo analizzato in dettaglio il protocollo di scambio chiavi SIDH, tuttavia esistono altri protocolli crittografici definibili sui grafi di isogenie. Così come avviene per le curve ellittiche, potrebbero infatti essere anche ridefiniti i protocolli di El-Gamal e il protocollo di Rostovtsev-Stolbunov. Per quanto riguarda invece i protocolli relativi alle firme digitali, essi sono più difficili da costruire in questo caso poiché basano il loro funzionamento sull'esistenza di una legge di gruppo sui dati pubblici, che non è sempre garantita nel caso delle isogenie supersingolari, come si legge in [1].

Bibliografia

[1] L. De Feo, *Mathematics of Isogeny Based Cryptography*, Université de Versailles & Inria Saclay, Thiès, Senegal, 2017.

[2] J. H. Silverman, *The Arithmetic of Elliptic Curves*, Springer, II Edition, 2009.

[3] J. H. Silverman, J. T. Tate, *Rational Points on Elliptic Curves*, Springer, II Edition, 2015.

[4] L. Capuano, A. Ferraguti, *Note del corso di dottorato "Curve ellittiche"*, Politecnico di Torino, 2020.

[5] A. Di Pierro, O. Morsch, *Computer Quantistici*, Mondo digitale, 2013.

[6] A. Conte, L. Picco Botta, D. Romagnoli, *Algebra*, Levrotto e Bella, 2009.

[7] C. Costello, B. Smith, *Montgomery curves and their arithmetic*, Journal of Cryptographic Engineering volume 8, pages 227–240, 2018.

[8] T. Tao, *Expansion in groups of Lie Type-basic theory of expander graphs*, https://terrytao.wordpress.com/2011/12/02/245b-notes-1-basic-theory-of-expander-graphs/, 2011.

[9] C. Costello, *Supersingular isogeny key exchange for beginners*, International Conference on Selected Areas in Cryptography - SAC 2019 - pp. 21–50, Springer, 2019.

[10] G. Adj, J. Chi-Domínguez e F. Rodríguez-Henríquez, *On new Vélu's formulae and their applications to CSIDH and B-SIDH constant-time implementations*, Cryptology ePrint Archive, Report 2020/1109, 2020.

[11] R. Azarderakhsh, D. Jao, K. Kalach, B. Koziel, C. Leonardi, *Key compression for isogeny-based cryptosystems*, Advances in Cryptology – CRYPTO 2016, pp. 572-601, 2016.

[12] L. De Feo, D. Jao, J. Plût, *Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies.* Journal of Mathematical Cryptology, 8(3): 209–247, 2014.

[13] S. J. Lomonaco, *Shor's Quantum Factoring Algorithm*, Proceedings of Symposia in Applied Mathematics, 58, 2002.

[14] R. Schoof, *Nonsingular Plane Cubic Curves over Finite Fields*, J. Combin. Theory Ser. A 46, 1987, no. 2, 183–211.

[15] F. Pintore, *An Introduction to Isogeny-based Crytptography, PhD Course*, Department of Mathematics, University of Bari, 2021.

Ringraziamenti

Ringrazio prima di tutto la dottoressa Laura Capuano che, oltre ad aver messo a mia disposizione le sue vaste conoscenze e ad avermi offerto gli strumenti necessari, si è rivelata davvero una bella persona, acoppagnandomi in questo percorso con serenità e aiutandomi a dare il meglio di me.
Ringrazio poi mamma e papà che mi hanno sopportata in questi lunghi 5 anni, tra minacce di abbandonare l'università e braccia rotte.
Ringrazio Billo perché mi ha sopportata durante la sessione e perché è stato il mio braccio sinistro in un periodo davvero difficile per me.
Ringrazio il professor Silvio Mercadante perchè, anche se lui non lo sa, ha creduto in me anche quando io stessa non riuscivo a farlo, spronandomi a tirar fuori le mie capacità e indirizzandomi verso il percorso più giusto per me.
Ringrazio tutti i miei compagni di corso, soprattutto Marco, che mi ha sopportata in innumerevoli progetti di gruppo e che c'è stato per me quando ho avuto davvero bisogno di un amico al mio fianco. Luca, che rimarrà sempre il mio compagno di scleri senza senso e Adri che si è accertato che non perdessi anche la poca sanità mentale che mi era rimasta.
Non potrei non ringraziare anche tutti gli amici dello skate che sono entrati nella mia vita per caso e non ne sono più usciti, che hanno fatto si che ogni tanto dimenticassi il poli in mezzo a tante risate.
Ringrazio anche la tavola, perché nonostante tutto mi ha tenuta a galla durante una pandemia e mi ha fatto avere voglia di rimettermi a posto dopo il 6 gennaio.
Ringrazio Mariaannina e Pietro perché...loro lo sanno il perché.
Ringrazio tutti gli amici che ho conosciuto e con cui ho scambiato pezzetti di vita, tutti voi che avete fatto di Torino la mia casa lontano da casa.
Infine, ringrazio me stessa, perché anche quando non ci credevo più ho continuato a provarci. Perché sono riuscita sempre a mettermi in gioco e a superare le mie stesse aspettative. Perché quando 5 anni fa sono entrata nel poli per la prima volta di certo non mi sarei immaginata di vivere un'esperienza così bella, complicata ed emozionante. Perché ci sono riuscita senza mai mettere da parte le mie passioni e i miei interessi. Perché nonostante tutto, ci sono riuscita!