Abstract

L'obiettivo di questo documento è presentare al lettore un test di primalità con tempo medio di esecuzione polinomiale rispetto alla dimensione dell'input. Questo test è basato sui lavori di Goldwasser e Kilian e mira a verificare la primalità di un numero utilizzando alcune nozioni di teoria dei gruppi applicate a gruppi generati da curve ellittiche su campi finiti. Faremo una presentazione introduttiva sulle curve ellittiche (Capitolo Uno) in modo che il lettore abbia tutti gli strumenti necessari per poter analizzare e comprendere l'algoritmo e farne una analisi elementare sulla complessità computazionale (Capitolo Due).

Indice

1 Nozioni elementari su curve ellittiche **1**
1.1 Definizione di curva ellittica 1
1.2 Legge di gruppo su curve ellittiche 4
1.3 Proprità delle curve ellittiche 6

2 Test di Primalittà su curve ellittiche **8**
2.1 Nozioni di teoria 8
2.2 Descrizione dell'algoritmo 10
2.3 Analisi degli algoritmi utilizzati 12
2.4 Complessità computazionale e considerazioni 14

Appendices **17**

A Test di primalità proabilistici **18**

Capitolo 1

Nozioni elementari su curve ellittiche

1.1 Definizione di curva ellittica

Definizione 1.1.1. ***(Curva Ellittica)***
Una curva ellittica, definita su un campo K, è il grafico di una equazione, detta Equazione di Weierstrass, della forma

$$y^2 = x^3 + Ax + B \tag{1.1}$$

in cui $A, B \in K$ sono costanti. Con la notazione $E_{A,B}(K)$ indichiamo una tale curva in cui i parametri A, B sono specificati e in cui viene aggiunto un punto formale ∞, ovvero:

$$E_{A,B}(K) \coloneqq \{\infty\} \cup \{(x, y) \in K \times K : y^2 = x^3 + Ax + B\}.$$

Dove le proprietà formali del punto ∞ verrano specificate a seguire. Nei casi cui non sia importante specificare i parametri A, B, possiamo indicare una curva ellittica con la notazione $E(K)$.
Se $4A^3 + 27B^2 = 0$, allora la curva è detta singolare.

All'interno della nostra trattazione, assumiamo sempre che la curva sia non singolare.

Osservazione 1.1.1 L'ipotesi K campo è necessaria per dimostrare molte delle proprietà delle curve ellittiche in seguito utilizzate. Tuttavia, più in generale, potremmo definire i termini A, B ed x di Equazione (1.1) non in un campo K ma in un insieme differente, come ad esempio $\mathbb{Z}/n\mathbb{Z}$ con $n > 1$ intero.
Osserviamo che una curva ellittica definita su un tale insieme, pur non esseno in grado di soddisfare alcune delle ipotesi ai teoremi in seguito presentati, continua ad avere significato e per questo con la notazione $E_{A,B}(\mathbb{Z}/n\mathbb{Z})$ intenderemo una curva ellittica anche se $\mathbb{Z}/n\mathbb{Z}$ non è sempre un campo.

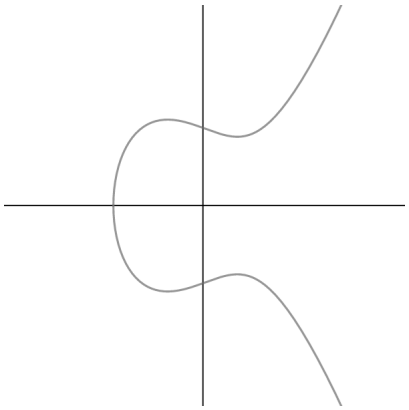

Figura 1.1: *Esempio di curva ellittica a valori reali*

Osservazione 1.1.2 Potremmo enunciare una definizione più generale di curva ellittica tramite l'equazione seguente:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{1.2}$$

dove $a_i \in K$ costante $\forall\, i \in \{1, ..., 6\}$.

Tuttavia è possibile dimostrare che se il campo K su cui è costruita la curva ha caratteristica *diversa* da 2 e 3, allora l'equazione (1.2) può essere ricondotta ad una della forma (1.1), infatti

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \implies$$
$$\left(y + \frac{a_1}{2}x + \frac{a_3}{2}\right)^2 = x^3 + \left(a_2 + \frac{a_1^2}{4}\right)x^2 + \left(a_4 + \frac{a_1a_3}{2}\right)x + \left(\frac{a_3^2}{4} + a_6\right).$$

Se $\mathrm{char}(K) \neq 2$, possiamo porre

$$\tilde{y} \overset{s}{=} y + \frac{a_1}{2}x + \frac{a_3}{2}$$

e ottenere

$$\begin{aligned}\tilde{y}^2 &= x^3 + a_2'x^2 + a_4'x + a_6' \\ &= \left(x^3 + x^2a_2' + x\frac{a_2'^2}{3} + \frac{a_2'^3}{27}\right) + \left(xa_4' - x\frac{a_2'^2}{3} - \frac{a_2'a_4'}{3} + \frac{a_2'^3}{9}\right) + k \\ &= \left(x + \frac{a_2'}{3}\right)^3 + \left(x + \frac{a_2'}{3}\right)\left(a_4' + \frac{a_2'^2}{3}\right) + k.\end{aligned}$$

Se $\mathrm{char}(K) \neq 3$, possiamo porre

$$\tilde{x} \overset{s}{=} x + \frac{a_2'}{3}$$

e ottenere dunque

$$\tilde{y}^2 = \tilde{x}^3 + A\tilde{x} + B.$$

Dove i coefficienti $a_2', a_4', a_6', k, A, B \in K$ sono nuove costanti opportunamente definiti.

Osservazione 1.1.3 Possiamo creare una struttura di gruppo su una curva ellittica non singolare. Per compiere questa operazione, ci serviremo di una formulazione di curva ellittica basata sul piano proiettivo, in cui il punto ∞ verrà propriamente definito.

Ricordiamo per comodità la definizione di piano proiettivo.

Definizione 1.1.2. ***(Piano Proiettivo)***
Dato un campo K, il piano proiettivo $\mathbb{P}^2(K)$ sul campo K è l'insieme delle classi di equivalenza della relazione $\sim$ defnita su $K^3/\{(0,0,0)\}$ e data da $(x_1,x_2,x_3) \sim (y_1,y_2,y_3)$ se e solo se esiste $\lambda \in K\backslash\{0\}$ tale che $(x_1,x_2,x_3) = = (\lambda y_1, \lambda y_2, \lambda y_3)$.
La classe di equivalenza di (x_1,x_2,x_3) viene denotata con $(x_1 : x_2 : x_3)$.

Osservazione 1.1.4 Tutti i punti $P \in \mathbb{P}^2(K)$ appartengono ad una delle seguenti classi di equivalenza:

- $P = (x : y : 1)$ per certi $x, y \in K$,
- $P = (x : y : 0)$ per certi $x, y \in K$, non entrambi nulli.

I punti del primo tipo sono detti *punti finiti* di $P \in \mathbb{P}^2(K)$, mentre i punti del secondo tipo sono detti *punti all'infinito* di $P \in \mathbb{P}^2(K)$.

Ripetiamo la definizione di curva ellittica nel caso particolare in cui i punti appartengono al piano proiettivo.

Definizione 1.1.3. ***(Curva Ellittica)***
Consideriamo l'equazione di Wierstrass (1.1) in cui $A, B \in K$ e $4A^3+27B^2 = 0$. Sul piano proiettivo, la sua forma omogenea risulta essere

$$y^2z = x^3 + Axz^2 + Bz^3. \tag{1.3}$$

I punti $(x : y : z) \in \mathbb{P}^2(K)$ soluzione di questa equazione definiscono una curva ellittica.

Osservazione 1.1.5 Consideriamo una curva ellittica su un campo K. Allora i punti (x, y) della curva originale corrispondono ai punti $(x : y : 1)$ nella sua versione proiettiva. Per mostrare questo fatto, è sufficiente ricordare che l'inclusione canonica

$$K \times K \hookrightarrow \mathbb{P}^2(K)$$

definita da

$$(x, y) \mapsto (x : y : 1)$$

è biettiva.
Se consideriamo la versione proiettiva dell'Equazione di Wierstrass e poniamo $z = 0$, abbiamo che $x = 0$ e dunque il punto $(0 : 1 : 0) \in \mathbb{P}^2(K)$ appartiene alla curva ellittica. Viene detto *punto all'inifinito* ed è identificato con quello che precedentemente era denotato con ∞.

1.2 Legge di gruppo su curve ellittiche

Mostriamo come definire una operazione di somma + tra punti di una curva ellittica.

Sia $E_{A,B}(K)$ una curva ellittica non singolare su un campo K e siano $P_1, P_2 \in E_{A,B}(K)$. Per ora non facciamo alcuna ipotesi su P_1, P_2:

$$P_1 = (x_1, y_1), \qquad P_2 = (x_2, y_2).$$

Vogliamo defnire un terzo punto $P_3 := P_1 + P_2$. L'idea è quella di tracciare la retta passante per P_1, P_2, intersecarla con la curva $E_{A,B}(K)$ e poi riflettere il punto di intersezione rispetto all'asse delle ascisse. Formalizziamo questo concetto, procedendo per casi.
Caso 1 Supponiamo $P_1 \neq P_2$, ed entrambi diversi da ∞. Sia L retta passante per P_1, P_2.
Caso 1.1 Supponiamo $x_1 \neq x_2$. Allora la retta L ha coefficiente angolare

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

Dunque L ha equazione $y = m(x - x_1) + y_1$ ed interseca $E_{A,B}(K)$ nel punto

$$(m(x - x_1) + y_1)^2 = x^3 + Ax + B$$

che può essere scirtto nella forma

$$\begin{aligned} f(x) &:= x^3 + x^2\left(-m^2\right) + x\left(A - 2my_1 + 2mx_1\right) + \\ &\quad + \left(B - y_1^2 - (mx_1)^2 + 2mx_1y_1\right) \\ &= x^3 + x^2a + xb + c. \end{aligned}$$

Osserviamo che le tre radici di f devono ovviamente essere (x_1, x_2, x_3), con x_1, x_2 noti e x_3 ascissa del terzo punto di intersezione. Possiamo dunque calcolare ascissa e ordinata del punto P_3. In particolare possiamo dire che

$$\begin{aligned} f(x) &= x^3 + ax^2 + bx + c \\ &= (x - x_1)(x - x_2)(x - x_3) \\ &= x^3 - x^2\left(x_1 + x_2 + x_3\right) + x\left(x_1x_2 + x_2x_3 + x_1x_3\right) - x_1x_2x_3. \end{aligned}$$

Ne viene che $x_1 + x_2 + x_3 = m^2$ e pertanto

$$P_3 = \left(m^3 - x_1 - x_2, -m(x_3 - x_1) - y_1\right).$$

Caso 1.2 Supponiamo $x_1 = x_2$, di conseguenza dobbiamo avere $y_1 = -y_2$ ed L è una retta verticale che interseca $E(K)$ nel punto ∞, cioé

$$P_3 = P_1 + P_2 = \infty.$$

Caso 2 Supponiamo $P_1 = P_2 = (x_1, y_1)$. L è retta tangente ad E e passante per il punto P_1. Il suo coefficiente angolare m è

$$m = \left.\frac{dy}{dx}\right|_{(x_1,y_1)} = \frac{3x_1^2 + A}{y_1}.$$

Caso 2.1 Se $y_1 = 0$, allora L è una retta verticale e $P_1 + P_2 = 2P_1 = \infty$.
Caso 2.2 Se $y_1 \neq 0$, L è descritta dall'equazione

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

e si procede come in *Caso 1.1.*
Caso 3 Se $P_2 = \infty$, L è una retta verticale e interseca $E(K)$ nel punto $(x_1, -y_1)$, dunque

$$P_1 + \infty = (x_1, -(-y_1)) = (x_1, y_1) = P_1.$$

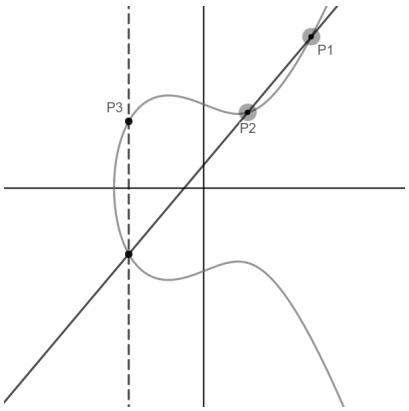

Figura 1.2: *Rappresentazione grafica di operazione di somma su una curva ellittica reale*

Possiamo ora enunciare il seguente teorema, una cui dimostrazione può essere trovata su [2].

Teorema 1.2.1. ***(Group Law for Elliptic Curves)*** *Sia $E_{A,B}(K)$ una curva ellittica non singolare su campo K, con $A, B \in K$, e sia $+$ operazione come sopra definita su punti di $E_{A,B}(K)$.*

Allora $E_{A,B}(K)$ è chiusa rispetto a $+$ e $(E_{A,B}(K), +, \infty)$ è un gruppo abeliano.

Osservazione 1.2.1 Dato che $\mathbb{Z}/n\mathbb{N}$ non è sempre un campo, il teorema appena mostrato non vale in generale per una curva ellittica del tipo $E_{A,B}(\mathbb{Z}/n\mathbb{Z})$. Ne viene che se $L, M \in E_{A,B}(\mathbb{Z}/n\mathbb{Z})$ sono punti di una tale curva ellittica, potrebbe essere che $L + M$ non sia ben definito. Tuttavia nei casi cui $L + M$ è definito vale il seguente lemma, che useremo in seguito.

Lemma 1.2.2. *Sia n intero con $2 \nmid n$, $3 \nmid n$. Sia $p > 3$ un divisore primo di n e siano A, B tali che* $\gcd(4A^3 + 27B^2, p) = 1$.
Per ogni $x \in \mathbb{Z}/n\mathbb{Z}$ definisco $x_p \in \mathbb{Z}/p\mathbb{Z}$ come $x_p \coloneqq x \pmod{p}$. Per ogni $L = (x, y) \in E_{A,B}(\mathbb{Z}/n\mathbb{Z})$ deinfisco $L_p \coloneqq (x_p, y_p) \in E_{A,B}(\mathbb{Z}/p\mathbb{Z})$, infine definisco $\infty_p \coloneqq \infty$.
Se $L, M \in E_{A,B}(\mathbb{Z}/n\mathbb{Z})$ e se $L + M$ è definito, allora vale che:

$$(L + M)_p = L_p + M_p.$$

Per una dimostrazione di questo lemma, si veda [1].

1.3 Proprità delle curve ellittiche

Nella sezione precedente abbiamo mostramo come costruire una struttura di gruppo abeliano su una curva ellittica. Se la curva è però definita su un campo finito $\mathbb{F}_q$, possiamo fare affermazioni più forti.

Teorema 1.3.1. *Sia $E(\mathbb{F}_q)$ una curva ellittica definita su un campo finito $\mathbb{F}_q$. Allora vale una delle seguenti:*

i. $E(\mathbb{F}_q) \cong \mathbb{Z}_n$ *per un certo* $n \in \mathbb{N}$,

ii. $E(\mathbb{F}_q) \cong \mathbb{Z}_{n_1} \oplus \mathbb{Z}_{n_2}$ *oer certi* $n_1, n_2 \in \mathbb{N}$ *con* $n_1 \mid n_2$.

Dimostrazione. Per il Teorema 1.2.1, $E(\mathbb{F}_q)$ è un gruppo abeliano finito. Possiamo utilizzare il teorema di struttura per gruppi abeliani finitiamente generati e dire che esiste $k \in \mathbb{N}$ tale che

$$E(\mathbb{F}_q) \cong \mathbb{Z}_{p_1} \oplus ... \oplus \mathbb{Z}_{p_k}$$

per certi $\{p_i\}_{i=1...k}$ potenze di interi primi. Usando il teorema cinese del resto, possiamo enunciare una formulazione equivalente e dire che esistono r interi n_i, con $i \in \{1, ..., r\}$ tali che

$$E(\mathbb{F}_q) \cong \mathbb{Z}_{n_1} \oplus ... \oplus \mathbb{Z}_{n_r} \quad \text{con } n_i \mid n_{i+1}.$$

Dunque per ogni indice i, si ha che $\mathbb{Z}_{n_i}$ contiene n_1 elementi il cui ordine divide n_1; ne viene che $E(\mathbb{F}_q)$ contiene n_1^r elementi il cui ordine divide n_1. A questo punto ci avvaliamo del del seguente teorema, una cui dimostrazione può essere trovata in [2].

Teorema 1.3.1. ***(Structure of Torsion Points)***
Sia $E(K)$ curva ellittica su campo K. Sia $\overline{K}$ la chiusura algebrica del campo K. Definisco i punti di n-torsione come:

$$E[n] \coloneqq \left\{P \in E\left(\overline{K}\right) : nP = \infty\right\}.$$

Se la caratteristica di K non divide n o non è 0, allora

$$E[n] \cong \mathbb{Z}_n \oplus \mathbb{Z}_n.$$

Se invece $p \coloneqq \operatorname{char}(K)$, con $p > 0, p \mid n$, scriviamo $n = p^r n'$, con $p \nmid n'$, e vale che

$$E[n] \cong \mathbb{Z}_{n'} \oplus \mathbb{Z}_{n'} \quad \textit{oppure} \quad E[n] \cong \mathbb{Z}_n \oplus \mathbb{Z}_{n'}$$

Applichiamo quindi il teorema di Struttura sui Gruppi di Torsione al gruppo $E(\mathbb{F}_q)$. Dato che $\operatorname{char}(K) = p$, per qualche primo p, ho che

$$E[n_1] \cong \mathbb{Z}_{n_1} \oplus \mathbb{Z}_{n_1} \text{ oppure } E[n_1] \cong \mathbb{Z}_{n_1} \oplus \mathbb{Z}_{n_1'} \implies$$

$$\implies \#E[n_1] \leq n_1^2 \implies r \leq 2$$

Dunque ottengo la tesi

$$E(\mathbb{F}_q) \cong \mathbb{Z}_n \quad \text{oppure} \quad E(\mathbb{F}_q) \cong \mathbb{Z}_{n_1} \oplus \mathbb{Z}_{n_2}$$

□

Vediamo adesso un risultato importante sulle curve ellittiche noto come *Disuguaglianza di Hasse*. L'obiettivo è stimare il numero di punti di una curva $E_{A,B}(\mathbb{Z}/p\mathbb{Z})$. Osservando l'Equazione di Wierstrass 1.1 e supponendo che $x \in \mathbb{Z}/p\mathbb{Z}$, abbiamo una ovvia maggiorazione in

$$\#E_{A,B}(\mathbb{Z}/p\mathbb{Z}) \leq 2p + 1.$$

Ricordiamo adesso che nel gruppo $\mathbb{Z}/p\mathbb{Z}$ solamente metà dei suoi elementi sono residui quadratici. Per mostrare questo fatto, osserviamo che:

- sia $x \in (\mathbb{Z}/p\mathbb{Z})^*$ radice primitiva. Gli elementi di $(\mathbb{Z}/p\mathbb{Z})^*$ possono essere scritti come x^k, con $k \in \{1, \ldots, p-1\}$. Per i valori pari di $k = 2m$ posso individuare un residuo quadratico $x^{2m} = (x^k)^2 \equiv a \pmod{p}$. Dato che x è generatore di $(\mathbb{Z}/p\mathbb{Z})^*$ e metà degli esponenti k sono pari, almeno la metà degli elementi di $(\mathbb{Z}/p\mathbb{Z})^*$ sono residui quadratici.

- Supponiamo che $b^2 \equiv a \pmod{p}$, allora $(-b)^2 \equiv a \pmod{p}$ e dato che per $p > 2$ si cha che $b \neq -b$, dunque al più metà degli elementi di $(\mathbb{Z}/p\mathbb{Z})^*$ sono residui quadratici.

Di conseguenza, una equazione della forma

$$y^2 \equiv C \pmod{p}$$

ha probabilità di circa il $1/2$ di avere soluzione se l'intero C è scelto casualmente (in maniera uniforme). Questo fatto ci suggerisce che la cardinalità di $E_{A,B}(\mathbb{Z}/p\mathbb{Z})$è dell'ordine di p. Per una stima più precisa, ci avvaliamo del seguente teorema una cui dimostrazione può essere trovata in [2] e in [3].

Teorema 1.3.2. ***(Hasse inequality)*** *Sia E una curva ellittica su un campo finito $\mathbb{F}_q$. Allora vale che:*

$$|q + 1 - \#(E(\mathbb{F}_q))| \leq 2\sqrt{q} \tag{1.4}$$

o, equivalentemente

$$\#(E(\mathbb{F}_q)) \in \left[(\sqrt{q} - 1)^2, (\sqrt{q} + 1)^2\right]. \tag{1.5}$$

Capitolo 2

Test di Primalittà su curve ellittiche

In questo capitolo presentiamo un test di primalità che sfrutta le proprietà delle curve ellittiche come presentato in [1].

2.1 Nozioni di teoria

Introduciamo il seguente teorema generale valido per tutti i gruppi ciclici e presentato in [4]. L'obiettivo è quello di verificare la proprietà di numero primo per un numero intero n, che si supponde essere un probabile numero primo dopo aver effettuato un test di tipo probabilistico, come ad esempio il testi di Miller-Rabin in [5].

Teorema 2.1.1. *Sia $n \in \mathbb{Z}$, supponiamo che esistano $a \in \mathbb{Z}/n\mathbb{Z}$ e un esponente s intero tali che:*

i. $a^s = 1$;

ii. $a^{\frac{s}{q}} - 1 \in (\mathbb{Z}/n\mathbb{Z})^*$ *per ogni divisore primo q di s.*

Allora ogni intero primo che divide n è congruo a 1 modulo s. In particolare se $s > \sqrt{n}$, allora n è primo.

Dimostrazione. Sia $p \in \mathbb{N}$ primo con $p \mid n$. Mostriamo che $a' \in \mathbb{Z}/p\mathbb{Z}$ definito come $a' \coloneqq a \pmod{p}$ è una unità di ordine s.
Per ipotesi i, esiste $k \in \mathbb{Z}$ tale che $a^s = 1 + nk = 1 + \left(\frac{n}{p}p\right)k$, cioè $(a')^s \equiv 1 \pmod{p}$.
Per ipotesi ii, abbiamo che $a^{\frac{s}{q}} - 1$ è invertibile in $\mathbb{Z}/n\mathbb{Z}$, dunque vale che

$$\gcd(a^{\frac{s}{q}} - 1, n) = 1;$$

in particolare si ha che ogni p primo con $p \mid n$ soddisfa

$$\gcd(a^{\frac{s}{q}} - 1, p) = 1.$$

Abbiamo dunque mostrato che $(a')^{\frac{s}{q}} \not\equiv 1 \pmod{p}$, pertanto a' è una unità di ordine s in $\mathbb{Z}/p\mathbb{Z}$. Ne viene che

$$s \mid |U_p| = p - 1 \implies p \equiv 1 \pmod{s}.$$

Questo dimostra la prima parte del teorema. La seconda parte del teorema è ovvia: se $s > \sqrt{n}$, allora ogni divisore primo di n deve essere maggiore di $\sqrt{n}$, dunque n è primo. □

Osservazione 2.1.1 Potremmo utilizzare il teorema sopra riportato per verificare la primalità di un numero intero n. Si può procedere così facendo:

1. Si verifica la condizione *i.* di Teorema 2.1.1 cercando un inero s tra i divisori di $n-1$, scegliendo un qualunque $x \in \mathbb{Z}/n\mathbb{Z}$ e definendo $a \coloneqq x^{\frac{n-1}{s}}$. In questo modo, per il Teorema di Fermat vale $a^s = x^{n-1} = 1$.

2. Il modo più semplice per soddsifare la condizione *ii.* di Teorema 2.1.1 è quello di avere s primo, in modo da non avere proprio nulla da verificare. Possiamo effettuare un test probabilistico su s e poi ripetere ricorsivamente la procedura.

3. Se infine $s > \sqrt{n}$, allora abbiamo concluso.

Il passo 1 chiede di saper fattorizzare $n - 1$, che in generale è un problema difficile. Possiamo però sperare di trovare un divisore "piccolo" di $n - 1$, che chiamo r, e scegliere $s \coloneqq \frac{n-1}{r}$ e sperare che s sia un intero primo (o probabile primo, per usare ricorsivamente la tecnica).
Schoof in [4] afferma che la probabilità che $n - 1$ si fattorizzi in questo modo è in media $\mathcal{O}\left(\frac{1}{\log n}\right)$, di conseguenza procedere in questa direzione ci fornisce una scarsa probabilità di successo.

Una soluzione per risolvere i problemi posti nell'osservazione precedente è quello di utilizzare un gruppo abeliano generato da una curva ellittica. Come nel capitolo precedente, indichiamo con $E_{A,B}(\mathbb{Z}/n\mathbb{Z})$ una curva ellittica costruita su un gruppo finito e di parametri A e B. In alternativa, scriviamo più semplicemente $E(\mathbb{Z}/n\mathbb{Z})$ se non è importante specificarne i parametri. Se per caso $\mathbb{Z}/n\mathbb{Z}$ non dovesse essere un campo, allora potrebbe fallire la proprietà di gruppo per $E(\mathbb{Z}/n\mathbb{Z})$ e se accidentalmente ciò dovesse succedere, allora si ha che n non è un numero primo senza dover dimostrare altro.

Enunciamo ora un risultato chiave nell'algortimo di Goldwasser-Kilian. È un risultato analogo a Teorema 2.1.1.

Teorema 2.1.2. *Sia $n \in \mathbb{N}$ con $2 \nmid n$, $3 \nmid n$. Siano $A, B \in \mathbb{Z}$ tali che $\gcd(4A^3 + 27B^2, n) = 1$. Supponiamo che esistano un punto $L \in E_{A,B}(\mathbb{Z}/n\mathbb{Z}) \setminus \{\infty\}$ e un intero s tali che:*

i. $sL = \infty$,

ii. $\frac{s}{q}L \neq \infty$ *per ogni divisore primo q di s.*

Allora ogni numero primo p che divide n soddisfa $|E_{A,B}(\mathbb{Z}/p\mathbb{Z})| \equiv 0 \pmod{s}$. In particolare, se $s > (\sqrt[4]{n} + 1)^2$, allora n è primo.

Dimostrazione. Sia p primo divisore di n. Sia $L_p \in E_{A,B}(\mathbb{Z}/p\mathbb{Z})$ definito come in Lemma 1.2.2. Mostriamo che che L_p ha ordine s.
Per ipotesi *i.* e per il Lemma 1.2.2 abbiamo che

$$sL_p = (sL)_p = \infty_p = \infty.$$

Per le ipotesi *ii.* e per il Lemma 1.2.2 abbiamo che

$$\frac{s}{q}L_p = \left(\frac{s}{q}L\right)_p \neq (\infty)_p$$

e di conseguenza L_p ha ordine s. Dato che L_p ha ordine s, deve essere che $|E_{A,B}(\mathbb{Z}/p\mathbb{Z})|$ è un multiplo di s, cioè

$$|E_{A,B}(\mathbb{Z}/p\mathbb{Z})| \equiv 0 \pmod{s}$$

e questo dimostra la prima parte del teorema.
Per mostrare la seconda partedel teorema, utilizzando la Disuguaglianza di Hasse abbiamo che:

$$\left(\sqrt{p}+1\right)^2 \geq |E_{A,B}(\mathbb{Z}/p\mathbb{Z})| \geq s > \left(\sqrt[4]{n}+1\right)^2,$$

ne viene che $p > \sqrt{n}$, dunque n è primo. □

L'idea ora è quella di utilizzare questo teorema nella stessa maniera cui volevamo usare il teorema precedente.

Vogliamo verificare la primalità di un probabile primo p. Costruiamo delle curve ellittice $E(\mathbb{Z}/p\mathbb{Z})$ fino a quando non ne otteniamo una di ordine $|E(\mathbb{Z}/p\mathbb{Z})| = 2q$, in cui $q \in \mathbb{Z}$ è un probabile primo.

Preserviamo in memoria A, B, q, con q grande circa la metà di p, e ripetiamo ricorsivamente il procedimento su q, ottenendo così una successione di probabili primi $\{p_i\}_{i=1\ldots n}$ in cui l'ultimo termine p_n è sufficientemente piccolo ed è possibile determinarne la primalità per via diretta in un tempo breve.

Se l'ultimo elemento trovato p_n è effetivamente primo, allora tutta la successione $\{p_i\}$ contiene elementi primi per il Teorema 2.1.2.
Se invece l'ultimo elemento trovato p_n risulta non essere primo, l'algoritmo fallisce e non posso trarre alcuna conclusione su p.

2.2 Descrizione dell'algoritmo

Esponiamo gli algoritmi utilizzati dal test di primalità basato su curve ellittiche descritto da Goldwasser e Kilian. L'algoritmo finale vuole utilizzare Teorema 2.1.2 e l'unico input di cui necessita è un numero intero p, estratto da un algoritmo di tipo probabilistico. Come osserveremo in seguito, questo algoritmo è in grado di produrre un certificato di primalità per p.
Presentiamo GENERATE-CURVE, SELECT-POINT, PROVE-PRIME e CHECK utilizzando pseudocodice di alto livello e di tipo informale. Nella sezione successiva, forniamo una decrizione ed una analisi di ognuno degli algoritmi qui mostrati.

GENERATE-CURVE (p)

1: Genera in maniera casuale con distribuzione uniforme A e B fino a quando $\gcd(4A^3 + 27B^2, p) = 1$.
2: Computa $|E_{A,B}(\mathbb{F}_P)|$ usando l'algoritmo di Schoof. Se $|E_{A,B}(\mathbb{F}_P)|$ è dispari, torna al passo 1.
3: Poni $q \coloneqq |E_{A,B}(\mathbb{F}_P)|/2$. Se q è divisibile per 2 o per 3, torna al passo 1.
4: Verifica se q è un probabile primo usando un test probabilistico. Se non è primo, torna a 1.
5: **return** $((A, B), p)$.

Algorithm 1: GENERATE-CURVE (p). *Algoritmo per generare una curva di ordine* $2q$*, con* q *primo.*

SELECT-POINT (p, q, (A, B))

1: Scelgo x uniformemente in $\mathbb{Z}/p\mathbb{Z}$ fino a quando $z = x^3 + Ax + B$ è un residuo quadratico.
2: Calcolo $y = \sqrt{z}$, scegliendo casualmente quale radice prendere, e pongo $L \coloneqq (x, y)$.
3: Calcola qL. Se $qL \neq \infty$, torno al passo 1.
4: **return** L.

Algorithm 2: SELECT-POINT(p, q, (A, B)). *Algoritmo per scegliere un punto di ordine* q *su una curva ellittica descritta dai parametri* A *e* B*.*

PROVE-PRIME (p)

1: $i \leftarrow 0$.
2: $p_0 \leftarrow p$.
3: LOWERBOUND $\leftarrow \max\left\{2^{k^{\left(C/\log(\log(k))\right)}}, 37\right\}$, dove p è lungo k bits.
4: **while** $p_i >$ LOWERBOUND **do**
5: $\quad (A_i, B_i), p_{i+1} \leftarrow$ GENERATE-CURVE (p_i).
6: $\quad L_i \leftarrow$ SELECT-POINT $(p_i, p_{i+1}, (A_i, B_i))$.
7: $\quad i \leftarrow i + 1$.
8: $\quad$ **if** $2 \mid p_i$ **or** $3 \mid p_i$ **then goto 1**
9: **if** p_i is prime **then**
10: $\quad$ **return** $((A_0, B_0), L_0, p_0, ..., (A_{i-1}, B_{i-1}), L_{i-1}, p_i)$.
11: **else**
12: $\quad$ **goto 1**
13: Se ad un qualunque punto dell'esecuzione dell'algoritmo ho eseguito più di $p^{\log(\log(p))}$ passi, termina l'esecuzione.

Algorithm 3: PROVE-PRIME (p). *Algortimo del Test di Primalità.*

CHECK $(p, ((A_0, B_0), L_0, p_1, ..., (A_{i-1}, B_{i-1}), L_{i-1}, p_i))$

1: **if** $p_i > \max\left\{2^{k \exp\left\{{}^{C}/_{\log(\log(k))}\right\}}, 37\right\}$ **then**
2: **return false**
3: $p_0 \leftarrow p$
4: **for** $j \coloneqq 0$ to $i-1$ **do**
5: **if** $p \nmid 2$ or $p \nmid 3$ **then return false**
6: **if** $\gcd(4A_j^3 + 27B_j^2, p_j) \neq 1$ **then return false**
7: **if** $p_{j+1} \leq \sqrt{p_j} + 2\sqrt[4]{p_j} + 1$ **then return false**
8: **if** $L_j = \infty_{p_j}$ or $p_{j+1}L_j \neq \infty_{p_j}$ **then return false**
9: **return true**

Algorithm 4: CHECK $(p, ((A_0, B_0), L_0, p_1, ..., (A_{i-1}, B_{i-1}), L_{i-1}, p_i))$. *Algoritmo per verificare un certificato di primalità.*

2.3 Analisi degli algoritmi utilizzati

Analizziamo e studiamo gli algoritmi sopra proposti.

Algoritmo 1 **GENERATE-CURVE** (p) crea casualmente due parametri A, B e computa la curva ellittica $E_{A,B}(\mathbb{Z}/p\mathbb{Z})$ fino a quando non riesce a generare una curva il cui ordine è $2q$, in cui q è un probabile primo.
Possiamo fare uno studio elementare sulla complessità computazione di questo algoritmo. Il passo 1 ha complesità computazionale $\mathcal{O}(1)$. L'algoritmo di Schoof impiegato al passo 2 ha complessità computazionale $\mathcal{O}\left(\log^8 p\right)$. La complessità del test presente al passo 4 dipende dall'algoritmo probabilistico scelto, ma solitamente è significativamente minore di $\mathcal{O}\left(\log^8 p\right)$.

Osserviamo che i passi 1 e 2 dell'algortimo sono ripetuti fino a quando non riusciamo a trovare una curva ellittica idonea il cui ordine è il doppio di un probabile primo. Per una discussione sul numero di curve che bisogna generare per soddisfare questa condizione, si vada alla sezione succesiva del documento.

Algoritmo 2 **SELECT-POINT** (p, q, (A, B)) cerca un punto L di ordine q sulla curva ellittica $E_{A,B}(\mathbb{Z}/p\mathbb{Z})$. Se l'argomento di SELECT-POINT è una curva creata utilizzando Algoritmo 1 GENERATE-CURVE, possiamo dire che $|E_{A,B}(\mathbb{Z}/p\mathbb{Z})| = 2q$ e utilizzare questo fatto per fare uno studio elementare sulla complessità computazionale di SELECT-POINT (p, q, (A, B)).
La complessità del passo 1 di SELECT-POINT è costante, infatti, usando la Disuguaglianza di Hasse, dobbiamo cercare in media un numero

$$\frac{2p}{\#E_{A,B}(\mathbb{F}_p)} \geq \frac{2p}{p - 2\sqrt{p} + 1} \in \mathcal{O}(1)$$

di valori x prima di trovare un residuo quadratico.
La complessità del passo 2 di SELECT-POINT è al più $\mathcal{O}\left(\log^4 p\right)$ nell'ipotesi in cui p è primo. Infatti nei casi in cui p è primo possiamo usare l'algoritmo *RESSOL* come descritto in [10] per individuare $y = \sqrt{x^3 + Ax + B}$. Nei casi in cui p non è primo, possiamo trovare una discussione su [11] su quale algoritmo

utilizzare e quale sia la sua complessità computazionale.
Il numero di volte in cui il valore x viene cercato dipende dal numero di volte in cui il test al passo 3 fallisce. L'ordine di L divide $|E_{A,B}(\mathbb{Z}/p\mathbb{Z})|$ e assumendo q primo si ha che $|L| \in \{2, q\}$. Dato che $E(\mathbb{F}_p) \cong \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$, metà degli elementi di $E(\mathbb{F}_p)$ hanno ordine 2 e l'altra metà ha ordine q, per cui in media è necessario cercare il valore x due volte.
Al passo 3 viene richiesto di calcolare qL. Possiamo calcolare rapidamente i punti qL, con $L \in E_{A,B}(\mathbb{F}_P)$ nello stesso modo con cui calcoliamo potenze di numeri elevati:

$$qL = \begin{cases} L, & \text{se } q = 1 \\ (L+L)\frac{q}{2}, & \text{se } 2 \mid q \\ L + (q-1)L, & \text{se } 2 \nmid q, q \neq 1 \end{cases}.$$

Seguendo questo procedimento, il calcolo di qL richiede $\mathcal{O}(\log q)$ addizioni. Utilizzando un algoritmo di addizione elementare, come esempio quello presentato in [1], ognuna di queste addzioni ha complessità computazionale $\mathcal{O}(\log^3 q)$.
La complessità computazionale di SELECT-POINT risulta quindi essere $\mathcal{O}(\log^4 q)$.

Osservazione 2.3.1 Possiamo sempre cercare i punti della curva $E_{A,B}(\mathbb{Z}/n\mathbb{Z})$ per un qualunque intero $n > 1$, tutta via non è necessariamente detto che si riesca a calcolare qL. Fallire nel calcolare qL significa che $E_{A,B}(\mathbb{Z}/n\mathbb{Z})$ non ha proprietà di gruppo e quindi certamente n non può essere primo.

Osserviamo che quando l'argomento di SELECT-POINT è una curva generata da GENERATE-CURVE, allora l'algoritmo ha come valore di ritorno una curva ellittica e un intero q in grado di soddisfare le ipotesi di Teorema 2.1.2 a meno di assumere la primalità di q. Ripetendo ricorsivamente l'algoritmo, come mostrato in Algoritmo 3 **PROVE-PRIME** (p), possiamo dimostrare la primalità di q e dunque anche di p, concludendo così il nostro test. PROVE-PRIME (p), oltre a ripetere ricorsivamente GENERATE-CURVE e SELECT-POINT, genera anche un certificato di primalità per p, cioè un insieme di dati che consente a chiunque di verificare rapidamente la primalità di suddetto intero p.
Osserviamo che gli algoritmi GENERATE-CURVE e SELECT-POINT sono ripetuti fino a quando non viene ottenuto un termine sufficientemente piccolo p_i tale da verificare in tempo polinomiale la sua primalità. Il termine costante LOWERBOUND rappresenta la soglia entro il quale non ho più bisogno di ripetere ricorsivamente questi algoritmi. Il termine costante C di LOWERBOUND non è univocamente determinato, ma dipende dal test deterministico scelto. Come mostrato in [1], una tale costante esiste sempre; una analisi di tale costante è oltre gli interessi e gli obiettivi di questo documento.

Osservazione 2.3.2 Abortire il programma al passo 13 di PROVE-PRIME significa che l'algoritmo non è stato in grado di trovate un punto della curva in grado di soddisfare Teorema 2.1.2 in un lasso ragionevole di tempo. Un evento di questo tipo potrebbe accadere, ad esempio, nel raro caso cui il test probabilistico presente in GENERATE-CURVE non sia in grado di riconoscere un numero composto come tale. In questa situazione non saremmo in grado di terminare il ciclo definito al passo 4 di PROVE-PRIME, di conseguenza necessitiamo di dover arrestare forzatamente l'esecuzione del programma.

L'Algoritmo finale 4 **CHECK** $(p, ((A_0, B_0), L_0, p_1, ..., (A_{i-1}, B_{i-1}), L_{i-1}, p_i))$ è atto a dimostrare rapidamente che un certificato di primalità generato da

Prove-Prime (p) è valido. Dimostraimo qui che l'algoritmo Check è in grado di provare la primalità di p per un certificato di primalità generato da Prove-Prime.

Proposizione 2.3.1. *Se Algoritmo 4*
Check $(p, ((A_0, B_0), L_0, p_1, ..., (A_{i-1}, B_{i-1}), L_{i-1}, p_i))$ *accetta l'input* p *e* $((A_0, B_0), L_0, p_1, ..., (A_{i-1}, B_{i-1}), L_{i-1}, p_i)$ *è un certificato di primalità generato da Algoritmo 3* Prove-Prime *(p), allora* p *è primo.*

Dimostrazione. Supponiamo che la funzione Algoritmo 4 Check riceva dati di input di tipo corretto, cioè

$$p_i < \max\left\{2^{k \exp\left\{{}^{C}/_{\log(\log(k))}\right\}}, 37\right\}$$

Allora p_i è primo, infatti Teorema 2.1.2 mi garantisce che p_{j+1} primo $\implies p_j$ primo e per induzione concludo che $p_0 = p$ primo.
Dobbiamo ora verificare che l'input di Algoritmo 4 Check sia un certificato di primalità.

i. Per definizione di Algoritmo 3 Prove-Prime, p_i è primo.

ii. Per definizione di Algoritmo 1 Generate-Curve, vale che $\gcd(4A_j^3 + 27B_j^2, p_j) = 1$.

iii. Per definizione di Algoritmo 1 Generate-Curve, usando la disuguaglianza di Hasse
$$|E_{A,B}(\mathbb{Z}/p_j\mathbb{Z})| \geq p_j - 2\sqrt{p_j} + 1$$
ottengo che, nel caso $p_j > 37$
$$p_{j+1} \geq \frac{1}{2}\left(p_j + 2\sqrt{p_j}\right) > \sqrt{p_j} + 2\sqrt[4]{p_j} + 1.$$

iv. Per definizione di Algoritmo 3 Prove-Prime ho che $p_j > 37$.

v. Per deifnizione di Algoritmo 2 Select-Point, ho che $L_j \neq \infty_{p_j}$ e $p_{j+1}L_j = \infty_{p_j}$.

Dunque Algoritmo 4 Check accetta sempre l'imput desiderato. □

Per ciò che riguarda la complessità computazionale di Check, è facile mostrare che l'algoritmo termina in $\mathcal{O}\left(\log^4 p\right)$ passi. Nella definizione data di Check, ripeto $i = \mathcal{O}(\log p)$ volte una serie di condizioni. Di queste condizioni, quella che ha complessità computazionale maggiore è presente al passo 8 ed è $\mathcal{O}\left(\log^3 p\right)$.

2.4 Complessità computazionale e considerazioni

Per studiare la complessità computazionale dell'algoritmo, ci avvarremo del seguente Teorema presentato in [6] e di una congettura sulla distribuzione dei numeri primi in un intervallo breve.

Teorema 2.4.1. ***(Distribution of order of Elliptic Curves)*** *Sia $p > 5$ primo e sia*

$$S \subseteq \left[p + 1 - \lfloor\sqrt{p}\rfloor, p + 1 + \lfloor\sqrt{p}\rfloor\right].$$

Siano A, B scelti casualmente su $\mathbb{F}_p$. Allora esiste una costante k tale che:

$$prob\left(\#E_{A,B}\left(\mathbb{F}_p\right) \in S\right) > \frac{k}{\log p} \cdot \frac{|S| - 2}{2\lfloor\sqrt{p}\rfloor + 1}. \tag{2.1}$$

Il teorema ci dice che, se $|S| > 2$ è costruito in maniera tale da soddisfare una proprietà P, cioè

$$S := \{x \in \mathbb{N} : P[x]\}$$

allora l'ordine di una curva ellittica scelta casualmente è almeno $\mathcal{O}\left({}^{1}/_{\log p}\right)$ volte in grado di soddisfare quella proprietà P rispetto ad un singolo elemento scelto casualmente nell'intervallo.
Mostriamo come utilizzare questo teorema per stimare la complessità computazionale di Prove-Prime.

Dato che $p_{j+1} \approx p_j/2$, ci aspettiamo che il ciclo presente in Algoritmo 3 Prove-Prime venga iterato $i = \mathcal{O}(\log p)$ volte. Per ognuna di queste iterazioni, dobbiamo individuare il tempo di esecuzione impiegato dagli Algoritmi 1 Generate-Curve e 2 Select-Point.

Abbiamo visto nella sezione precedente che l'Algoritmo di Schoof per generare una curva ellittica, usato in Algoritmo 1 Generate-Curve, ha complessità computazionale di $\mathcal{O}\left(\log^8 p\right)$, ciò significa che Generate-Curve ha complessità computazionale $T_{p_j} \cdot \mathcal{O}\left(\log^8 p\right)$, dove T_{p_j} è il numero di curve da testare prima di soddisfare la condizione 3 di Generate-Curve e p_j è l'input dell'algoritmo al passo j-esimo.
Cerchiamo ora di stimare T_{p_j}, fornendone una maggiorazione usando un immediato corollario al Teorema di Lenstra 2.4.1.

Corollario 2.4.2. *Sia $p > 5$ primo, sia*

$$S(p) := \left\{q \mid q \in \left[\frac{p + 1 - \lfloor\sqrt{p}\rfloor}{2}, \frac{p + 1 + \lfloor\sqrt{p}\rfloor}{2}\right], q \text{ primo}\right\}.$$

Siano (A, B) scelti in maniera uniforme su $\mathbb{Z}/n\mathbb{Z}$. Allora

$$prob\left(\#E_{A,B}\left(\mathbb{F}_p\right) \text{ è doppio di un primo}\right) > \frac{k}{\log p} \cdot \frac{|S(p)| - 2}{2\lfloor\sqrt{p}\rfloor + 1} \tag{2.2}$$

per qualche costante k.

Dimostrazione. Esiste una ovvia biezione tra gli elementi di $S(p)$ e gli elementi nell'intervallo

$$\left[p + 1 - \lfloor\sqrt{p}\rfloor, p + 1 + \lfloor\sqrt{p}\rfloor\right]$$

che sono doppio di un numero primo. Applichiamo quindi Teorema 2.4.1 e concludiamo immediatamente. □

A meno di essere in grado di stimare $|S(p)|$, il corollario appena enunciato ci fornisce una maggiorazione sulla probabilità di successo del test effettuato in Algoritmo 1 GENERATE-CURVE. Per stimare $|S(p)|$, usiamo la seguente congettura.

Congettura 2.4.1. ***(Distribution of primes in short interval)*** *Sia*

$$S(p) \coloneqq \{q : p + 1 - \lfloor\sqrt{p}\rfloor \leq 2q \leq p + 1 + \lfloor\sqrt{p}\rfloor\,, \quad q \text{ primo } \}$$

Insieme dei primi in un breve intervallo di $p/2$. *Allora vale che*

$$|S(p)| = \mathcal{O}\left(\frac{\sqrt{p}}{\log^c p}\right)$$

Per qualche c *costante.*

Applichiamo la congettura per calcolare la stima fornita da Corollario 2.4.2

Corollario 2.4.3. ***(Distribution of Elliptic Curves in short interval)*** *Sia* $p > 5$ *primo, siano* A, B *scelti uniformemente su* $\mathbb{F}_p$. *Sia* $S(p)$ *definito come nella Congettura 2.4.1. Allora vale che*

$$prob\left(\#E_{A,B}\left(\mathbb{F}_p\right) \in S(p)\right) = \Omega\left(\frac{1}{log^{c+1}p}\right) \tag{2.3}$$

dove c *è la stessa costante della Congettura 2.4.1.*

Dimostrazione. Applicando Teorema 2.4.1 e ponendo $S = S(p)$ si ottiene, per alcune costanti k, k_1, k_2

$$\begin{aligned}\left(\#E_{A,B}\left(\mathbb{F}_p\right) \in S(p)\right) &> \frac{k}{\log p} \cdot \frac{|S(p)| - 2}{2\lfloor\sqrt{p}\rfloor + 1} > \\ &> \frac{k}{\log p} \cdot \frac{k_1\sqrt{p}}{\log^c p} \frac{1}{2\lfloor\sqrt{p}\rfloor + 1} > \\ &> \frac{k_2}{\log^{c+1} p}\end{aligned}$$

che conclude la dimostrazione. □

Abbiamo ora gli strumenti necessari per calcolare T_{p_j}. Utilizzando il reciproco della maggioranzione presentata in Corollario 2.4.3, possiamo affermare che il numero medio T_{p_j} di curve da testare è $\mathcal{O}(\log^{c+1} p)$. Ne consegue che il tempo *medio* di esecuzione di Algoritmo 1 GENERATE-CURVE risulta essere $\mathcal{O}(\log^{c+1} p)\mathcal{O}(\log^8 p) = \mathcal{O}(\log^{c+9} p)$

Ricordiamo che nella sezione precedente abbiamo mostrato che la complessità computazionale di Algoritmo 2 SELECT-POINT è $\mathcal{O}(\log^4 q)$, perciò concludiamo che Algoritmo 1 GENERATE-CURVE esegua in *media* $\mathcal{O}\left(\log^{k+9} p\right)$ passi e Algoritmo 3 PROVE-PRIME compie in *media* $\mathcal{O}\left(\log^{k+10} p\right)$ passi.

Appendices

Appendice A

Test di primalità proabilistici

Nel documento, abbiamo spesso parlato di numeri *probabili primi*, limitandoci a dire che sono numeri interi e che si può mostrare essere probabili primi in un tempo ragionevolmente limitato di tempo. In questa sezione di appendice, specifichiamo un metodo per cercare numeri probabili primi attraverso il test di Miller-Rabin.

Partiamo dal Piccolo Teorema di Fermat, sicuramente noto al lettore e che riportiamo per semplicità.

Teorema A.0.1. ***(Fermat, Piccolo)*** *Sia $p \in \mathbb{N}$ primo, sia $a \in \mathbb{N}$ tale che $\gcd(p, a) = 1$. Allora vale che:*

$$a^{p-1} \equiv 1 \pmod{p}.$$

Dato che ogni numero primo soddisfa il teorema di Fermat piccolo, se un numero intero n fallisce il teorema di Fermat, allora abbiamo motsrato che n sicuramente non è primo. Se invece un intero n *soddisfa* il teorema il Fermat, non possiamo concludere che n sia primo. Chiamiamo un tale numero *pseudoprimo di Fermat*.

Definizione A.0.1. ***(Pseudoprimo di Fermat)*** *Sia $N \in \mathbb{N}$ dispari. N è detto pseudoprimo di Fermat nella base $a \in \mathbb{N}$ se vale che:*

$$a^{N-1} \equiv 1 \pmod{N}.$$

Osservazione A.0.1 Per rendere un test di Fermat più accurato, possiamo utlizzare più basi a e verificare se il numero N oggetto del nostro test è pseudprimo in tutte queste basi. In questo modo possiamo rilevare che quasi tutti i numeri composti hanno qualche base a in cui non sono pseudoprimi, cioè non si comportano come numeri primi. Tuttavia ciò non è sempre vero: esistono numeri *non* primi in grado di soddisfare l'enuncito del teorema di Fermat in ogni base. Questi numeri sono detti *numeri di Carmichael*. Alcuni esempi di numeri di Carmichael sono $561, 1105$ e 1729.

Osservazione A.0.2 Consideriamo N pseudoprimo di Fermat dispari. Sappiamo che N soddisfa il teorema di Fermat per qualche base a, cioè vale che

$$a^{N-1} \equiv 1 \pmod{N}.$$

Dato che N è dispari, esiste M intero tale che $N-1=2M$, cioè

$$a^{2M} \equiv 1 \pmod{N}.$$

Ne viene dunque che

$$\begin{aligned} a^{2M} \equiv 1 \pmod{N} &\iff a^{2M}-1 \equiv 0 \pmod{N} \iff \\ &\iff \left(a^M+1\right)\left(a^M-1\right) \equiv 0 \pmod{N}. \end{aligned}$$

Se assumiamo N primo, abbiamo che per definizione di primalità N divide $\left(a^M+1\right)$ oppure $\left(a^M-1\right)$ ma non entrambi. Difatti se così fosse, avrei che N divide $\left(a^M+1\right)-\left(a^M-1\right)=2$ negando le ipotesi iniziali.
In questo modo abbiamo mostrato che per i numeri primi N dispari vale che

$$a^{(N-1)/2} \equiv \pm 1 \pmod{N}$$

per ogni base a coprima ad N.
Analogamente a prima, potremmo avere dei numeri composti in grado di soddisfare questa richiesta per qualche o per ogni base a. Raccogliamo i numeri primi e i numeri composti in grado di soddisfare questa proprietà con il nome di *pseudoprimi di Eulero*.

Definizione A.0.2. ***(Pseudoprimo di Eulero)*** *Sia $N \in \mathbb{N}$ dispari. N è detto pseudoprimo di Eulero nella base $a \in \mathbb{N}$ se vale che:*

$$a^{(N-1)/2} \equiv \pm 1 \pmod{N}$$

Osservazione A.0.3 Il percorso dimostrativo seguito nell'osservazione precedente mostra che un intero pseudprimo di Eulero è necessariamente un pseudoprimo di Fermat.

$$\{ \text{ Pseudoprimi di Eulero } \} \subset \{ \text{ Pseudoprimi di Fermat } \}$$

Ne viene che testare la proprietà che definisce gli pseudoprimi di Eulero ci consente di individuare più facilmente alcuni numeri composti tra gli pseudoprimi di Fermat.

Osservazione A.0.4 Possiamo approfondire l'idea precedente che ci ha portato a definire i numeri pseudoprimi di Eulero, questa volta raccogliendo *tutte* le potenze di 2 presenti in $N-1$.
Sia N pseudoprimo di Eulero nella base a. Siano M, b interi tali che $N-1=2^bM$ e con b tale che $2 \nmid M$. Per il teorema di Fermat, vale che

$$a^{2^bM} \equiv 1 \pmod{N}.$$

Ne viene dunque che

$$\begin{aligned} a^{2^bM} \equiv 1 \pmod{N} &\iff a^{2^bM}-1 \equiv 0 \pmod{N} \iff \\ &\iff \left(a^{2^{b-1}M}+1\right)\left(a^{2^{b-1}M}-1\right) \equiv 0 \pmod{N} \iff \\ &\iff \left(a^{2^{b-1}M}+1\right)\left(a^{2^{b-2}M}+1\right)\left(a^{2^{b-2}M}-1\right) \equiv 0 \pmod{N} \iff \\ &\vdots \\ &\iff \left(a^{2^{b-1}M}+1\right)\left(a^{2^{b-2}M}+1\right)\ldots\left(a^M+1\right)\left(a^M-1\right) \equiv 0 \pmod{N}. \end{aligned}$$

Se N è primo, allora divide almeno uno tra i fattori nella formula precedente. Ancora una volta, può essere che un numero composto sia in grado di soddisfare questa proprietà. In questo modo ricaviamo la definizione di *pseudoprimo forte*.

Definizione A.0.3. ***(Pseudoprimo forte)*** *Sia $N \in \mathbb{N}$ dispari. Siano M, b interi tali che $N - 1 = 2^b M$ e con b tale che $2 \nmid M$. N è detto pseudoprimo forte nella base $a \in \mathbb{N}$ se vale che:*

- $a^M \equiv 1 \pmod{N}$ *oppure*
- $a^{2^r M} \equiv -1 \pmod{N}$ *per qualche* $r \in \{0 \dots b-1\}$

Osservazione A.0.5 Analogamente a prima, abbiamo che

$$\{ \text{ Pseudoprimo forte } \} \subset \{ \text{ Pseduoprimo di Eulero } \}$$

e dunque la richiesta di pseduoprimo forte consente di individuare più numeri composti rispetto alla richiesta di pseudoprimo di Eulero. Questa richiesta viene usata per definire il *test di Miller-Rabin*, che si limita a verificare la richiesta di pseudoprimo forte in alcune basi $a < N$. Maggiore è il numero di basi testate e migliore è l'affidabilità del test.

Bibliografia

[1] Goldwasser S.; Klian J. - *Primality Testing Using Elliptic Curves*, Journal of the ACM, Vol. 46, No. 4, July 1999.

[2] Washington L. C. -*Elliptic Curves: Number Theory and Cryptography*, Chapman & Hall/CRC, May 2003.

[3] Silverman J. H. - *The Arithmetic of Elliptic Curves*, Springer, 2009.

[4] Schoof R. - *Four primality testing algorithms*, Amsterdam, February 2008.

[5] Rabin - *Probabilistic algorithm for testing primality*, Journal of Number Theory, **12**, 1980.

[6] Lenstra H. W. - *Factoring integers with elliptic curves*, Ann. of Math, 1987.

[7] Li K. - *An Overview of Elliptic Curve Primality Proving*, December 2011.

[8] Riesel H. - *Prime Numbers and Computer Methods for Factorization*, Springer, 1985.

[9] N. J. A. Sloane, editor - *The On-Line Encyclopedia of Integer Sequences*, published electronically at https://oeis.org .

[10] Niven I., Zuckerman H. S., Montgomery H. L. - *An Introduction to the Theory of Numbers*, Wiley, January 1991.

[11] Adleman L., Manders K., Miller G. - *On Taking Roots in Finite Fields*, 18th Annual Symposium on Foundations of Computer Science, October 1977.